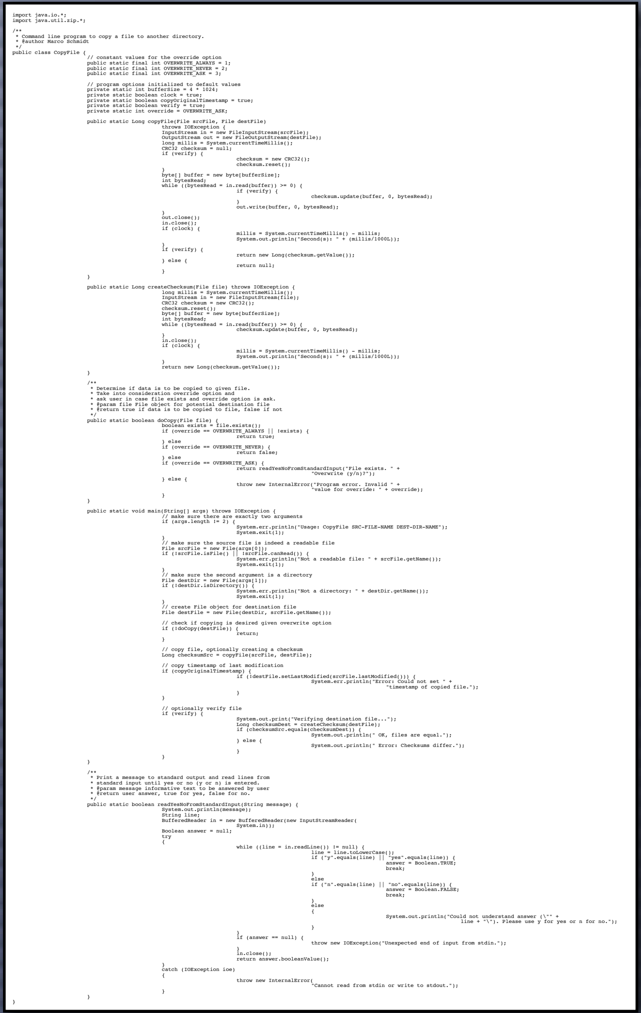
Application-Specific Models and Pointcuts using a Logic Meta Language

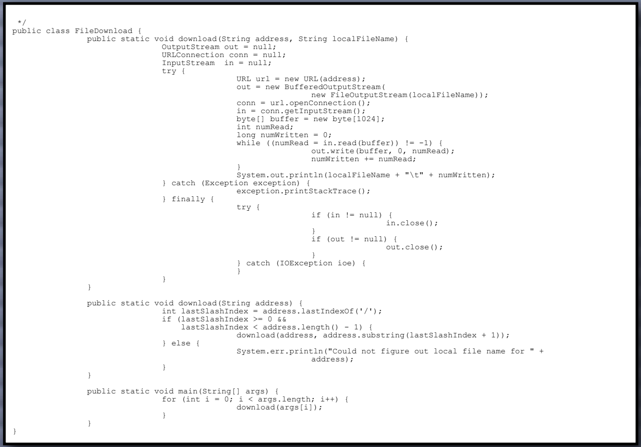
Johan Brichau Andy Kellens Kris Gybels Kim Mens

Robert Hirschfeld Theo D'Hondt

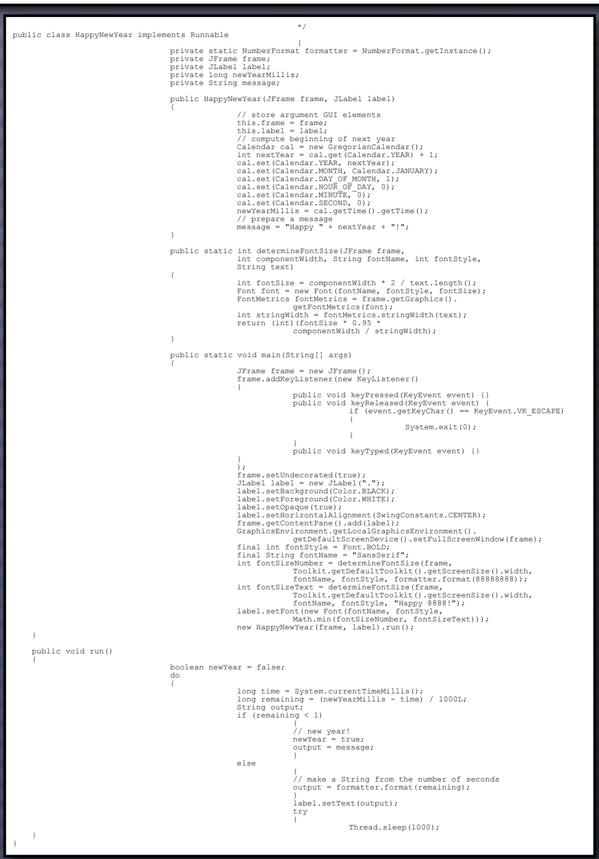
Crosscutting Concerns



```java
import java.io.*;
import java.util.zip.*;

/**
 * Command line program to copy a file to another directory.
 * @author Marco Schmidt
 */
public class CopyFile {
        // constant values for the override option
        public static final int OVERWRITE_ALWAYS = 1;
        public static final int OVERWRITE_NEVER = 2;
        public static final int OVERWRITE_ASK = 3;

        // program options initialized to default values
        private static int bufferSize = 4 * 1024;
        private static boolean clock = true;
        private static boolean copyOriginalTimestamp = true;
        private static boolean verify = true;
        private static int override = OVERWRITE_ASK;

        public static Long copyFile(File srcFile, File destFile)
                throws IOException {
                InputStream in = new FileInputStream(srcFile);
                OutputStream out = new FileOutputStream(destFile);
                long millis = System.currentTimeMillis();
                CRC32 checksum = null;
                if (verify) {
                        checksum = new CRC32();
                        checksum.reset();
                }
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        if (verify) {
                                checksum.update(buffer, 0, bytesRead);
                        }
                        out.write(buffer, 0, bytesRead);
                }
                out.close();
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                if (verify) {
                        return new Long(checksum.getValue());
                } else {
                        return null;
                }
        }

        public static Long createChecksum(File file) throws IOException {
                long millis = System.currentTimeMillis();
                InputStream in = new FileInputStream(file);
                CRC32 checksum = new CRC32();
                checksum.reset();
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        checksum.update(buffer, 0, bytesRead);
                }
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                return new Long(checksum.getValue());
        }

        /**
         * Determine if data is to be copied to given file.
         * Take into consideration override option and
         * ask user in case file exists and override option is ask.
         * @param file File object for potential destination file
         * @return true if data is to be copied to file, false if not
         */
        public static boolean doCopy(File file) {
                boolean exists = file.exists();
                if (override == OVERWRITE_ALWAYS || !exists) {
                        return true;
                } else
                if (override == OVERWRITE_NEVER) {
                        return false;
                } else
                if (override == OVERWRITE_ASK) {
                        return readYesNoFromStandardInput("File exists. " +
                                "Overwrite (y/n)?");
                } else {
                        throw new InternalError("Program error. Invalid " +
                                "value for override: " + override);
                }
        }

        public static void main(String[] args) throws IOException {
                // make sure there are exactly two arguments
                if (args.length != 2) {
                        System.err.println("Usage: CopyFile SRC-FILE-NAME DEST-DIR-NAME");
                        System.exit(1);
                }
                // make sure the source file is indeed a readable file
                File srcFile = new File(args[0]);
                if (!srcFile.isFile() || !srcFile.canRead()) {
                        System.err.println("Not a readable file: " + srcFile.getName());
                        System.exit(1);
                }
                // make sure the second argument is a directory
                File destDir = new File(args[1]);
                if (!destDir.isDirectory()) {
                        System.err.println("Not a directory: " + destDir.getName());
                        System.exit(1);
                }
                // create File object for destination file
                File destFile = new File(destDir, srcFile.getName());

                // check if copying is desired given overwrite option
                if (!doCopy(destFile)) {
                        return;
                }

                // copy file, optionally creating a checksum
                Long checksumSrc = copyFile(srcFile, destFile);

                // copy timestamp of last modification
                if (copyOriginalTimestamp) {
                        if (!destFile.setLastModified(srcFile.lastModified())) {
                                System.err.println("Error: Could not set " +
                                        "timestamp of copied file.");
                        }
                }

                // optionally verify file
                if (verify) {
                        System.out.print("Verifying destination file...");
                        Long checksumDest = createChecksum(destFile);
                        if (checksumSrc.equals(checksumDest)) {
                                System.out.println(" OK, files are equal.");
                        } else {
                                System.out.println(" Error: Checksums differ.");
                        }
                }
        }

        /**
         * Print a message to standard output and read lines from
         * standard input until yes or no (y or n) is entered.
         * @param message informative text to be answered by user
         * @return user answer, true for yes, false for no.
         */
        public static boolean readYesNoFromStandardInput(String message) {
                System.out.println(message);
                String line;
                BufferedReader in = new BufferedReader(new InputStreamReader(
                        System.in));
                Boolean answer = null;
                try
                {
                        while ((line = in.readLine()) != null) {
                                line = line.toLowerCase();
                                if ("y".equals(line) || "yes".equals(line)) {
                                        answer = Boolean.TRUE;
                                        break;
                                }
                                else
                                if ("n".equals(line) || "no".equals(line)) {
                                        answer = Boolean.FALSE;
                                        break;
                                }
                                else
                                {
                                        System.out.println("Could not understand answer (\"" +
                                                line + "\"). Please use y for yes or n for no.");
                                }
                        }
                        if (answer == null) {
                                throw new IOException("Unexpected end of input from stdin.");
                        }
                        in.close();
                        return answer.booleanValue();
                }
                catch (IOException ioe)
                {
                        throw new InternalError(
                                "Cannot read from stdin or write to stdout.");
                }
        }
}
```



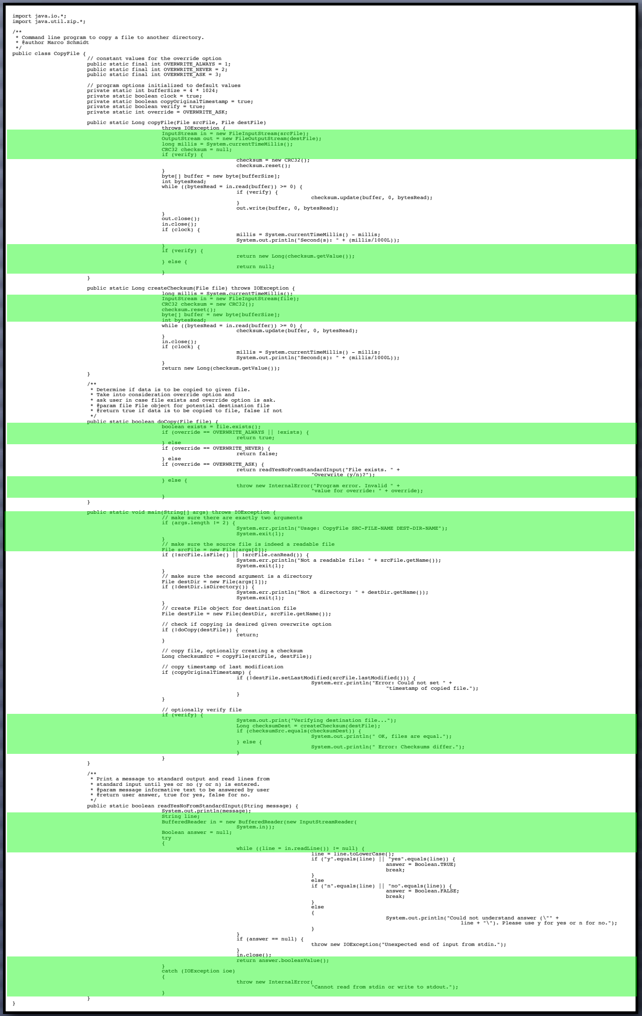
```java
 */
public class FileDownload {
        public static void download(String address, String localFileName) {
                OutputStream out = null;
                URLConnection conn = null;
                InputStream  in = null;
                try {
                        URL url = new URL(address);
                        out = new BufferedOutputStream(
                                new FileOutputStream(localFileName));
                        conn = url.openConnection();
                        in = conn.getInputStream();
                        byte[] buffer = new byte[1024];
                        int numRead;
                        long numWritten = 0;
                        while ((numRead = in.read(buffer)) != -1) {
                                out.write(buffer, 0, numRead);
                                numWritten += numRead;
                        }
                        System.out.println(localFileName + "\t" + numWritten);
                } catch (Exception exception) {
                        exception.printStackTrace();
                } finally {
                        try {
                                if (in != null) {
                                        in.close();
                                }
                                if (out != null) {
                                        out.close();
                                }
                        } catch (IOException ioe) {
                        }
                }
        }

        public static void download(String address) {
                int lastSlashIndex = address.lastIndexOf('/');
                if (lastSlashIndex >= 0 &&
                  lastSlashIndex < address.length() - 1) {
                        download(address, address.substring(lastSlashIndex + 1));
                } else {
                        System.err.println("Could not figure out local file name for " +
                                address);
                }
        }

        public static void main(String[] args) {
                for (int i = 0; i < args.length; i++) {
                        download(args[i]);
                }
        }
}
```



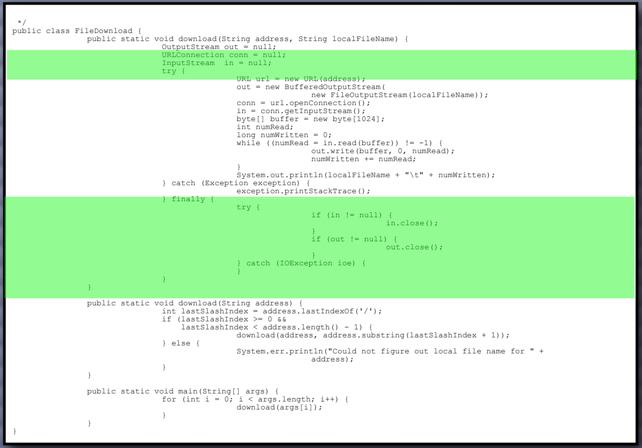
```java
                                                                 */
public class HappyNewYear implements Runnable
                                                                 {
                        private static NumberFormat formatter = NumberFormat.getInstance();
                        private JFrame frame;
                        private JLabel label;
                        private long newYearMillis;
                        private String message;

                        public HappyNewYear(JFrame frame, JLabel label)
                        {
                                // store argument GUI elements
                                this.frame = frame;
                                this.label = label;
                                // compute beginning of next year
                                Calendar cal = new GregorianCalendar();
                                int nextYear = cal.get(Calendar.YEAR) + 1;
                                cal.set(Calendar.YEAR, nextYear);
                                cal.set(Calendar.MONTH, Calendar.JANUARY);
                                cal.set(Calendar.DAY_OF_MONTH, 1);
                                cal.set(Calendar.HOUR_OF_DAY, 0);
                                cal.set(Calendar.MINUTE, 0);
                                cal.set(Calendar.SECOND, 0);
                                newYearMillis = cal.getTime().getTime();
                                // prepare a message
                                message = "Happy " + nextYear + "!";
                        }

                        public static int determineFontSize(JFrame frame,
                                int componentWidth, String fontName, int fontStyle,
                                String text)
                        {
                                int fontSize = componentWidth * 2 / text.length();
                                Font font = new Font(fontName, fontStyle, fontSize);
                                FontMetrics fontMetrics = frame.getGraphics().
                                        getFontMetrics(font);
                                int stringWidth = fontMetrics.stringWidth(text);
                                return (int)(fontSize * 0.95 *
                                        componentWidth / stringWidth);
                        }

                        public static void main(String[] args)
                        {
                                JFrame frame = new JFrame();
                                frame.addKeyListener(new KeyListener()
                                {
                                        public void keyPressed(KeyEvent event) {}
                                        public void keyReleased(KeyEvent event) {
                                                if (event.getKeyChar() == KeyEvent.VK_ESCAPE)
                                                {
                                                        System.exit(0);
                                                }
                                        }
                                        public void keyTyped(KeyEvent event) {}
                                }
                                );
                                frame.setUndecorated(true);
                                JLabel label = new JLabel(".");
                                label.setBackground(Color.BLACK);
                                label.setForeground(Color.WHITE);
                                label.setOpaque(true);
                                label.setHorizontalAlignment(SwingConstants.CENTER);
                                frame.getContentPane().add(label);
                                GraphicsEnvironment.getLocalGraphicsEnvironment().
                                        getDefaultScreenDevice().setFullScreenWindow(frame);
                                final int fontStyle = Font.BOLD;
                                final String fontName = "SansSerif";
                                int fontSizeNumber = determineFontSize(frame,
                                        Toolkit.getDefaultToolkit().getScreenSize().width,
                                        fontName, fontStyle, formatter.format(88888888));
                                int fontSizeText = determineFontSize(frame,
                                        Toolkit.getDefaultToolkit().getScreenSize().width,
                                        fontName, fontStyle, "Happy 8888!");
                                label.setFont(new Font(fontName, fontStyle,
                                        Math.min(fontSizeNumber, fontSizeText)));
                                new HappyNewYear(frame, label).run();
        }

        public void run()
        {
                        boolean newYear = false;
                        do
                        {
                                long time = System.currentTimeMillis();
                                long remaining = (newYearMillis - time) / 1000L;
                                String output;
                                if (remaining < 1)
                                        {
                                        // new year!
                                        newYear = true;
                                        output = message;
                                        }
                                else
                                        {
                                        // make a String from the number of seconds
                                        output = formatter.format(remaining);
                                        }
                                        label.setText(output);
                                        try
                                        {
                                                Thread.sleep(1000);
        }
}
```

Crosscutting Concerns

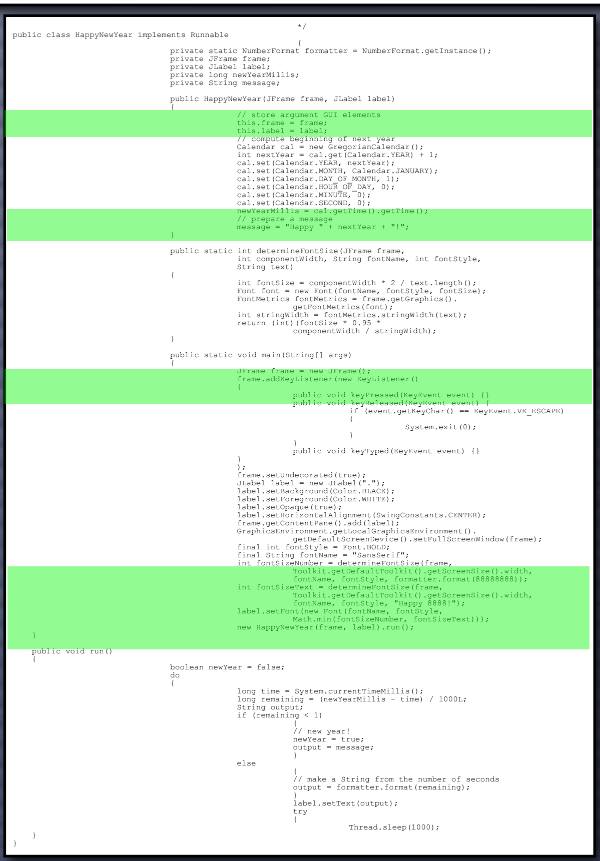
Synchronization



```java
import java.io.*;
import java.util.zip.*;

/**
 * Command line program to copy a file to another directory.
 * @author Marco Schmidt
 */
public class CopyFile {
        // constant values for the override option
        public static final int OVERWRITE_ALWAYS = 1;
        public static final int OVERWRITE_NEVER = 2;
        public static final int OVERWRITE_ASK = 3;

        // program options initialized to default values
        private static int bufferSize = 4 * 1024;
        private static boolean clock = true;
        private static boolean copyOriginalTimestamp = true;
        private static boolean verify = true;
        private static int override = OVERWRITE_ASK;

        public static Long copyFile(File srcFile, File destFile)
                throws IOException {
                InputStream in = new FileInputStream(srcFile);
                OutputStream out = new FileOutputStream(destFile);
                long millis = System.currentTimeMillis();
                CRC32 checksum = null;
                if (verify) {
                        checksum = new CRC32();
                        checksum.reset();
                }
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        if (verify) {
                                checksum.update(buffer, 0, bytesRead);
                        }
                        out.write(buffer, 0, bytesRead);
                }
                out.close();
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                if (verify) {
                        return new Long(checksum.getValue());
                } else {
                        return null;
                }
        }

        public static Long createChecksum(File file) throws IOException {
                long millis = System.currentTimeMillis();
                InputStream in = new FileInputStream(file);
                CRC32 checksum = new CRC32();
                checksum.reset();
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        checksum.update(buffer, 0, bytesRead);
                }
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                return new Long(checksum.getValue());
        }

        /**
         * Determine if data is to be copied to given file.
         * Take into consideration override option and
         * ask user in case file exists and override option is ask.
         * @param file File object for potential destination file
         * @return true if data is to be copied to file, false if not
         */
        public static boolean doCopy(File file) {
                boolean exists = file.exists();
                if (override == OVERWRITE_ALWAYS || !exists) {
                        return true;
                } else
                if (override == OVERWRITE_NEVER) {
                        return false;
                } else
                if (override == OVERWRITE_ASK) {
                        return readYesNoFromStandardInput("File exists. " +
                                "Overwrite (y/n)?");
                } else {
                        throw new InternalError("Program error. Invalid " +
                                "value for override: " + override);
                }
        }

        public static void main(String[] args) throws IOException {
                // make sure there are exactly two arguments
                if (args.length != 2) {
                        System.err.println("Usage: CopyFile SRC-FILE-NAME DEST-DIR-NAME");
                        System.exit(1);
                }
                // make sure the source file is indeed a readable file
                File srcFile = new File(args[0]);
                if (!srcFile.isFile() || !srcFile.canRead()) {
                        System.err.println("Not a readable file: " + srcFile.getName());
                        System.exit(1);
                }
                // make sure the second argument is a directory
                File destDir = new File(args[1]);
                if (!destDir.isDirectory()) {
                        System.err.println("Not a directory: " + destDir.getName());
                        System.exit(1);
                }
                // create File object for destination file
                File destFile = new File(destDir, srcFile.getName());

                // check if copying is desired given overwrite option
                if (!doCopy(destFile)) {
                        return;
                }

                // copy file, optionally creating a checksum
                Long checksumSrc = copyFile(srcFile, destFile);

                // copy timestamp of last modification
                if (copyOriginalTimestamp) {
                        if (!destFile.setLastModified(srcFile.lastModified())) {
                                System.err.println("Error: Could not set " +
                                        "timestamp of copied file.");
                        }
                }

                // optionally verify file
                if (verify) {
                        System.out.print("Verifying destination file...");
                        Long checksumDest = createChecksum(destFile);
                        if (checksumSrc.equals(checksumDest)) {
                                System.out.println(" OK, files are equal.");
                        } else {
                                System.out.println(" Error: Checksums differ.");
                        }
                }
        }

        /**
         * Print a message to standard output and read lines from
         * standard input until yes or no (y or n) is entered.
         * @param message informative text to be answered by user
         * @return user answer, true for yes, false for no.
         */
        public static boolean readYesNoFromStandardInput(String message) {
                System.out.println(message);
                String line;
                BufferedReader in = new BufferedReader(new InputStreamReader(
                        System.in));
                Boolean answer = null;
                try
                {
                        while ((line = in.readLine()) != null) {
                                line = line.toLowerCase();
                                if ("y".equals(line) || "yes".equals(line)) {
                                        answer = Boolean.TRUE;
                                        break;
                                }
                                else
                                if ("n".equals(line) || "no".equals(line)) {
                                        answer = Boolean.FALSE;
                                        break;
                                }
                                else
                                {
                                        System.out.println("Could not understand answer (\"" +
                                                line + "\"). Please use y for yes or n for no.");
                                }
                        }
                        if (answer == null) {
                                throw new IOException("Unexpected end of input from stdin.");
                        }
                        in.close();
                        return answer.booleanValue();
                }
                catch (IOException ioe)
                {
                        throw new InternalError(
                                "Cannot read from stdin or write to stdout.");
                }
        }
}
```



```java
 */
public class FileDownload {
        public static void download(String address, String localFileName) {
                OutputStream out = null;
                URLConnection conn = null;
                InputStream  in = null;
                try {
                        URL url = new URL(address);
                        out = new BufferedOutputStream(
                                new FileOutputStream(localFileName));
                        conn = url.openConnection();
                        in = conn.getInputStream();
                        byte[] buffer = new byte[1024];
                        int numRead;
                        long numWritten = 0;
                        while ((numRead = in.read(buffer)) != -1) {
                                out.write(buffer, 0, numRead);
                                numWritten += numRead;
                        }
                        System.out.println(localFileName + "\t" + numWritten);
                } catch (Exception exception) {
                        exception.printStackTrace();
                } finally {
                        try {
                                if (in != null) {
                                        in.close();
                                }
                                if (out != null) {
                                        out.close();
                                }
                        } catch (IOException ioe) {
                        }
                }
        }

        public static void download(String address) {
                int lastSlashIndex = address.lastIndexOf('/');
                if (lastSlashIndex >= 0 &&
                    lastSlashIndex < address.length() - 1) {
                        download(address, address.substring(lastSlashIndex + 1));
                } else {
                        System.err.println("Could not figure out local file name for " +
                                address);
                }
        }

        public static void main(String[] args) {
                for (int i = 0; i < args.length; i++) {
                        download(args[i]);
                }
        }
}
```

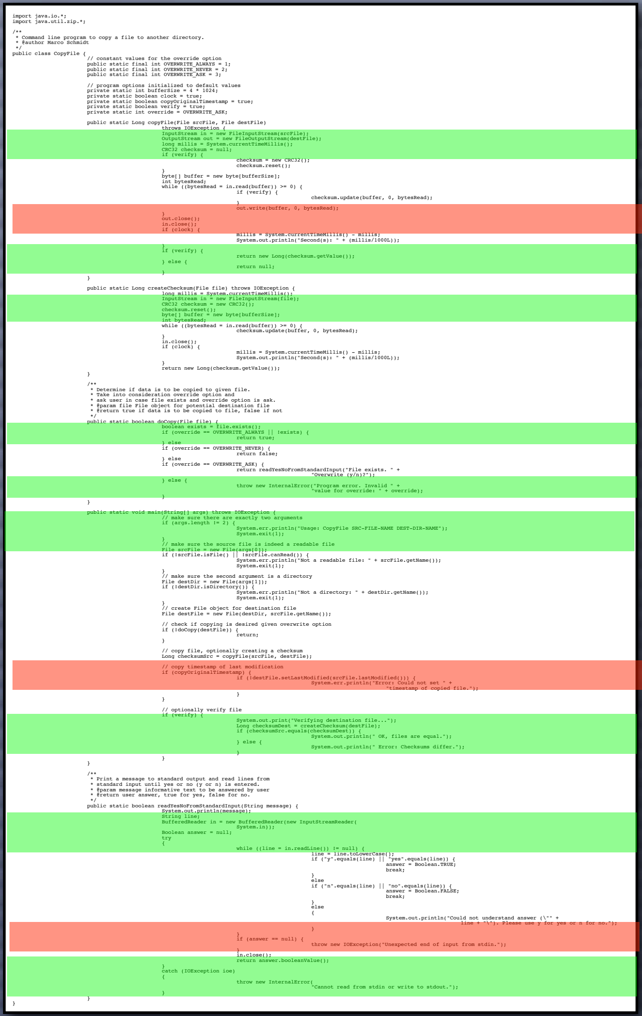


```java
                                        */
public class HappyNewYear implements Runnable
                                        {
                private static NumberFormat formatter = NumberFormat.getInstance();
                private JFrame frame;
                private JLabel label;
                private long newYearMillis;
                private String message;

                public HappyNewYear(JFrame frame, JLabel label)
                {
                        // store argument GUI elements
                        this.frame = frame;
                        this.label = label;
                        // compute beginning of next year
                        Calendar cal = new GregorianCalendar();
                        int nextYear = cal.get(Calendar.YEAR) + 1;
                        cal.set(Calendar.YEAR, nextYear);
                        cal.set(Calendar.MONTH, Calendar.JANUARY);
                        cal.set(Calendar.DAY_OF_MONTH, 1);
                        cal.set(Calendar.HOUR_OF_DAY, 0);
                        cal.set(Calendar.MINUTE, 0);
                        cal.set(Calendar.SECOND, 0);
                        newYearMillis = cal.getTime().getTime();
                        // prepare a message
                        message = "Happy " + nextYear + "!";
                }

                public static int determineFontSize(JFrame frame,
                        int componentWidth, String fontName, int fontStyle,
                        String text)
                {
                        int fontSize = componentWidth * 2 / text.length();
                        Font font = new Font(fontName, fontStyle, fontSize);
                        FontMetrics fontMetrics = frame.getGraphics().
                                getFontMetrics(font);
                        int stringWidth = fontMetrics.stringWidth(text);
                        return (int)(fontSize * 0.95 *
                                componentWidth / stringWidth);
                }

                public static void main(String[] args)
                {
                        JFrame frame = new JFrame();
                        frame.addKeyListener(new KeyListener()
                        {
                                public void keyPressed(KeyEvent event) {}
                                public void keyReleased(KeyEvent event) {
                                        if (event.getKeyChar() == KeyEvent.VK_ESCAPE)
                                        {
                                                System.exit(0);
                                        }
                                }
                                public void keyTyped(KeyEvent event) {}
                        }
                        );
                        frame.setUndecorated(true);
                        JLabel label = new JLabel(".");
                        label.setBackground(Color.BLACK);
                        label.setForeground(Color.WHITE);
                        label.setOpaque(true);
                        label.setHorizontalAlignment(SwingConstants.CENTER);
                        frame.getContentPane().add(label);
                        GraphicsEnvironment.getLocalGraphicsEnvironment().
                                getDefaultScreenDevice().setFullScreenWindow(frame);
                        final int fontStyle = Font.BOLD;
                        final String fontName = "SansSerif";
                        int fontSizeNumber = determineFontSize(frame,
                                Toolkit.getDefaultToolkit().getScreenSize().width,
                                fontName, fontStyle, formatter.format(88888888));
                        int fontSizeText = determineFontSize(frame,
                                Toolkit.getDefaultToolkit().getScreenSize().width,
                                fontName, fontStyle, "Happy 8888!");
                        label.setFont(new Font(fontName, fontStyle,
                                Math.min(fontSizeNumber, fontSizeText)));
                        new HappyNewYear(frame, label).run();
    }

    public void run()
    {
                boolean newYear = false;
                do
                {
                        long time = System.currentTimeMillis();
                        long remaining = (newYearMillis - time) / 1000L;
                        String output;
                        if (remaining < 1)
                                {
                                // new year!
                                newYear = true;
                                output = message;
                                }
                        else
                                {
                                // make a String from the number of seconds
                                output = formatter.format(remaining);
                                }
                                label.setText(output);
                                try
                                {
                                        Thread.sleep(1000);

    }
}
```

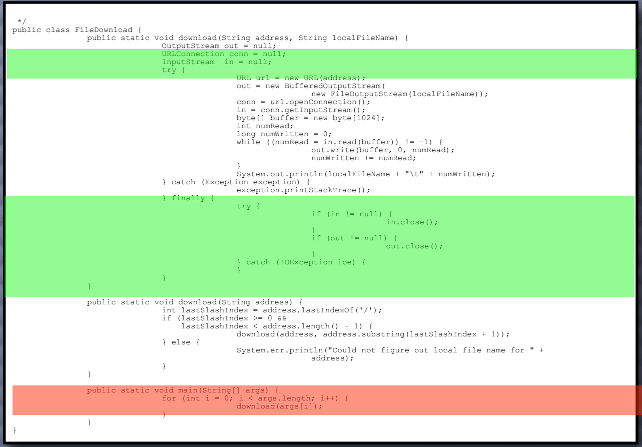
Crosscutting Concerns

🟩 Synchronization

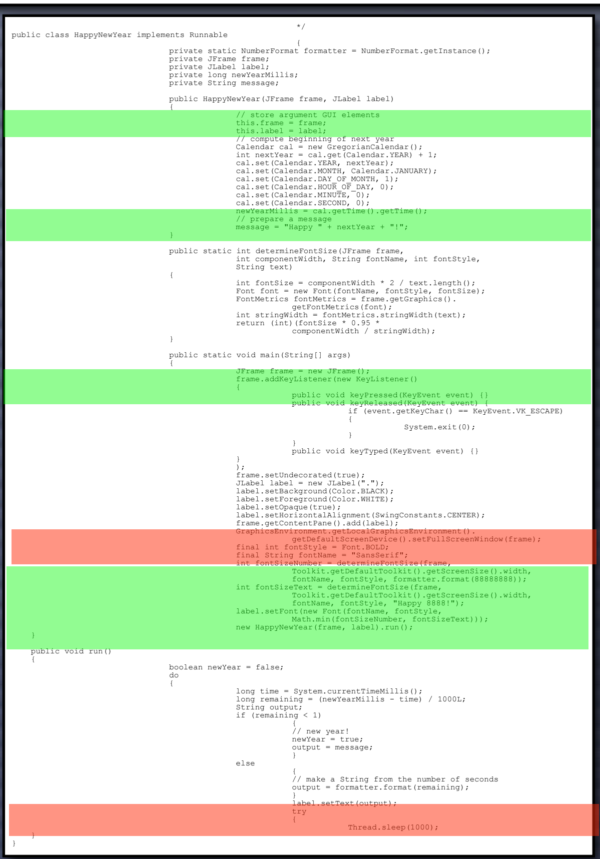
🟥 UI dependency



```java
import java.io.*;
import java.util.zip.*;

/**
 * Command line program to copy a file to another directory.
 * @author Marco Schmidt
 */
public class CopyFile {
        // constant values for the override option
        public static final int OVERWRITE_ALWAYS = 1;
        public static final int OVERWRITE_NEVER = 2;
        public static final int OVERWRITE_ASK = 3;

        // program options initialized to default values
        private static int bufferSize = 4 * 1024;
        private static boolean clock = true;
        private static boolean copyOriginalTimestamp = true;
        private static boolean verify = true;
        private static int override = OVERWRITE_ASK;

        public static Long copyFile(File srcFile, File destFile)
                throws IOException {
                InputStream in = new FileInputStream(srcFile);
                OutputStream out = new FileOutputStream(destFile);
                long millis = System.currentTimeMillis();
                CRC32 checksum = null;
                if (verify) {
                        checksum = new CRC32();
                        checksum.reset();
                }
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        if (verify) {
                                checksum.update(buffer, 0, bytesRead);
                        }
                        out.write(buffer, 0, bytesRead);
                }
                out.close();
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                if (verify) {
                        return new Long(checksum.getValue());
                } else {
                        return null;
                }
        }

        public static Long createChecksum(File file) throws IOException {
                long millis = System.currentTimeMillis();
                InputStream in = new FileInputStream(file);
                CRC32 checksum = new CRC32();
                checksum.reset();
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        checksum.update(buffer, 0, bytesRead);
                }
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                return new Long(checksum.getValue());
        }

        /**
         * Determine if data is to be copied to given file.
         * Take into consideration override option and
         * ask user in case file exists and override option is ask.
         * @param file File object for potential destination file
         * @return true if data is to be copied to file, false if not
         */
        public static boolean doCopy(File file) {
                boolean exists = file.exists();
                if (override == OVERWRITE_ALWAYS || !exists) {
                        return true;
                } else
                if (override == OVERWRITE_NEVER) {
                        return false;
                } else
                if (override == OVERWRITE_ASK) {
                        return readYesNoFromStandardInput("File exists. " +
                                "Overwrite (y/n)?");
                } else {
                        throw new InternalError("Program error. Invalid " +
                                "value for override: " + override);
                }
        }

        public static void main(String[] args) throws IOException {
                // make sure there are exactly two arguments
                if (args.length != 2) {
                        System.err.println("Usage: CopyFile SRC-FILE-NAME DEST-DIR-NAME");
                        System.exit(1);
                }
                // make sure the source file is indeed a readable file
                File srcFile = new File(args[0]);
                if (!srcFile.isFile() || !srcFile.canRead()) {
                        System.err.println("Not a readable file: " + srcFile.getName());
                        System.exit(1);
                }
                // make sure the second argument is a directory
                File destDir = new File(args[1]);
                if (!destDir.isDirectory()) {
                        System.err.println("Not a directory: " + destDir.getName());
                        System.exit(1);
                }
                // create File object for destination file
                File destFile = new File(destDir, srcFile.getName());

                // check if copying is desired given overwrite option
                if (!doCopy(destFile)) {
                        return;
                }

                // copy file, optionally creating a checksum
                Long checksumSrc = copyFile(srcFile, destFile);

                // copy timestamp of last modification
                if (copyOriginalTimestamp) {
                        if (!destFile.setLastModified(srcFile.lastModified())) {
                                System.err.println("Error: Could not set " +
                                        "timestamp of copied file.");
                        }
                }

                // optionally verify file
                if (verify) {
                        System.out.print("Verifying destination file...");
                        Long checksumDest = createChecksum(destFile);
                        if (checksumSrc.equals(checksumDest)) {
                                System.out.println(" OK, files are equal.");
                        } else {
                                System.out.println(" Error: Checksums differ.");
                        }
                }
        }

        /**
         * Print a message to standard output and read lines from
         * standard input until yes or no (y or n) is entered.
         * @param message informative text to be answered by user
         * @return user answer, true for yes, false for no.
         */
        public static boolean readYesNoFromStandardInput(String message) {
                System.out.println(message);
                String line;
                BufferedReader in = new BufferedReader(new InputStreamReader(
                        System.in));
                Boolean answer = null;
                try
                {
                        while ((line = in.readLine()) != null) {
                                line = line.toLowerCase();
                                if ("y".equals(line) || "yes".equals(line)) {
                                        answer = Boolean.TRUE;
                                        break;
                                }
                                else
                                if ("n".equals(line) || "no".equals(line)) {
                                        answer = Boolean.FALSE;
                                        break;
                                }
                                else
                                {
                                        System.out.println("Could not understand answer (\"" +
                                                line + "\"). Please use y for yes or n for no.");
                                }
                        }
                        if (answer == null) {
                                throw new IOException("Unexpected end of input from stdin.");
                        }
                        in.close();
                        return answer.booleanValue();
                }
                catch (IOException ioe)
                {
                        throw new InternalError(
                                "Cannot read from stdin or write to stdout.");
                }
        }
}
```

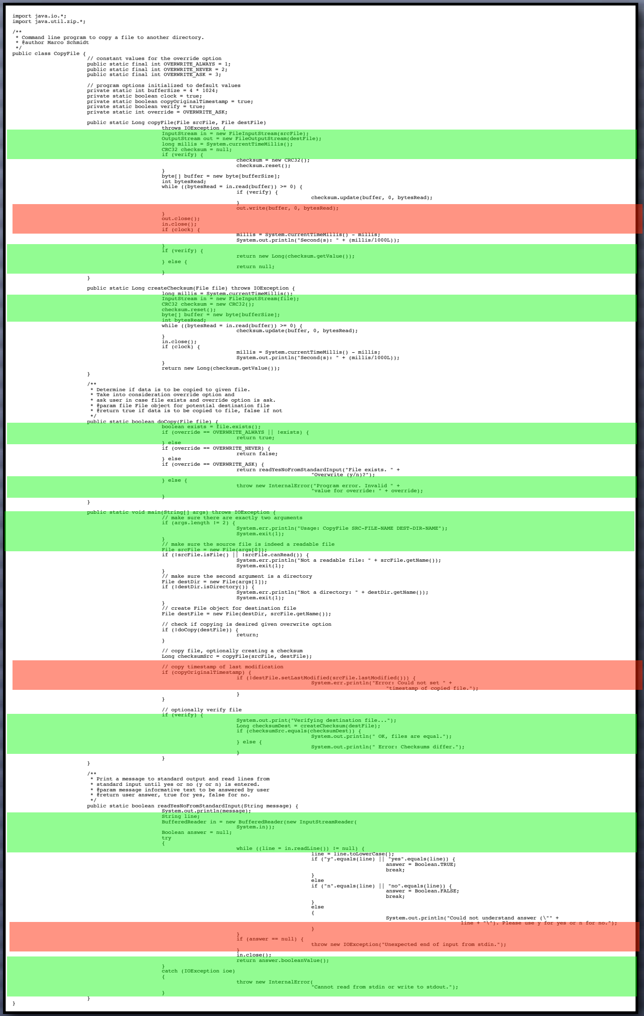


```java
 */
public class FileDownload {
        public static void download(String address, String localFileName) {
                OutputStream out = null;
                URLConnection conn = null;
                InputStream  in = null;
                try {
                        URL url = new URL(address);
                        out = new BufferedOutputStream(
                                new FileOutputStream(localFileName));
                        conn = url.openConnection();
                        in = conn.getInputStream();
                        byte[] buffer = new byte[1024];
                        int numRead;
                        long numWritten = 0;
                        while ((numRead = in.read(buffer)) != -1) {
                                out.write(buffer, 0, numRead);
                                numWritten += numRead;
                        }
                        System.out.println(localFileName + "\t" + numWritten);
                } catch (Exception exception) {
                        exception.printStackTrace();
                } finally {
                        try {
                                if (in != null) {
                                        in.close();
                                }
                                if (out != null) {
                                        out.close();
                                }
                        } catch (IOException ioe) {
                        }
                }
        }

        public static void download(String address) {
                int lastSlashIndex = address.lastIndexOf('/');
                if (lastSlashIndex >= 0 &&
                    lastSlashIndex < address.length() - 1) {
                        download(address, address.substring(lastSlashIndex + 1));
                } else {
                        System.err.println("Could not figure out local file name for " +
                                address);
                }
        }

        public static void main(String[] args) {
                for (int i = 0; i < args.length; i++) {
                        download(args[i]);
                }
        }
}
```

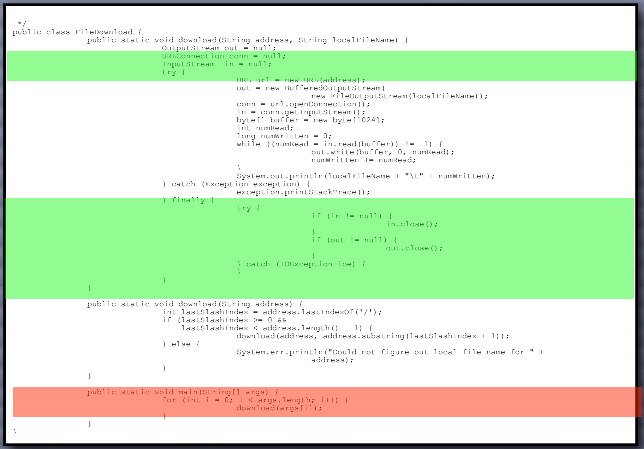


```java
                                        */
public class HappyNewYear implements Runnable
                                        {
                private static NumberFormat formatter = NumberFormat.getInstance();
                private JFrame frame;
                private JLabel label;
                private long newYearMillis;
                private String message;

                public HappyNewYear(JFrame frame, JLabel label)
                {
                        // store argument GUI elements
                        this.frame = frame;
                        this.label = label;
                        // compute beginning of next year
                        Calendar cal = new GregorianCalendar();
                        int nextYear = cal.get(Calendar.YEAR) + 1;
                        cal.set(Calendar.YEAR, nextYear);
                        cal.set(Calendar.MONTH, Calendar.JANUARY);
                        cal.set(Calendar.DAY_OF_MONTH, 1);
                        cal.set(Calendar.HOUR_OF_DAY, 0);
                        cal.set(Calendar.MINUTE, 0);
                        cal.set(Calendar.SECOND, 0);
                        newYearMillis = cal.getTime().getTime();
                        // prepare a message
                        message = "Happy " + nextYear + "!";
                }

                public static int determineFontSize(JFrame frame,
                        int componentWidth, String fontName, int fontStyle,
                        String text)
                {
                        int fontSize = componentWidth * 2 / text.length();
                        Font font = new Font(fontName, fontStyle, fontSize);
                        FontMetrics fontMetrics = frame.getGraphics().
                                getFontMetrics(font);
                        int stringWidth = fontMetrics.stringWidth(text);
                        return (int)(fontSize * 0.95 *
                                componentWidth / stringWidth);
                }

                public static void main(String[] args)
                {
                        JFrame frame = new JFrame();
                        frame.addKeyListener(new KeyListener()
                        {
                                public void keyPressed(KeyEvent event) {}
                                public void keyReleased(KeyEvent event) {
                                        if (event.getKeyChar() == KeyEvent.VK_ESCAPE)
                                        {
                                                System.exit(0);
                                        }
                                }
                                public void keyTyped(KeyEvent event) {}
                        }
                        );
                        frame.setUndecorated(true);
                        JLabel label = new JLabel(".");
                        label.setBackground(Color.BLACK);
                        label.setForeground(Color.WHITE);
                        label.setOpaque(true);
                        label.setHorizontalAlignment(SwingConstants.CENTER);
                        frame.getContentPane().add(label);
                        GraphicsEnvironment.getLocalGraphicsEnvironment().
                                getDefaultScreenDevice().setFullScreenWindow(frame);
                        final int fontStyle = Font.BOLD;
                        final String fontName = "SansSerif";
                        int fontSizeNumber = determineFontSize(frame,
                                Toolkit.getDefaultToolkit().getScreenSize().width,
                                fontName, fontStyle, formatter.format(88888888));
                        int fontSizeText = determineFontSize(frame,
                                Toolkit.getDefaultToolkit().getScreenSize().width,
                                fontName, fontStyle, "Happy 8888!");
                        label.setFont(new Font(fontName, fontStyle,
                                Math.min(fontSizeNumber, fontSizeText)));
                        new HappyNewYear(frame, label).run();
    }

    public void run()
    {
                boolean newYear = false;
                do
                {
                        long time = System.currentTimeMillis();
                        long remaining = (newYearMillis - time) / 1000L;
                        String output;
                        if (remaining < 1)
                                {
                                // new year!
                                newYear = true;
                                output = message;
                                }
                        else
                                {
                                // make a String from the number of seconds
                                output = formatter.format(remaining);
                                }
                                label.setText(output);
                                try
                                {
                                        Thread.sleep(1000);
    }
}
```

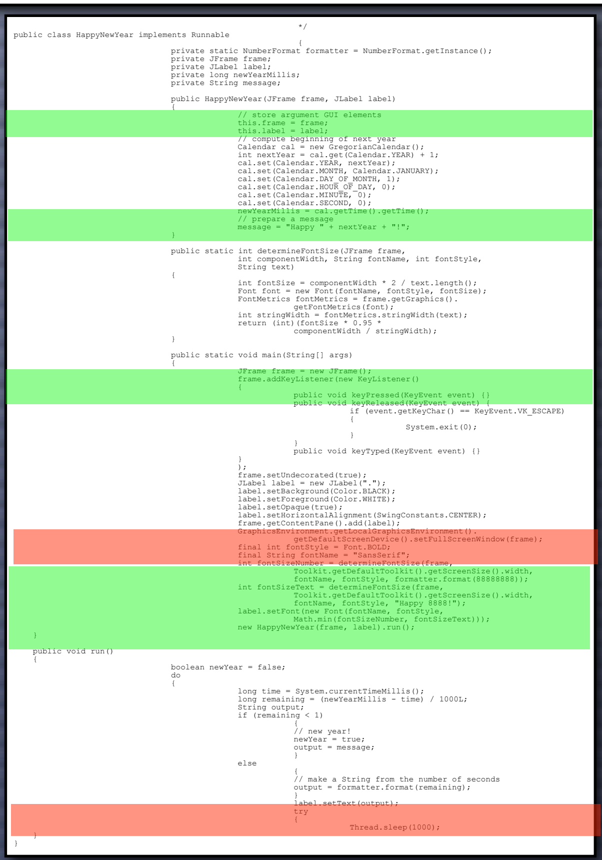
Crosscutting Concerns



```java
import java.io.*;
import java.util.zip.*;

/**
 * Command line program to copy a file to another directory.
 * @author Marco Schmidt
 */
public class CopyFile {
        // constant values for the override option
        public static final int OVERWRITE_ALWAYS = 1;
        public static final int OVERWRITE_NEVER = 2;
        public static final int OVERWRITE_ASK = 3;

        // program options initialized to default values
        private static int bufferSize = 4 * 1024;
        private static boolean clock = true;
        private static boolean copyOriginalTimestamp = true;
        private static boolean verify = true;
        private static int override = OVERWRITE_ASK;

        public static Long copyFile(File srcFile, File destFile)
                throws IOException {
                InputStream in = new FileInputStream(srcFile);
                OutputStream out = new FileOutputStream(destFile);
                long millis = System.currentTimeMillis();
                CRC32 checksum = null;
                if (verify) {
                        checksum = new CRC32();
                        checksum.reset();
                }
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        if (verify) {
                                checksum.update(buffer, 0, bytesRead);
                        }
                        out.write(buffer, 0, bytesRead);
                }
                out.close();
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                if (verify) {
                        return new Long(checksum.getValue());
                } else {
                        return null;
                }
        }

        public static Long createChecksum(File file) throws IOException {
                long millis = System.currentTimeMillis();
                InputStream in = new FileInputStream(file);
                CRC32 checksum = new CRC32();
                checksum.reset();
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        checksum.update(buffer, 0, bytesRead);
                }
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                return new Long(checksum.getValue());
        }

        /**
         * Determine if data is to be copied to given file.
         * Take into consideration override option and
         * ask user in case file exists and override option is ask.
         * @param file File object for potential destination file
         * @return true if data is to be copied to file, false if not
         */
        public static boolean doCopy(File file) {
                boolean exists = file.exists();
                if (override == OVERWRITE_ALWAYS || !exists) {
                        return true;
                } else
                if (override == OVERWRITE_NEVER) {
                        return false;
                } else
                if (override == OVERWRITE_ASK) {
                        return readYesNoFromStandardInput("File exists. " +
                                "Overwrite (y/n)?");
                } else {
                        throw new InternalError("Program error. Invalid " +
                                "value for override: " + override);
                }
        }

        public static void main(String[] args) throws IOException {
                // make sure there are exactly two arguments
                if (args.length != 2) {
                        System.err.println("Usage: CopyFile SRC-FILE-NAME DEST-DIR-NAME");
                        System.exit(1);
                }
                // make sure the source file is indeed a readable file
                File srcFile = new File(args[0]);
                if (!srcFile.isFile() || !srcFile.canRead()) {
                        System.err.println("Not a readable file: " + srcFile.getName());
                        System.exit(1);
                }
                // make sure the second argument is a directory
                File destDir = new File(args[1]);
                if (!destDir.isDirectory()) {
                        System.err.println("Not a directory: " + destDir.getName());
                        System.exit(1);
                }
                // create File object for destination file
                File destFile = new File(destDir, srcFile.getName());

                // check if copying is desired given overwrite option
                if (!doCopy(destFile)) {
                        return;
                }

                // copy file, optionally creating a checksum
                Long checksumSrc = copyFile(srcFile, destFile);

                // copy timestamp of last modification
                if (copyOriginalTimestamp) {
                        if (!destFile.setLastModified(srcFile.lastModified())) {
                                System.err.println("Error: Could not set " +
                                        "timestamp of copied file.");
                        }
                }

                // optionally verify file
                if (verify) {
                        System.out.print("Verifying destination file...");
                        Long checksumDest = createChecksum(destFile);
                        if (checksumSrc.equals(checksumDest)) {
                                System.out.println(" OK, files are equal.");
                        } else {
                                System.out.println(" Error: Checksums differ.");
                        }
                }
        }

        /**
         * Print a message to standard output and read lines from
         * standard input until yes or no (y or n) is entered.
         * @param message informative text to be answered by user
         * @return user answer, true for yes, false for no.
         */
        public static boolean readYesNoFromStandardInput(String message) {
                System.out.println(message);
                String line;
                BufferedReader in = new BufferedReader(new InputStreamReader(
                        System.in));
                Boolean answer = null;
                try
                {
                        while ((line = in.readLine()) != null) {
                                line = line.toLowerCase();
                                if ("y".equals(line) || "yes".equals(line)) {
                                        answer = Boolean.TRUE;
                                        break;
                                }
                                else
                                if ("n".equals(line) || "no".equals(line)) {
                                        answer = Boolean.FALSE;
                                        break;
                                }
                                else
                                {
                                        System.out.println("Could not understand answer (\"" +
                                                line + "\"). Please use y for yes or n for no.");
                                }
                        }
                        if (answer == null) {
                                throw new IOException("Unexpected end of input from stdin.");
                        }
                        in.close();
                        return answer.booleanValue();
                }
                catch (IOException ioe)
                {
                        throw new InternalError(
                                "Cannot read from stdin or write to stdout.");
                }
        }
}
```



```java
 */
public class FileDownload {
        public static void download(String address, String localFileName) {
                OutputStream out = null;
                URLConnection conn = null;
                InputStream  in = null;
                try {
                        URL url = new URL(address);
                        out = new BufferedOutputStream(
                                new FileOutputStream(localFileName));
                        conn = url.openConnection();
                        in = conn.getInputStream();
                        byte[] buffer = new byte[1024];
                        int numRead;
                        long numWritten = 0;
                        while ((numRead = in.read(buffer)) != -1) {
                                out.write(buffer, 0, numRead);
                                numWritten += numRead;
                        }
                        System.out.println(localFileName + "\t" + numWritten);
                } catch (Exception exception) {
                        exception.printStackTrace();
                } finally {
                        try {
                                if (in != null) {
                                        in.close();
                                }
                                if (out != null) {
                                        out.close();
                                }
                        } catch (IOException ioe) {
                        }
                }
        }

        public static void download(String address) {
                int lastSlashIndex = address.lastIndexOf('/');
                if (lastSlashIndex >= 0 &&
                    lastSlashIndex < address.length() - 1) {
                        download(address, address.substring(lastSlashIndex + 1));
                } else {
                        System.err.println("Could not figure out local file name for " +
                                address);
                }
        }

        public static void main(String[] args) {
                for (int i = 0; i < args.length; i++) {
                        download(args[i]);
                }
        }
}
```



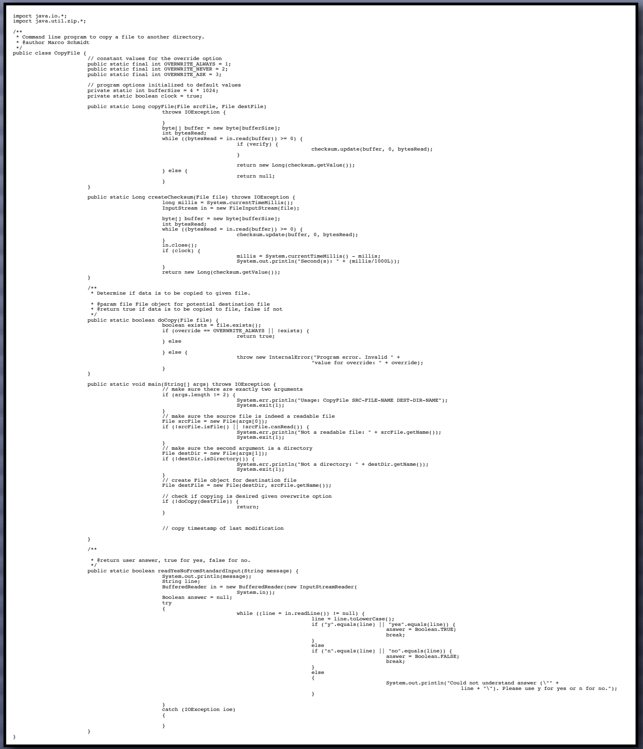
```java
                                        */
public class HappyNewYear implements Runnable
                                        {
                private static NumberFormat formatter = NumberFormat.getInstance();
                private JFrame frame;
                private JLabel label;
                private long newYearMillis;
                private String message;

                public HappyNewYear(JFrame frame, JLabel label)
                {
                        // store argument GUI elements
                        this.frame = frame;
                        this.label = label;
                        // compute beginning of next year
                        Calendar cal = new GregorianCalendar();
                        int nextYear = cal.get(Calendar.YEAR) + 1;
                        cal.set(Calendar.YEAR, nextYear);
                        cal.set(Calendar.MONTH, Calendar.JANUARY);
                        cal.set(Calendar.DAY_OF_MONTH, 1);
                        cal.set(Calendar.HOUR_OF_DAY, 0);
                        cal.set(Calendar.MINUTE, 0);
                        cal.set(Calendar.SECOND, 0);
                        newYearMillis = cal.getTime().getTime();
                        // prepare a message
                        message = "Happy " + nextYear + "!";
                }

                public static int determineFontSize(JFrame frame,
                        int componentWidth, String fontName, int fontStyle,
                        String text)
                {
                        int fontSize = componentWidth * 2 / text.length();
                        Font font = new Font(fontName, fontStyle, fontSize);
                        FontMetrics fontMetrics = frame.getGraphics().
                                getFontMetrics(font);
                        int stringWidth = fontMetrics.stringWidth(text);
                        return (int)(fontSize * 0.95 *
                                componentWidth / stringWidth);
                }

                public static void main(String[] args)
                {
                        JFrame frame = new JFrame();
                        frame.addKeyListener(new KeyListener()
                        {
                                public void keyPressed(KeyEvent event) {}
                                public void keyReleased(KeyEvent event) {
                                        if (event.getKeyChar() == KeyEvent.VK_ESCAPE)
                                        {
                                                System.exit(0);
                                        }
                                }
                                public void keyTyped(KeyEvent event) {}
                        }
                        );
                        frame.setUndecorated(true);
                        JLabel label = new JLabel(".");
                        label.setBackground(Color.BLACK);
                        label.setForeground(Color.WHITE);
                        label.setOpaque(true);
                        label.setHorizontalAlignment(SwingConstants.CENTER);
                        frame.getContentPane().add(label);
                        GraphicsEnvironment.getLocalGraphicsEnvironment().
                                getDefaultScreenDevice().setFullScreenWindow(frame);
                        final int fontStyle = Font.BOLD;
                        final String fontName = "SansSerif";
                        int fontSizeNumber = determineFontSize(frame,
                                Toolkit.getDefaultToolkit().getScreenSize().width,
                                fontName, fontStyle, formatter.format(88888888));
                        int fontSizeText = determineFontSize(frame,
                                Toolkit.getDefaultToolkit().getScreenSize().width,
                                fontName, fontStyle, "Happy 8888!");
                        label.setFont(new Font(fontName, fontStyle,
                                Math.min(fontSizeNumber, fontSizeText)));
                        new HappyNewYear(frame, label).run();
    }

    public void run()
    {
                boolean newYear = false;
                do
                {
                        long time = System.currentTimeMillis();
                        long remaining = (newYearMillis - time) / 1000L;
                        String output;
                        if (remaining < 1)
                                {
                                // new year!
                                newYear = true;
                                output = message;
                                }
                        else
                                {
                                // make a String from the number of seconds
                                output = formatter.format(remaining);
                                }
                                label.setText(output);
                                try
                                {
                                        Thread.sleep(1000);
    }
}
```

Synchronization

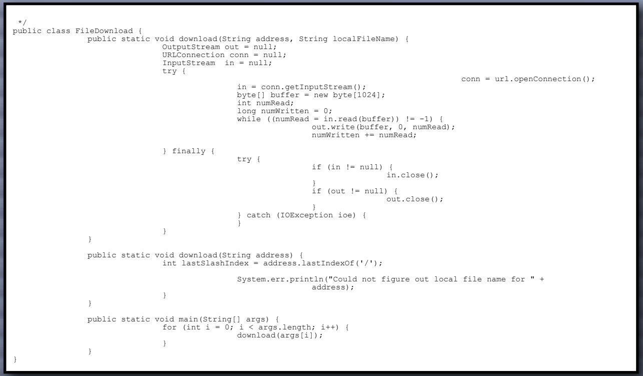
UI dependency

Scattering & Tangling

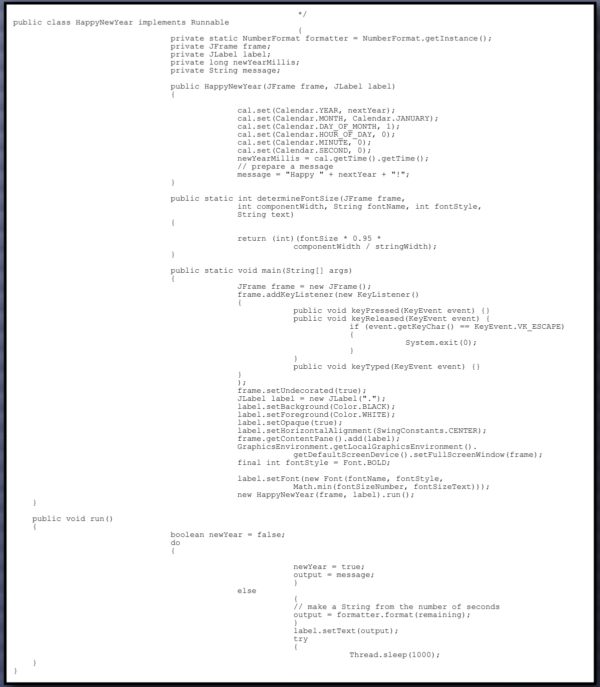
AOP



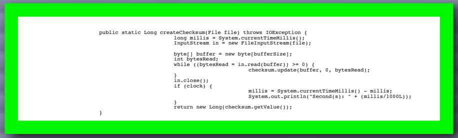
```
import java.io.*;
import java.util.zip.*;

/**
 * Command line program to copy a file to another directory.
 * @author Marco Schmidt
 */
public class CopyFile {
            // constant values for the override option
            public static final int OVERWRITE_ALWAYS = 1;
            public static final int OVERWRITE_NEVER = 2;
            public static final int OVERWRITE_ASK = 3;

            // program options initialized to default values
            private static int bufferSize = 4 * 1024;
            private static boolean clock = true;

            public static Long copyFile(File srcFile, File destFile)
                        throws IOException {

                        }
                        byte[] buffer = new byte[bufferSize];
                        int bytesRead;
                        while ((bytesRead = in.read(buffer)) >= 0) {
                                    if (verify) {
                                                checksum.update(buffer, 0, bytesRead);
                                    }

                                    return new Long(checksum.getValue());
                        } else {
                                    return null;
                        }
            }

            public static Long createChecksum(File file) throws IOException {
                        long millis = System.currentTimeMillis();
                        InputStream in = new FileInputStream(file);

                        byte[] buffer = new byte[bufferSize];
                        int bytesRead;
                        while ((bytesRead = in.read(buffer)) >= 0) {
                                    checksum.update(buffer, 0, bytesRead);
                        }
                        in.close();
                        if (clock) {
                                    millis = System.currentTimeMillis() - millis;
                                    System.out.println("Second(s): " + (millis/1000L));
                        }
                        return new Long(checksum.getValue());
            }

            /**
             * Determine if data is to be copied to given file.

             * @param file File object for potential destination file
             * @return true if data is to be copied to file, false if not
             */
            public static boolean doCopy(File file) {
                        boolean exists = file.exists();
                        if (override == OVERWRITE_ALWAYS || !exists) {
                                    return true;
                        } else

                        } else {
                                    throw new InternalError("Program error. Invalid " +
                                                "value for override: " + override);
                        }
            }

            public static void main(String[] args) throws IOException {
                        // make sure there are exactly two arguments
                        if (args.length != 2) {
                                    System.err.println("Usage: CopyFile SRC-FILE-NAME DEST-DIR-NAME");
                                    System.exit(1);
                        }
                        // make sure the source file is indeed a readable file
                        File srcFile = new File(args[0]);
                        if (!srcFile.isFile() || !srcFile.canRead()) {
                                    System.err.println("Not a readable file: " + srcFile.getName());
                                    System.exit(1);
                        }
                        // make sure the second argument is a directory
                        File destDir = new File(args[1]);
                        if (!destDir.isDirectory()) {
                                    System.err.println("Not a directory: " + destDir.getName());
                                    System.exit(1);
                        }
                        // create File object for destination file
                        File destFile = new File(destDir, srcFile.getName());

                        // check if copying is desired given overwrite option
                        if (!doCopy(destFile)) {
                                    return;
                        }


                        // copy timestamp of last modification

            }

            /**

             * @return user answer, true for yes, false for no.
             */
            public static boolean readYesNoFromStandardInput(String message) {
                        System.out.println(message);
                        String line;
                        BufferedReader in = new BufferedReader(new InputStreamReader(
                                    System.in));
                        Boolean answer = null;
                        try
                        {
                                    while ((line = in.readLine()) != null) {
                                                line = line.toLowerCase();
                                                if ("y".equals(line) || "yes".equals(line)) {
                                                            answer = Boolean.TRUE;
                                                            break;
                                                }
                                                else
                                                if ("n".equals(line) || "no".equals(line)) {
                                                            answer = Boolean.FALSE;
                                                            break;
                                                }
                                                else
                                                {
                                                            System.out.println("Could not understand answer (\"" +
                                                                        line + "\"). Please use y for yes or n for no.");
                                                }

                        }
                        catch (IOException ioe)
                        {

                        }
            }
}
```



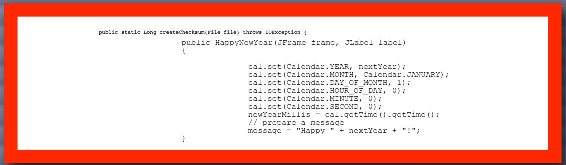
```
 */
public class FileDownload {
            public static void download(String address, String localFileName) {
                        OutputStream out = null;
                        URLConnection conn = null;
                        InputStream  in = null;
                        try {
                                                                                conn = url.openConnection();
                                    in = conn.getInputStream();
                                    byte[] buffer = new byte[1024];
                                    int numRead;
                                    long numWritten = 0;
                                    while ((numRead = in.read(buffer)) != -1) {
                                                out.write(buffer, 0, numRead);
                                                numWritten += numRead;

                        } finally {
                                    try {
                                                if (in != null) {
                                                            in.close();
                                                }
                                                if (out != null) {
                                                            out.close();
                                                }
                                    } catch (IOException ioe) {
                                    }
                        }
            }

            public static void download(String address) {
                        int lastSlashIndex = address.lastIndexOf('/');

                                    System.err.println("Could not figure out local file name for " +
                                                address);
                        }
            }

            public static void main(String[] args) {
                        for (int i = 0; i < args.length; i++) {
                                    download(args[i]);
                        }
            }
}
```



```
                                                                                 */
public class HappyNewYear implements Runnable
                                                                                  {
                    private static NumberFormat formatter = NumberFormat.getInstance();
                    private JFrame frame;
                    private JLabel label;
                    private long newYearMillis;
                    private String message;

                    public HappyNewYear(JFrame frame, JLabel label)
                    {

                              cal.set(Calendar.YEAR, nextYear);
                              cal.set(Calendar.MONTH, Calendar.JANUARY);
                              cal.set(Calendar.DAY_OF_MONTH, 1);
                              cal.set(Calendar.HOUR_OF_DAY, 0);
                              cal.set(Calendar.MINUTE, 0);
                              cal.set(Calendar.SECOND, 0);
                              newYearMillis = cal.getTime().getTime();
                              // prepare a message
                              message = "Happy " + nextYear + "!";
                    }

                    public static int determineFontSize(JFrame frame,
                              int componentWidth, String fontName, int fontStyle,
                              String text)
                    {

                              return (int)(fontSize * 0.95 *
                                        componentWidth / stringWidth);
                    }

                    public static void main(String[] args)
                    {
                              JFrame frame = new JFrame();
                              frame.addKeyListener(new KeyListener()
                              {
                                        public void keyPressed(KeyEvent event) {}
                                        public void keyReleased(KeyEvent event) {
                                                  if (event.getKeyChar() == KeyEvent.VK_ESCAPE)
                                                  {
                                                            System.exit(0);
                                                  }
                                        }
                                        public void keyTyped(KeyEvent event) {}
                              }
                              );
                              frame.setUndecorated(true);
                              JLabel label = new JLabel(".");
                              label.setBackground(Color.BLACK);
                              label.setForeground(Color.WHITE);
                              label.setOpaque(true);
                              label.setHorizontalAlignment(SwingConstants.CENTER);
                              frame.getContentPane().add(label);
                              GraphicsEnvironment.getLocalGraphicsEnvironment().
                                        getDefaultScreenDevice().setFullScreenWindow(frame);
                              final int fontStyle = Font.BOLD;

                              label.setFont(new Font(fontName, fontStyle,
                                        Math.min(fontSizeNumber, fontSizeText)));
                              new HappyNewYear(frame, label).run();
    }

    public void run()
    {
                    boolean newYear = false;
                    do
                    {

                                        newYear = true;
                                        output = message;
                                        }
                              else
                                        {
                                        // make a String from the number of seconds
                                        output = formatter.format(remaining);
                                        }
                                        label.setText(output);
                                        try
                                        {
                                                  Thread.sleep(1000);
    }
}
```



```
            public static Long createChecksum(File file) throws IOException {
                        long millis = System.currentTimeMillis();
                        InputStream in = new FileInputStream(file);

                        byte[] buffer = new byte[bufferSize];
                        int bytesRead;
                        while ((bytesRead = in.read(buffer)) >= 0) {
                                    checksum.update(buffer, 0, bytesRead);
                        }
                        in.close();
                        if (clock) {
                                    millis = System.currentTimeMillis() - millis;
                                    System.out.println("Second(s): " + (millis/1000L));
                        }
                        return new Long(checksum.getValue());
            }
```

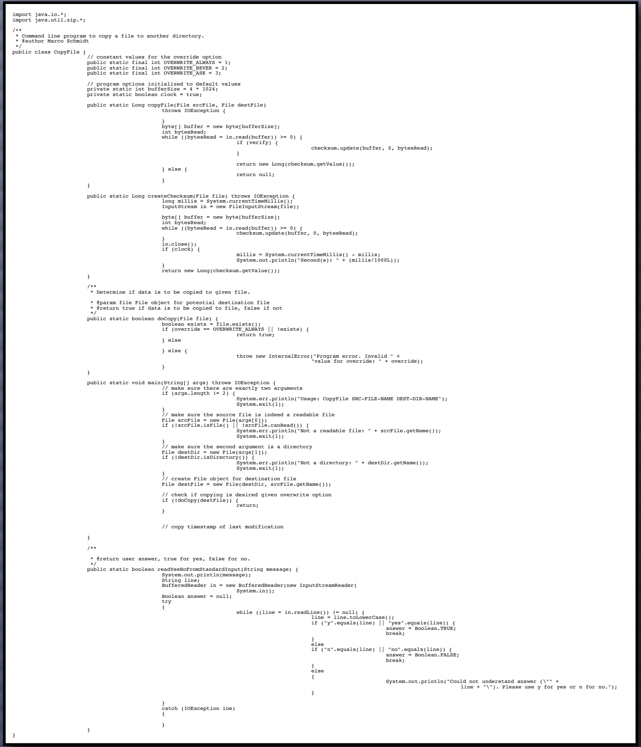


```
            public static Long createChecksum(File file) throws IOException {
                    public HappyNewYear(JFrame frame, JLabel label)
                    {

                              cal.set(Calendar.YEAR, nextYear);
                              cal.set(Calendar.MONTH, Calendar.JANUARY);
                              cal.set(Calendar.DAY_OF_MONTH, 1);
                              cal.set(Calendar.HOUR_OF_DAY, 0);
                              cal.set(Calendar.MINUTE, 0);
                              cal.set(Calendar.SECOND, 0);
                              newYearMillis = cal.getTime().getTime();
                              // prepare a message
                              message = "Happy " + nextYear + "!";
                    }
```

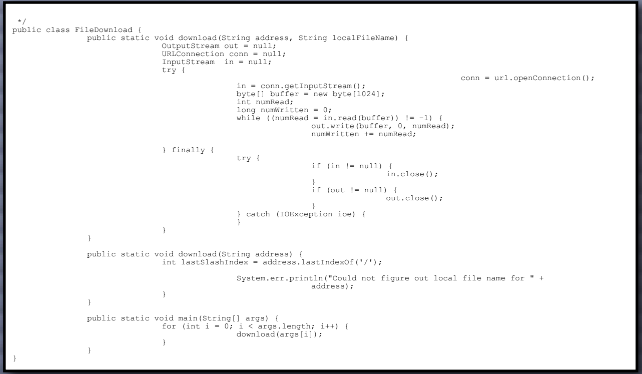
Modularized

Pointcuts

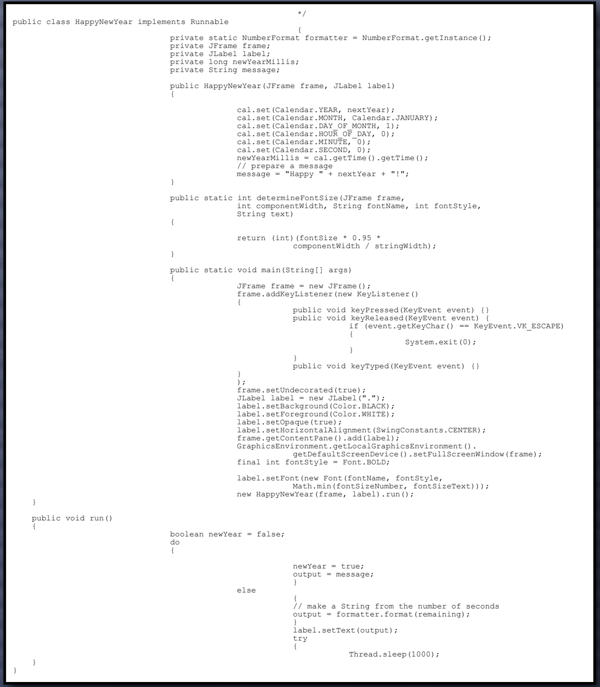
Often point to details in source code (e.g. names)



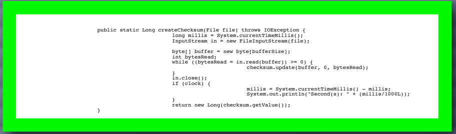
```
import java.io.*;
import java.util.zip.*;

/**
 * Command line program to copy a file to another directory.
 * @author Marco Schmidt
 */
public class CopyFile {
        // constant values for the override option
        public static final int OVERWRITE_ALWAYS = 1;
        public static final int OVERWRITE_NEVER = 2;
        public static final int OVERWRITE_ASK = 3;

        // program options initialized to default values
        private static int bufferSize = 4 * 1024;
        private static boolean clock = true;

        public static Long copyFile(File srcFile, File destFile)
                throws IOException {

                }
                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        if (verify) {
                                checksum.update(buffer, 0, bytesRead);
                        }

                        return new Long(checksum.getValue());
                } else {
                        return null;
                }
        }

        public static Long createChecksum(File file) throws IOException {
                long millis = System.currentTimeMillis();
                InputStream in = new FileInputStream(file);

                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                        checksum.update(buffer, 0, bytesRead);
                }
                in.close();
                if (clock) {
                        millis = System.currentTimeMillis() - millis;
                        System.out.println("Second(s): " + (millis/1000L));
                }
                return new Long(checksum.getValue());
        }

        /**
         * Determine if data is to be copied to given file.

         * @param file File object for potential destination file
         * @return true if data is to be copied to file, false if not
         */
        public static boolean doCopy(File file) {
                boolean exists = file.exists();
                if (override == OVERWRITE_ALWAYS || !exists) {
                        return true;
                } else

                } else {
                        throw new InternalError("Program error. Invalid " +
                                "value for override: " + override);
                }
        }

        public static void main(String[] args) throws IOException {
                // make sure there are exactly two arguments
                if (args.length != 2) {
                        System.err.println("Usage: CopyFile SRC-FILE-NAME DEST-DIR-NAME");
                        System.exit(1);
                }
                // make sure the source file is indeed a readable file
                File srcFile = new File(args[0]);
                if (!srcFile.isFile() || !srcFile.canRead()) {
                        System.err.println("Not a readable file: " + srcFile.getName());
                        System.exit(1);
                }
                // make sure the second argument is a directory
                File destDir = new File(args[1]);
                if (!destDir.isDirectory()) {
                        System.err.println("Not a directory: " + destDir.getName());
                        System.exit(1);
                }
                // create File object for destination file
                File destFile = new File(destDir, srcFile.getName());

                // check if copying is desired given overwrite option
                if (!doCopy(destFile)) {
                        return;
                }


                // copy timestamp of last modification

        }

        /**

         * @return user answer, true for yes, false for no.
         */
        public static boolean readYesNoFromStandardInput(String message) {
                System.out.println(message);
                String line;
                BufferedReader in = new BufferedReader(new InputStreamReader(
                        System.in));
                Boolean answer = null;
                try
                {
                        while ((line = in.readLine()) != null) {
                                line = line.toLowerCase();
                                if ("y".equals(line) || "yes".equals(line)) {
                                        answer = Boolean.TRUE;
                                        break;
                                }
                                else
                                if ("n".equals(line) || "no".equals(line)) {
                                        answer = Boolean.FALSE;
                                        break;
                                }
                                else
                                {
                                        System.out.println("Could not understand answer (\"" +
                                                line + "\"). Please use y for yes or n for no.");
                                }
                }
                catch (IOException ioe)
                {

                }
        }
}
```



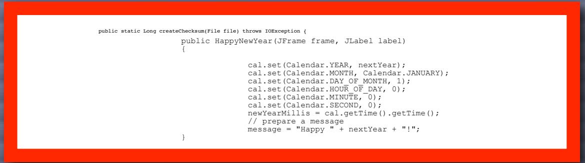
```
 */
public class FileDownload {
        public static void download(String address, String localFileName) {
                OutputStream out = null;
                URLConnection conn = null;
                InputStream  in = null;
                try {
                                                conn = url.openConnection();
                        in = conn.getInputStream();
                        byte[] buffer = new byte[1024];
                        int numRead;
                        long numWritten = 0;
                        while ((numRead = in.read(buffer)) != -1) {
                                out.write(buffer, 0, numRead);
                                numWritten += numRead;

                } finally {
                        try {
                                if (in != null) {
                                        in.close();
                                }
                                if (out != null) {
                                        out.close();
                                }
                        } catch (IOException ioe) {
                        }
                }
        }

        public static void download(String address) {
                int lastSlashIndex = address.lastIndexOf('/');

                        System.err.println("Could not figure out local file name for " +
                                address);
                }
        }

        public static void main(String[] args) {
                for (int i = 0; i < args.length; i++) {
                        download(args[i]);
                }
        }
}
```



```
                                */
public class HappyNewYear implements Runnable
                                {
                private static NumberFormat formatter = NumberFormat.getInstance();
                private JFrame frame;
                private JLabel label;
                private long newYearMillis;
                private String message;

                public HappyNewYear(JFrame frame, JLabel label)
                {

                        cal.set(Calendar.YEAR, nextYear);
                        cal.set(Calendar.MONTH, Calendar.JANUARY);
                        cal.set(Calendar.DAY_OF_MONTH, 1);
                        cal.set(Calendar.HOUR_OF_DAY, 0);
                        cal.set(Calendar.MINUTE, 0);
                        cal.set(Calendar.SECOND, 0);
                        newYearMillis = cal.getTime().getTime();
                        // prepare a message
                        message = "Happy " + nextYear + "!";
                }

                public static int determineFontSize(JFrame frame,
                        int componentWidth, String fontName, int fontStyle,
                        String text)
                {

                        return (int)(fontSize * 0.95 *
                                componentWidth / stringWidth);
                }

                public static void main(String[] args)
                {
                        JFrame frame = new JFrame();
                        frame.addKeyListener(new KeyListener()
                        {
                                public void keyPressed(KeyEvent event) {}
                                public void keyReleased(KeyEvent event) {
                                        if (event.getKeyChar() == KeyEvent.VK_ESCAPE)
                                        {
                                                System.exit(0);
                                        }
                                }
                                public void keyTyped(KeyEvent event) {}
                        }
                        );
                        frame.setUndecorated(true);
                        JLabel label = new JLabel(".");
                        label.setBackground(Color.BLACK);
                        label.setForeground(Color.WHITE);
                        label.setOpaque(true);
                        label.setHorizontalAlignment(SwingConstants.CENTER);
                        frame.getContentPane().add(label);
                        GraphicsEnvironment.getLocalGraphicsEnvironment().
                                getDefaultScreenDevice().setFullScreenWindow(frame);
                        final int fontStyle = Font.BOLD;

                        label.setFont(new Font(fontName, fontStyle,
                                Math.min(fontSizeNumber, fontSizeText)));
                        new HappyNewYear(frame, label).run();
        }

        public void run()
        {
                boolean newYear = false;
                do
                {

                                newYear = true;
                                output = message;
                                }
                        else
                                {
                                // make a String from the number of seconds
                                output = formatter.format(remaining);
                                }
                                label.setText(output);
                                try
                                {
                                        Thread.sleep(1000);
        }
}
```



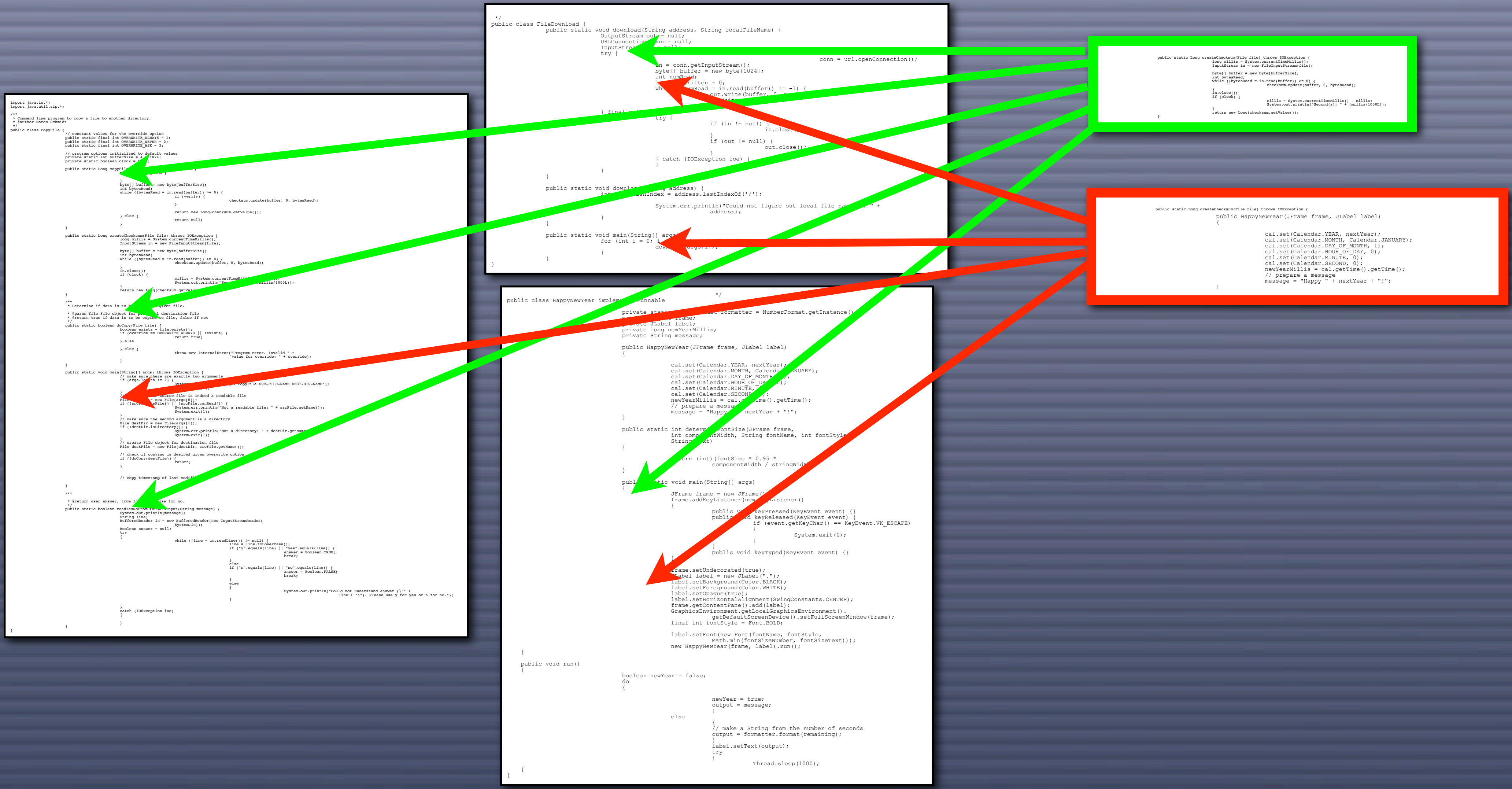
```
public static Long createChecksum(File file) throws IOException {
        long millis = System.currentTimeMillis();
        InputStream in = new FileInputStream(file);

        byte[] buffer = new byte[bufferSize];
        int bytesRead;
        while ((bytesRead = in.read(buffer)) >= 0) {
                checksum.update(buffer, 0, bytesRead);
        }
        in.close();
        if (clock) {
                millis = System.currentTimeMillis() - millis;
                System.out.println("Second(s): " + (millis/1000L));
        }
        return new Long(checksum.getValue());
}
```



```
public static Long createChecksum(File file) throws IOException {
        public HappyNewYear(JFrame frame, JLabel label)
        {

                cal.set(Calendar.YEAR, nextYear);
                cal.set(Calendar.MONTH, Calendar.JANUARY);
                cal.set(Calendar.DAY_OF_MONTH, 1);
                cal.set(Calendar.HOUR_OF_DAY, 0);
                cal.set(Calendar.MINUTE, 0);
                cal.set(Calendar.SECOND, 0);
                newYearMillis = cal.getTime().getTime();
                // prepare a message
                message = "Happy " + nextYear + "!";
        }
```

Pointcuts

Often point to details in source code (e.g. names)



```
public static Long createChecksum(File file) throws IOException {
                long millis = System.currentTimeMillis();
                InputStream in = new FileInputStream(file);

                byte[] buffer = new byte[bufferSize];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) >= 0) {
                                checksum.update(buffer, 0, bytesRead);
                }
                in.close();
                if (clock) {
                                millis = System.currentTimeMillis() - millis;
                                System.out.println("Second(s): " + (millis/1000L));
                }
                return new Long(checksum.getValue());
}
```

```
public static Long createChecksum(File file) throws IOException {
                public HappyNewYear(JFrame frame, JLabel label)
                {

                                cal.set(Calendar.YEAR, nextYear);
                                cal.set(Calendar.MONTH, Calendar.JANUARY);
                                cal.set(Calendar.DAY_OF_MONTH, 1);
                                cal.set(Calendar.HOUR_OF_DAY, 0);
                                cal.set(Calendar.MINUTE, 0);
                                cal.set(Calendar.SECOND, 0);
                                newYearMillis = cal.getTime().getTime();
                                // prepare a message
                                message = "Happy " + nextYear + "!";
                }
```

Aspects in Smalltalk

AspectS

- Framework based
- Pointcut using Smalltalk
- Advice using block

```
AsBeforeAfterAdvice
  qualifier:...
  pointcut: [WindowSensor withAllSubclasses
     select:
      [:each includesSelector: #eventDoubleClick:]
     thenCollect:[:each | AsJoinPointDescriptor
         targetClass: each
         targetSelector:#eventDoubleClick:]]
  beforeBlock: [:receiver :args :aspect :client |
                 ...]
```

Aspects in Smalltalk

AspectS

- Framework based
- Pointcut using Smalltalk
- Advice using block

```
AsBeforeAfterAdvice
   qualifier:...
   pointcut: [WindowSensor withAllSubclasses
      select:
       [:each includesSelector: #eventDoubleClick:]
      thenCollect:[:each | AsJoinPointDescriptor
          targetClass: each
          targetSelector:#eventDoubleClick:]]
   beforeBlock: [:receiver :args :aspect :client |
                   ...]
```

Pointcut

Aspects in Smalltalk

AspectS

- Framework based
- Pointcut using Smalltalk
- Advice using block

```
AsBeforeAfterAdvice
   qualifier:...
   pointcut: [WindowSensor withAllSubclasses
      select:
       [:each includesSelector: #eventDoubleClick:]
      thenCollect:[:each | AsJoinPointDescriptor
          targetClass: each
          targetSelector:#eventDoubleClick:]]
   beforeBlock: [:receiver :args :aspect :client |
                      ...]
```

Pointcut

Advice

Aspects in Carma

Carma

- Pointcut language
- Declarative Meta Programming
- Based on SOUL

```
after ?jp matching
    reception(?jp, #eventDoubleClick:,?args),
    within(?jp, ?class, ?selector),
    classInHierarchyOf(?class,[WindowSensor])
do
  Transcript show: ?class.
  ...
```

Aspects in Carma

Carma
- Pointcut language
- Declarative Meta Programming
- Based on SOUL

- Declarative
- Unification
- Recursion

```
after ?jp matching
     reception(?jp, #eventDoubleClick:,?args),
     within(?jp, ?class, ?selector),
     classInHierarchyOf(?class,[WindowSensor])
do
  Transcript show: ?class.
  ...
```

Aspects in Carma

Carma

- Pointcut language
- Declarative Meta Programming
- Based on SOUL

- Declarative
- Unification
- Recursion

```
after ?jp matching
    reception(?jp, #eventDoubleClick:,?args),
    within(?jp, ?class, ?selector),
    classInHierarchyOf(?class,[WindowSensor])
do
  Transcript show: ?class.
  ...
```

Pointcut

Aspects in Carma

Carma

- Pointcut language
- Declarative Meta Programming
- Based on SOUL

- Declarative
- Unification
- Recursion

Pointcut

```
after ?jp matching
     reception(?jp, #eventDoubleClick:,?args),
     within(?jp, ?class, ?selector),
     classInHierarchyOf(?class,[WindowSensor])
```

Advice

```
do
  Transcript show: ?class.
```

AspectSOUL

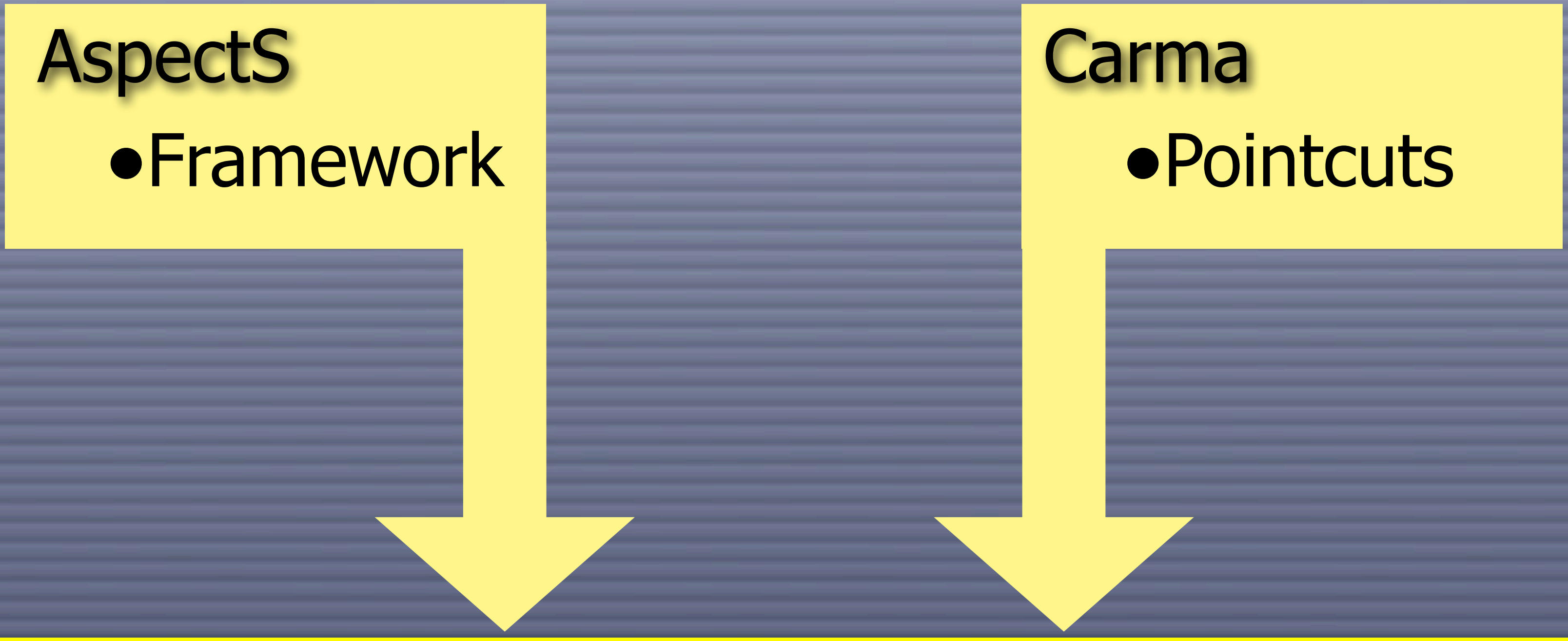
AspectS

- Framework

Carma

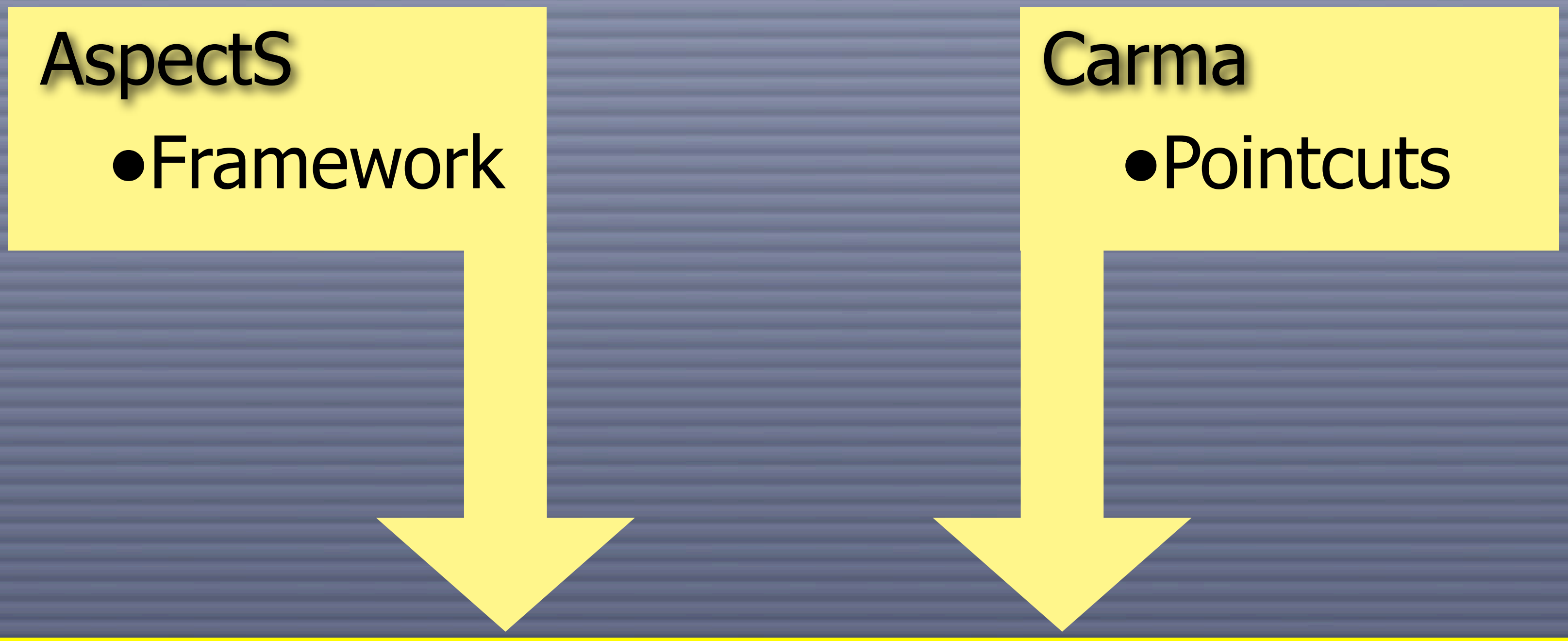
- Pointcuts

AspectSOUL



```
AsCARMABeforeAfterAdvice
  qualifier:...
  pointcutQuery: ‘reception(?jp, #eventDoubleClick:,?args),
    within(?jp, ?class, ?selector),
    classInHierarchyOf(?class,[WindowSensor])’
  beforeBlock: [:receiver :args :aspect :client |
                 ...]
```

AspectSOUL



```
AsCARMABeforeAfterAdvice
  qualifier:...
  pointcutQuery: 'reception(?jp, #eventDoubleClic
    within(?jp, ?class, ?selector),
    classInHierarchyOf(?class,[WindowSensor])'
  beforeBlock: [:receiver :args :aspect :client |
                 ...]
```

Pointcut

AspectSOUL

AspectS
- Framework

Carma
- Pointcuts

```
AsCARMABeforeAfterAdvice
  qualifier:...
  pointcutQuery: 'reception(?jp, #eventDoubleClic
    within(?jp, ?class, ?selector),
    classInHierarchyOf(?class,[WindowSensor])'
  beforeBlock: [:receiver :args :aspect :client |
                 ...]
```

Pointcut

Advice

Fragility

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

1. accessing protocol
2. selector corresponds to variable name

Fragility

1. accessing protocol
2. selector corresponds to variable name

```
Person>>name
    ^name

Person>>name: anObject
    name := anObject
```

Synchronization

```
class(?class),
methodWithNameInClass(?method,?accessor,?class),
instanceVariableInClassChain(?accessor,?class),
methodInProtocol(?method,accessing),
reception(?joinpoint,?accessor,?args),
withinClass(?joinpoint,?class)
```

Fragility

1. accessing protocol
2. selector corresponds to variable name

```
Person>>name
    ^name

Person>>name: anObject
    name := anObject
```

```
Person>>getName
    ^name

Person>>setName: anObject
    name := anObject
```

Synchronization

```
class(?class),
methodWithNameInClass(?method,?accessor,?class),
instanceVariableInClassChain(?accessor,?class),
methodInProtocol(?method,accessing),
reception(?joinpoint,?accessor,?args),
withinClass(?joinpoint,?class)
```

Fragility

1. accessing protocol
2. selector corresponds to variable name

~~**Person>>name**~~
~~^name~~

~~**Person>>name:** anObject~~
~~name := anObject~~

? ?

```
Person>>getName
    ^name

Person>>setName: anObject
    name := anObject
```

Synchronization

```
class(?class),
methodWithNameInClass(?method,?accessor,?class),
instanceVariableInClassChain(?accessor,?class),
methodInProtocol(?method,accessing),
reception(?joinpoint,?accessor,?args),
withinClass(?joinpoint,?class)
```

Complexity

1. accessor: return instance var
2. mutator: assign instance var

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

Complexity

1. accessor: return instance var
2. mutator: assign instance var

```
Person>>name
    ^name

Person>>name: anObject
    name := anObject
```

Synchronization

```
class(?class),
methodWithNameInClass(?method,?accessor,?class),
instanceVariableInClassChain(?accessor,?class),
returnStatement(?method,variable(?var)),
reception(?joinpoint,?accessor,?args),
withinClass(?joinpoint,?class)
```

Complexity

1. accessor: return instance var
2. mutator: assign instance var

```
Person>>name
    ^name

Person>>name: anObject
    name := anObject
```

```
Person>>friends
    ^friends isNil
        ifTrue:[friends := Set new]
        ifFalse:[friends]
```

?

Synchronization

```
class(?class),
methodWithNameInClass(?method,?accessor,?class),
instanceVariableInClassChain(?accessor,?class),
returnStatement(?method,variable(?var)),
reception(?joinpoint,?accessor,?args),
withinClass(?joinpoint,?class)
```

Complexity

1.accessor: return instance var
2.mutator: assign instance var

```
Person>>name
    ^name

Person>>name: anObject
    name := anObject
```

```
Person>>friends
    ^friends isNil
        ifTrue:[friends := Set new]
        ifFalse:[friends]
```

?

Synchronization

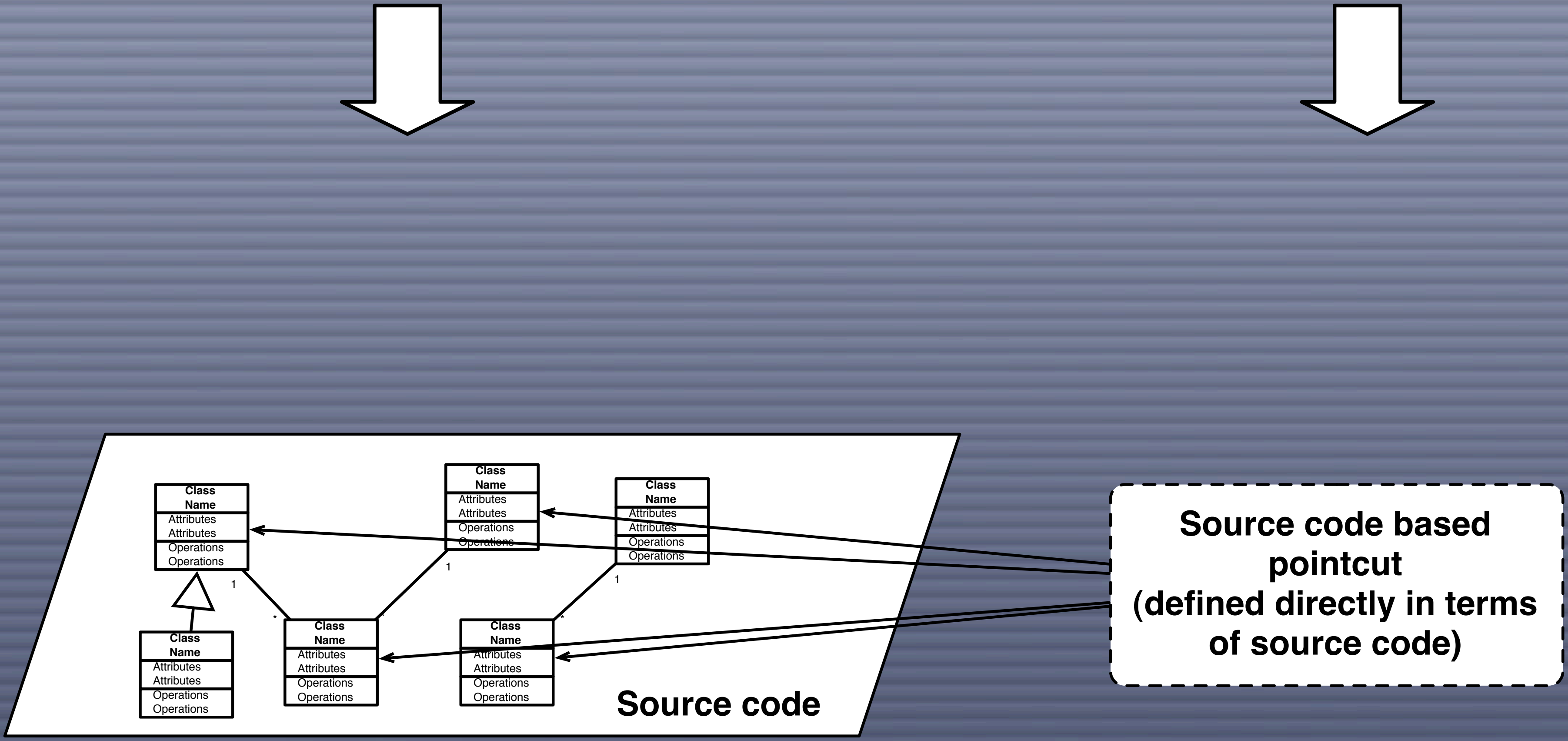
```
class(?class),
methodWithNameInClass(?method,?accessor,?class),
instanceVariableInClassChain(?accessor,?class),
returnStatement(?method,variable(?var)),
reception(?joinpoint,?accessor,?args),
withinClass(?joinpoint,?class)
```

withinClass(?joinpoint,?class)
withinClass(?joinpoint,?class)

Problem Analysis

Base program developer

Aspect developer

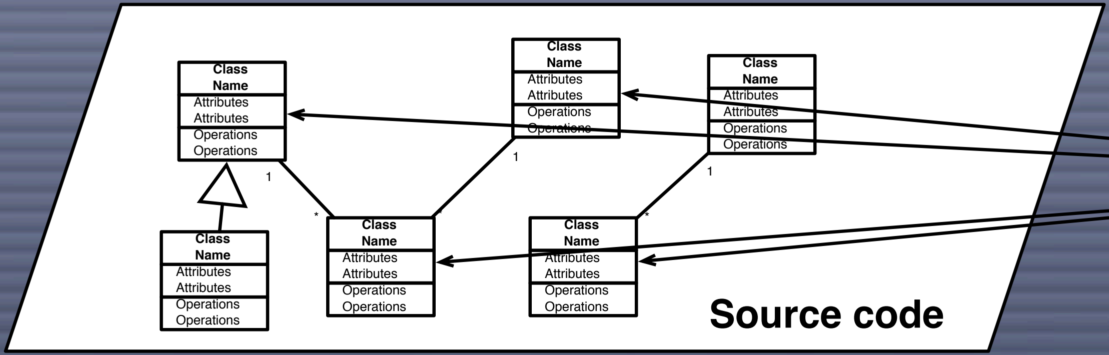


Class Name
Attributes
Operations
1
*
Source code
Source code based pointcut (defined directly in terms of source code)

Problem Analysis

Base program developer

Aspect developer



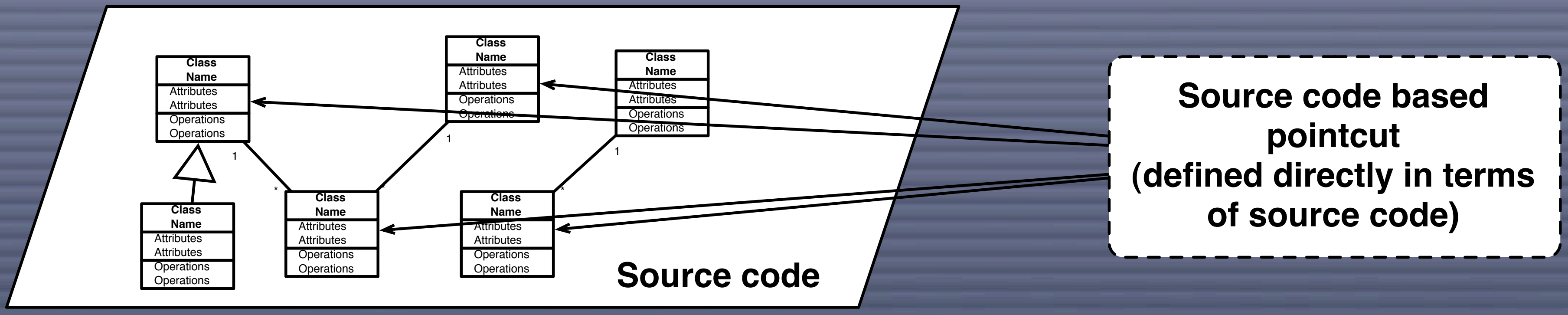
Source code based pointcut (defined directly in terms of source code)

Tight coupling

Problem Analysis

Base program developer

Aspect developer

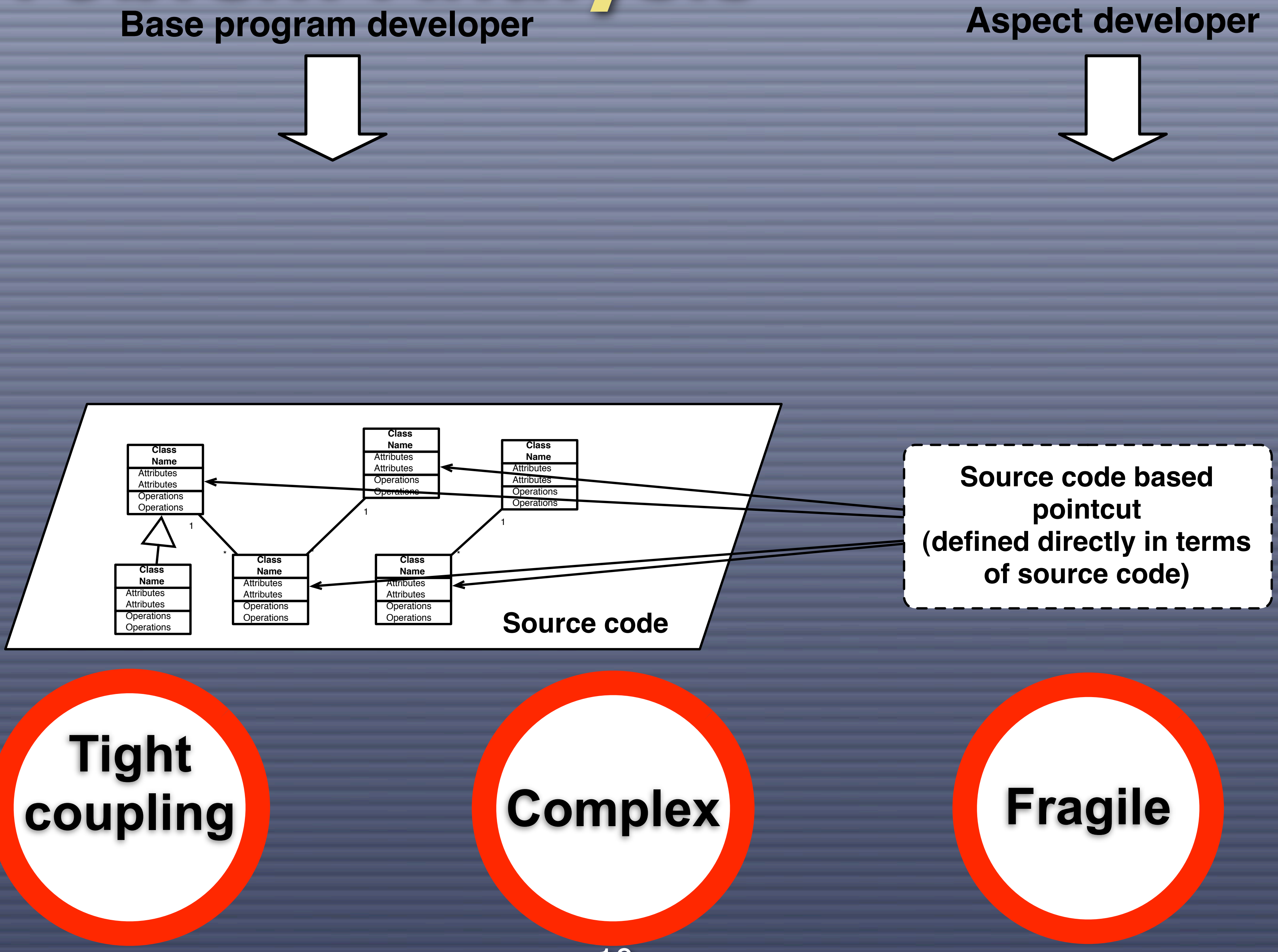


Class Name
Attributes
Attributes
Operations
Operations
1
*
Source code
Source code based pointcut (defined directly in terms of source code)

Tight coupling

Complex

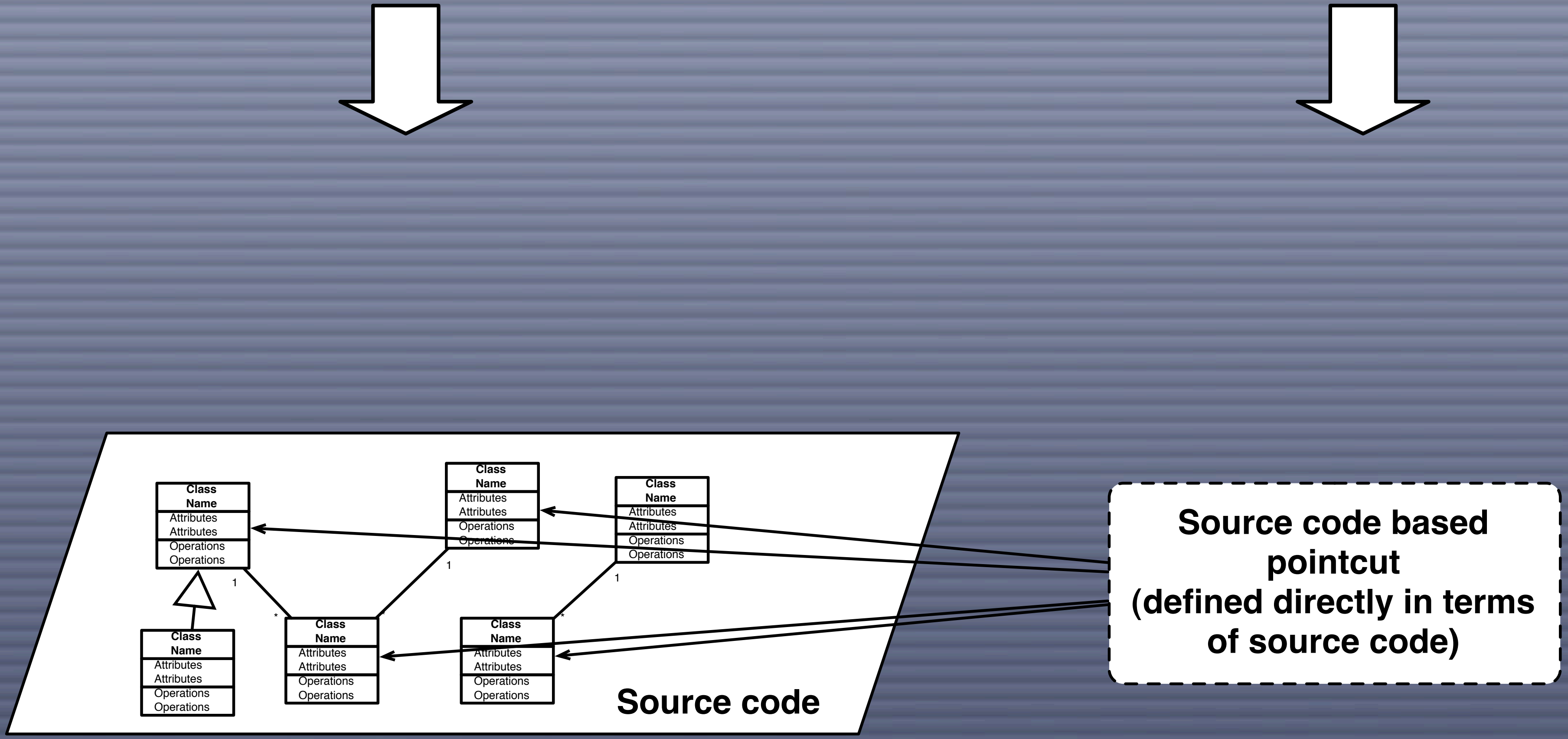
Problem Analysis



Application-specific

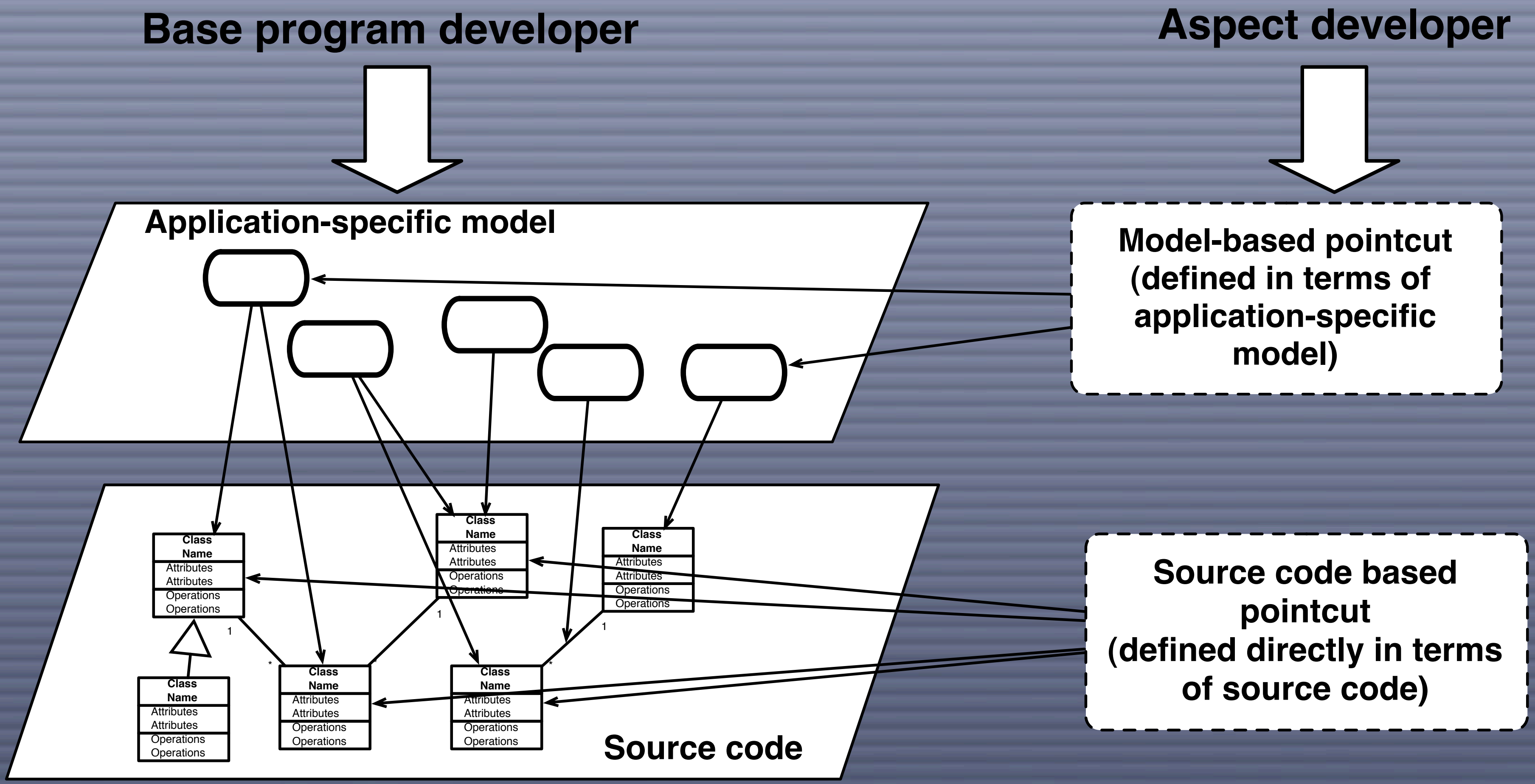
Base program developer

Aspect developer



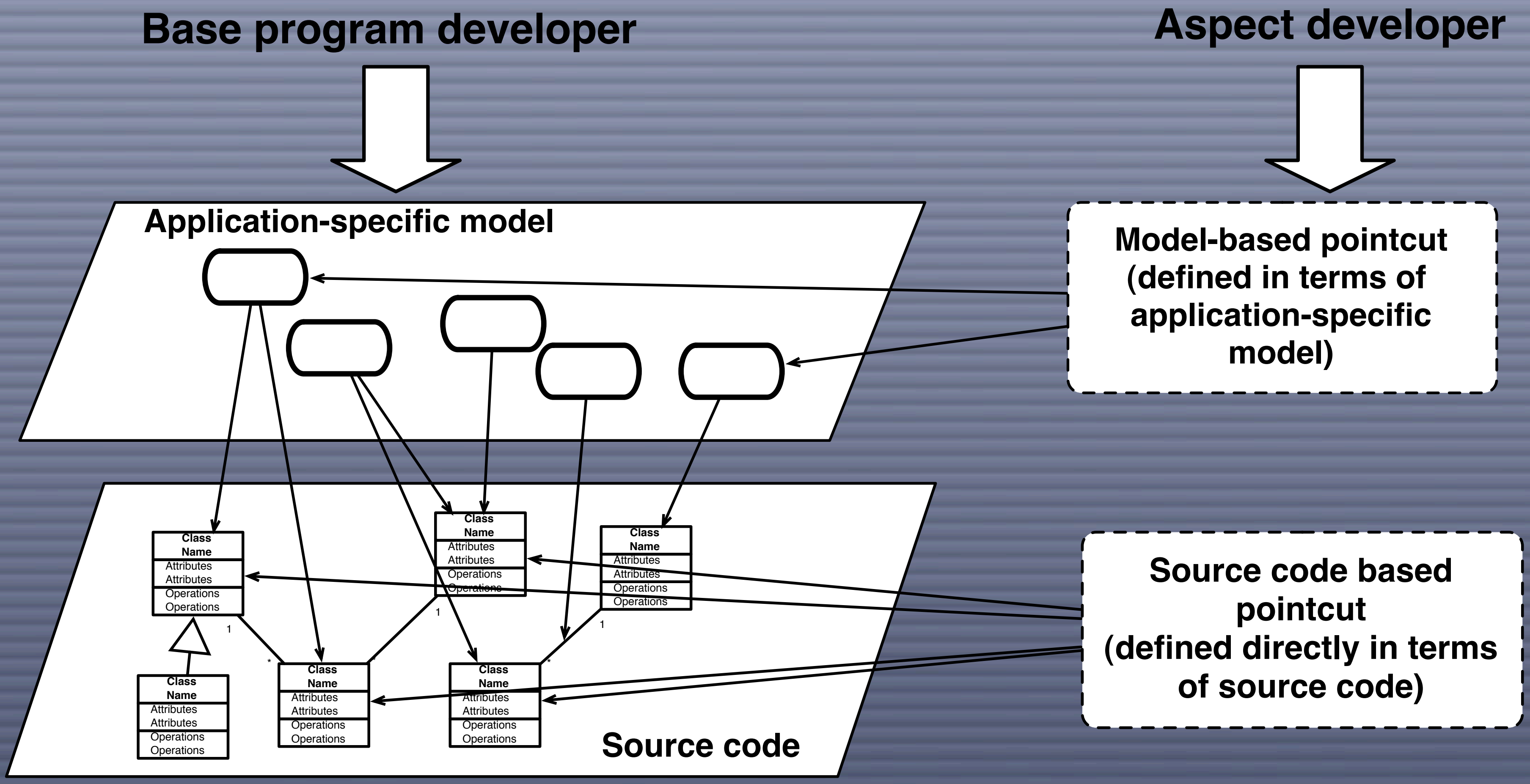
Class Name
Attributes
Attributes
Operations
Operations
1
*
Source code
Source code based pointcut (defined directly in terms of source code)

Application-specific

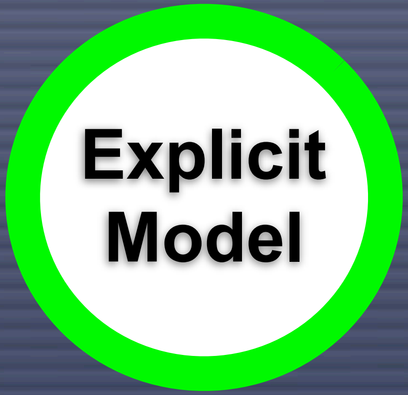


Base program developer
Aspect developer
Application-specific model
Model-based pointcut (defined in terms of application-specific model)
Class Name
Attributes
Operations
1
*
Source code
Source code based pointcut (defined directly in terms of source code)

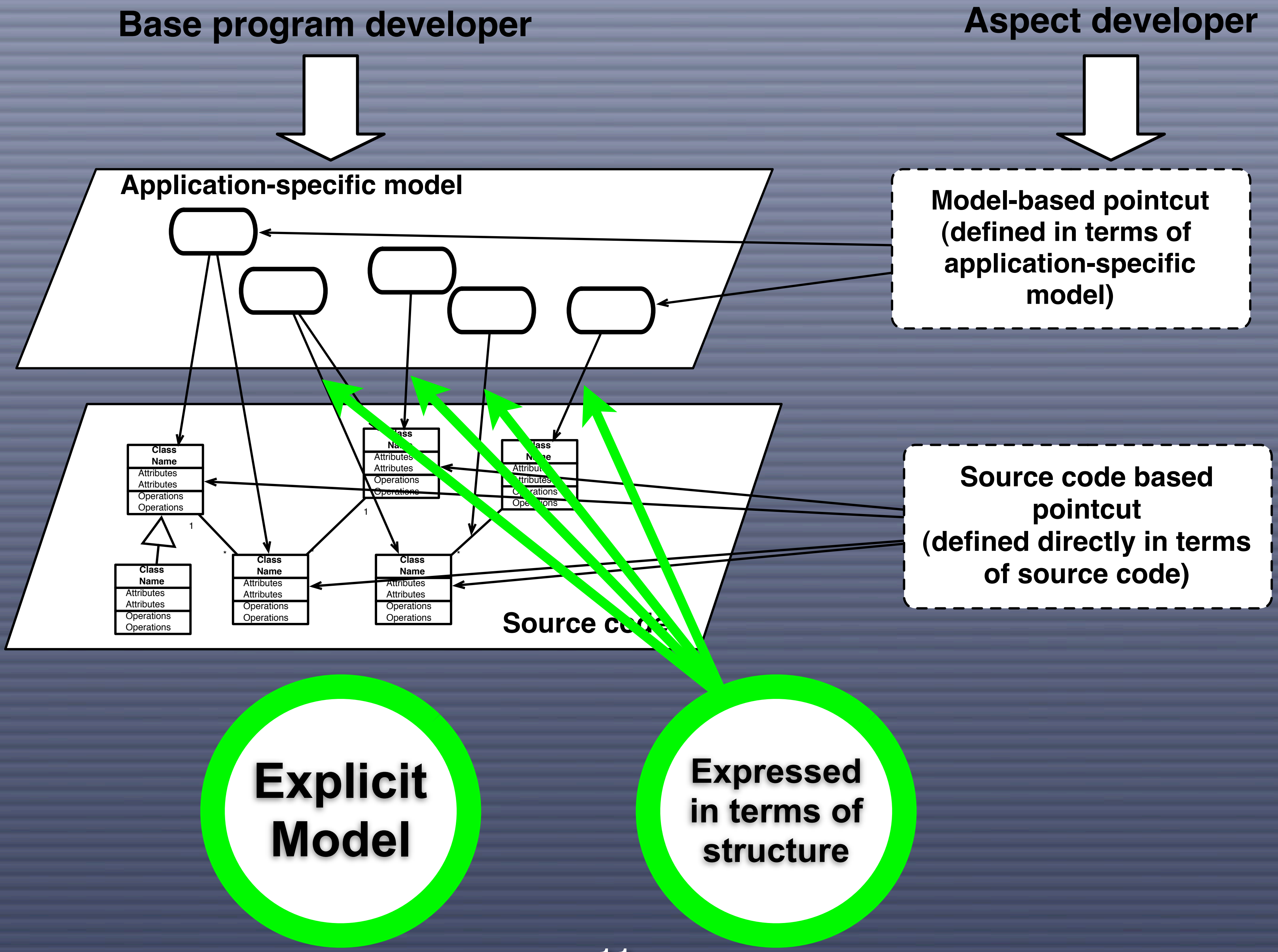
Application-specific



Base program developer
Aspect developer
Application-specific model
Model-based pointcut (defined in terms of application-specific model)
Class Name
Attributes
Operations
1
*
Source code
Source code based pointcut (defined directly in terms of source code)

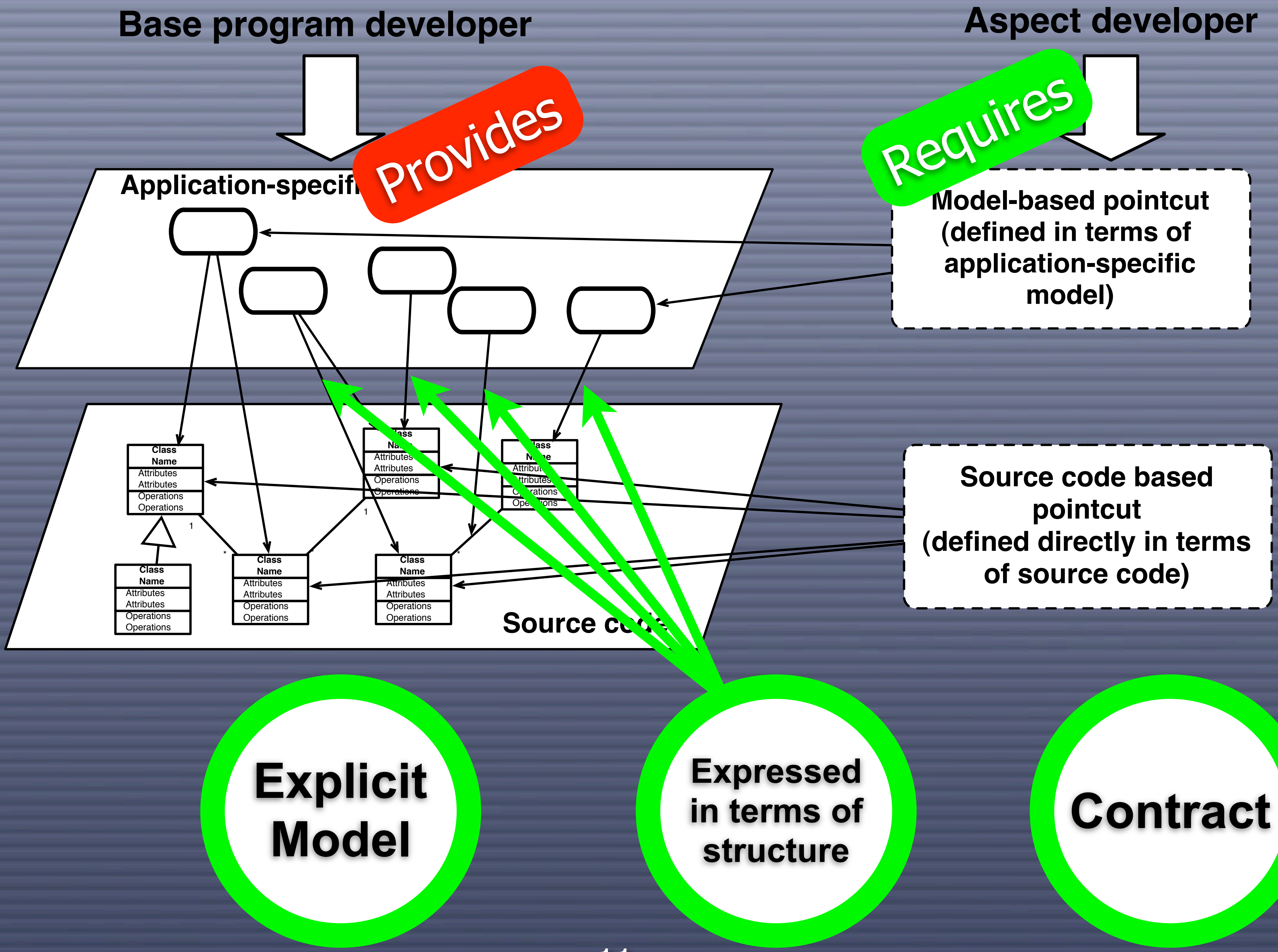


Application-specific



Base program developer
Aspect developer
Application-specific model
Model-based pointcut (defined in terms of application-specific model)
Class Name
Attributes
Attributes
Operations
Operations
1
*
Source code
Source code based pointcut (defined directly in terms of source code)
Explicit Model
Expressed in terms of structure

Application-specific


Base program developer
Aspect developer
Provides
Requires
Application-specifi
Model-based pointcut (defined in terms of application-specific model)
Class Name
Attributes
Attributes
Operations
Operations
1
Source code based pointcut (defined directly in terms of source code)
Source code
Explicit Model
Expressed in terms of structure
Contract

Application-specific pointcuts in AspectSOUL

Logic Pointcuts

- Extensible pointcut language

Application-specific model

- Keep in sync with code
- Logic representation
  - specialisation
  - parameters & unification

Application-specific pointcuts in AspectSOUL

Logic Pointcuts

- Extensible pointcut language

Application-specific model

- Keep in sync with code
- Logic representation
  - specialisation
  - parameters & unification

Application-specific pointcuts in AspectSOUL

Logic Pointcuts

- Extensible pointcut language

Application-specific model

- Keep in sync with code
- Logic representation
  - specialisation
  - parameters & unification

Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts
Andy Kellens, Kim Mens, Johan Brichau, Kris Gybels
In "Proceedings of the 20th European Conference on Object-Oriented Programming (ECOOP)", 2006

Accessors

Source

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

Pointcut

Model

Accessors

Source

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

Pointcut

Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

Model

Accessors

Source

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

Synchronization

Pointcut

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

Model

```
accessor(?class,?method,?varname) if
   class(?class),
   instanceVariableInClassChain(?varName,?class),
   methodWithNameInClass(?method,?varName,?class),
   methodInProtocol(?method,accessing),
   accessorForm(?method,?varname)
```

```
accessorForm(?method,?var) if
   returnStatement(?method,variable(?var))
```

Model specialisation

Source

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

Pointcut

Model

Model specialisation

Source

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

```
Person>>friends
   ^friends isNil
      ifTrue:[friends := Set new]
      ifFalse:[friends]
```

Pointcut

Model

Model specialisation

Source

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

```
Person>>friends
   ^friends isNil
      ifTrue:[friends := Set new]
      ifFalse:[friends]
```

Pointcut

Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

Model

Model specialisation

Source

```
Person>>name
   ^name

Person>>name: anObject
   name := anObject
```

```
Person>>friends
   ^friends isNil
      ifTrue:[friends := Set new]
      ifFalse:[friends]
```

Pointcut

Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

Model

```
accessor(?class,?method,?varname) if
   ...
```

```
accessorForm(?method,?var) if
 returnStatement(?method,send(?var))

accessorForm(?method,?var) if
 returnStatement(?method,send(?check),<?true,?false>),
 nilCheckStatement(?check),
 statementsOfBlock(assign(?var,?varinit),?true),
 statementsOfBlock(<?var>,?false)
```

Parameters & Unification

Parameters & Unification

Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

Parameters & Unification

Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(Array,at:put:,?var),
withinClass(?joinpoint,?class)
```

Parameters & Unification

Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

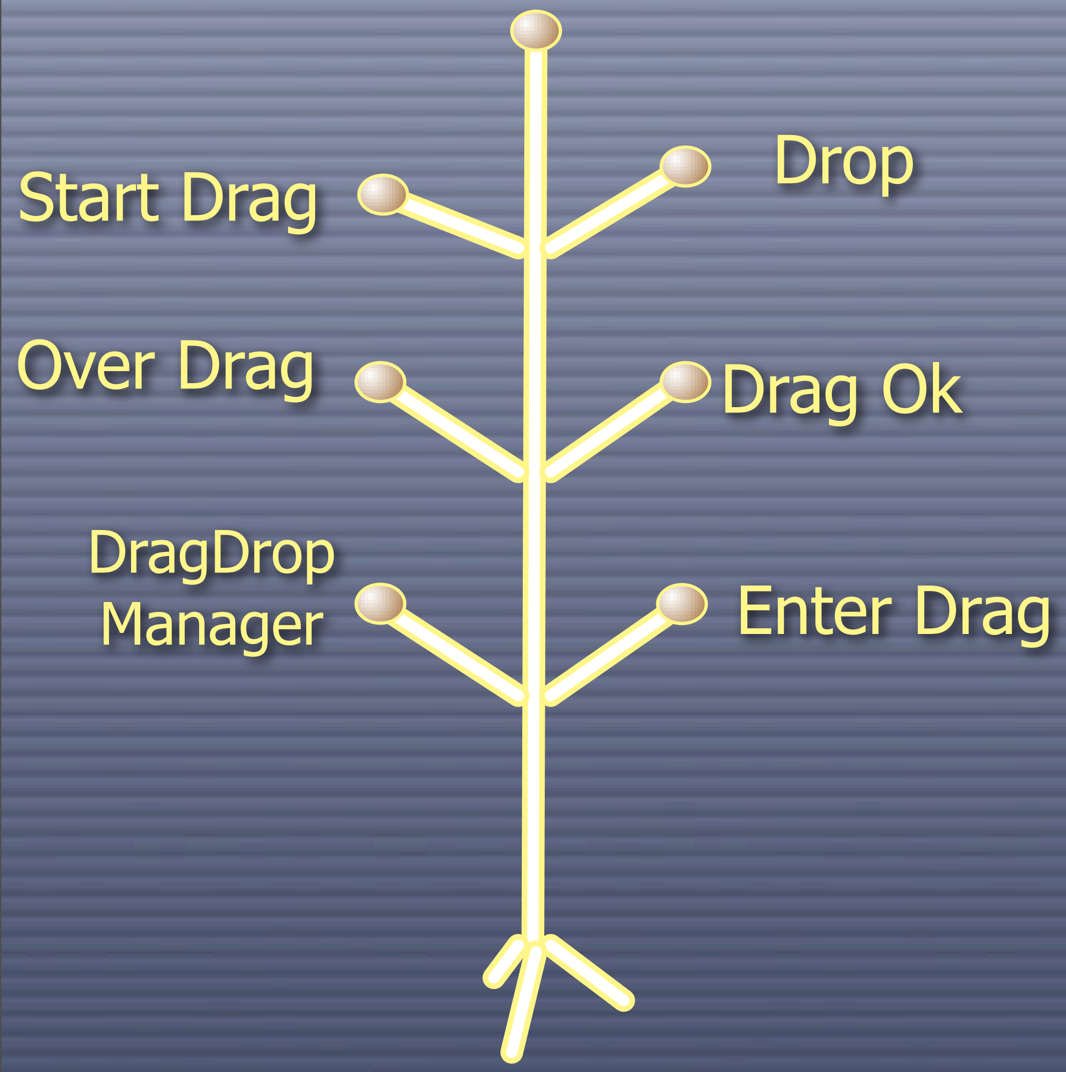
Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(Array,at:put:,?var),
withinClass(?joinpoint,?class)
```

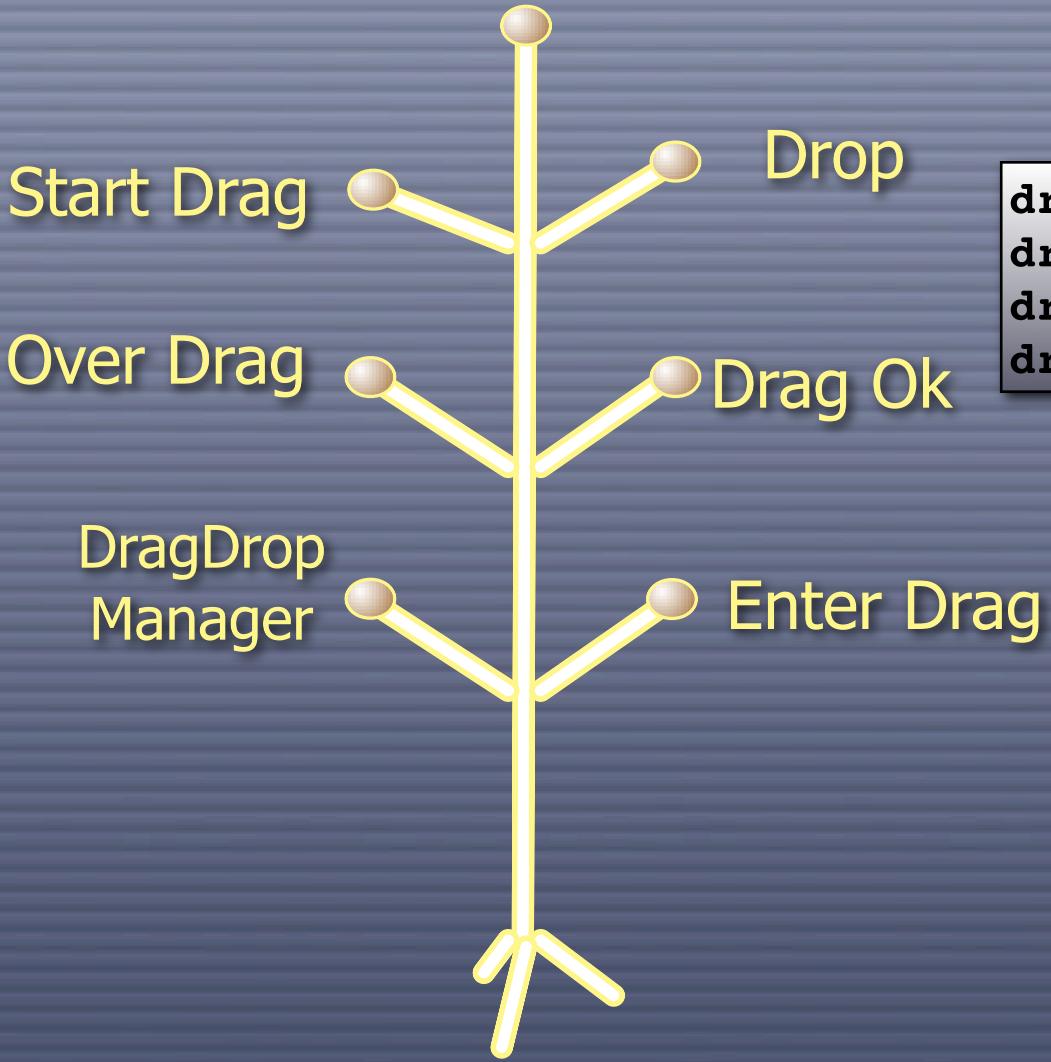
Synchronization

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,address),
withinClass(?joinpoint,?class)
```

Drag & Drop

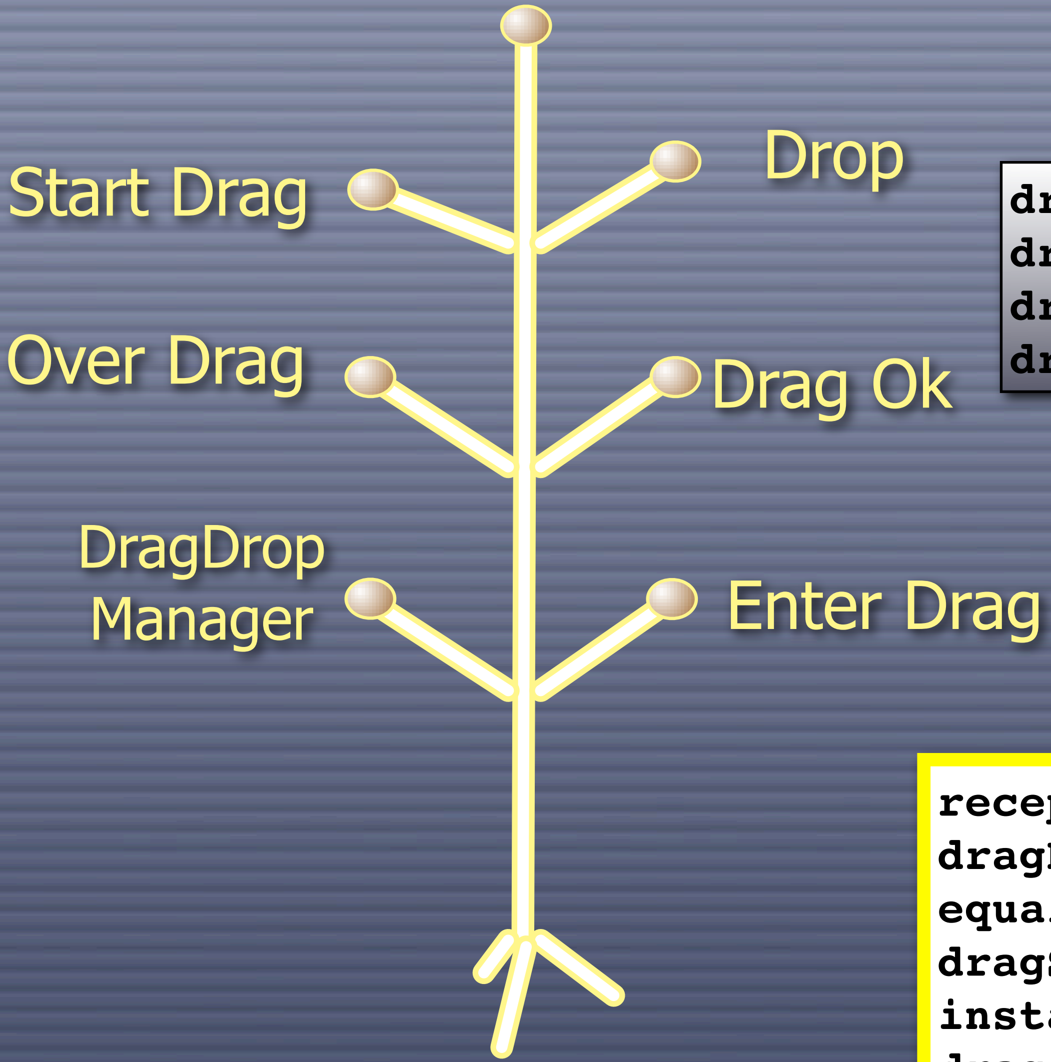

Start Drag
Drop
Over Drag
Drag Ok
DragDrop Manager
Enter Drag

Drag & Drop



```
dragOkMethod(?class,?sel,?component)
dragEnterMethod(?class,?sel,?component)
dragSource(?dragdropmanager,?source)
draggedObject(?dragdropmanager,?object)
```

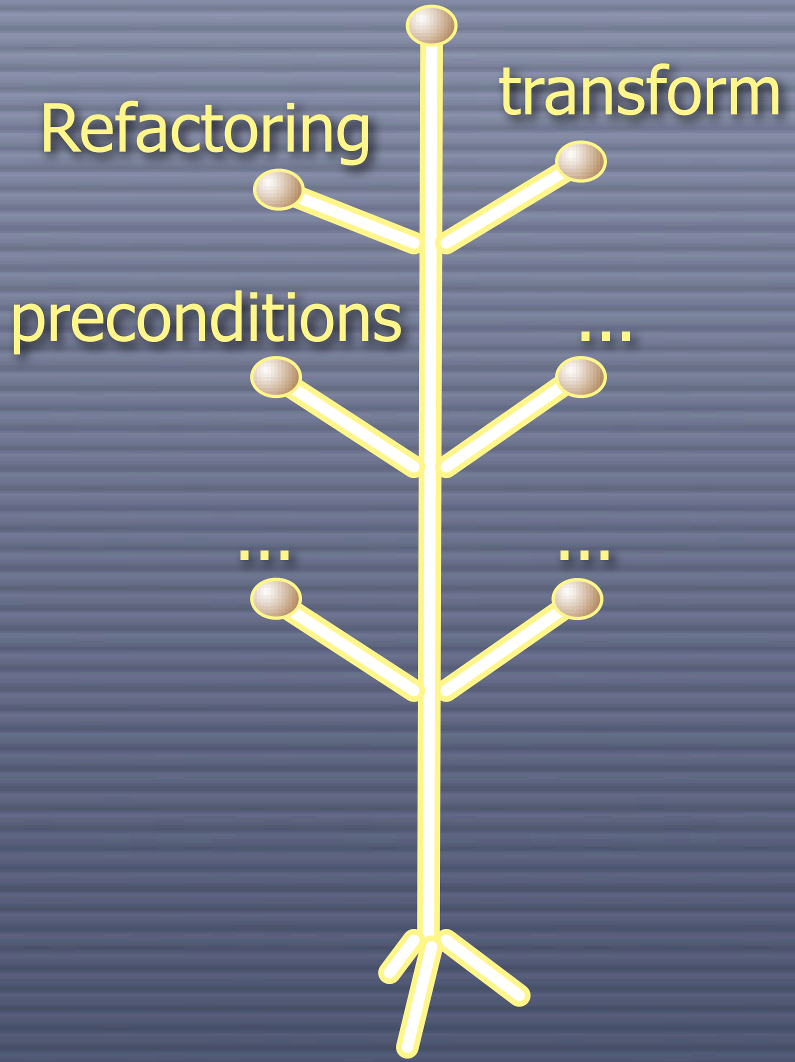
Drag & Drop



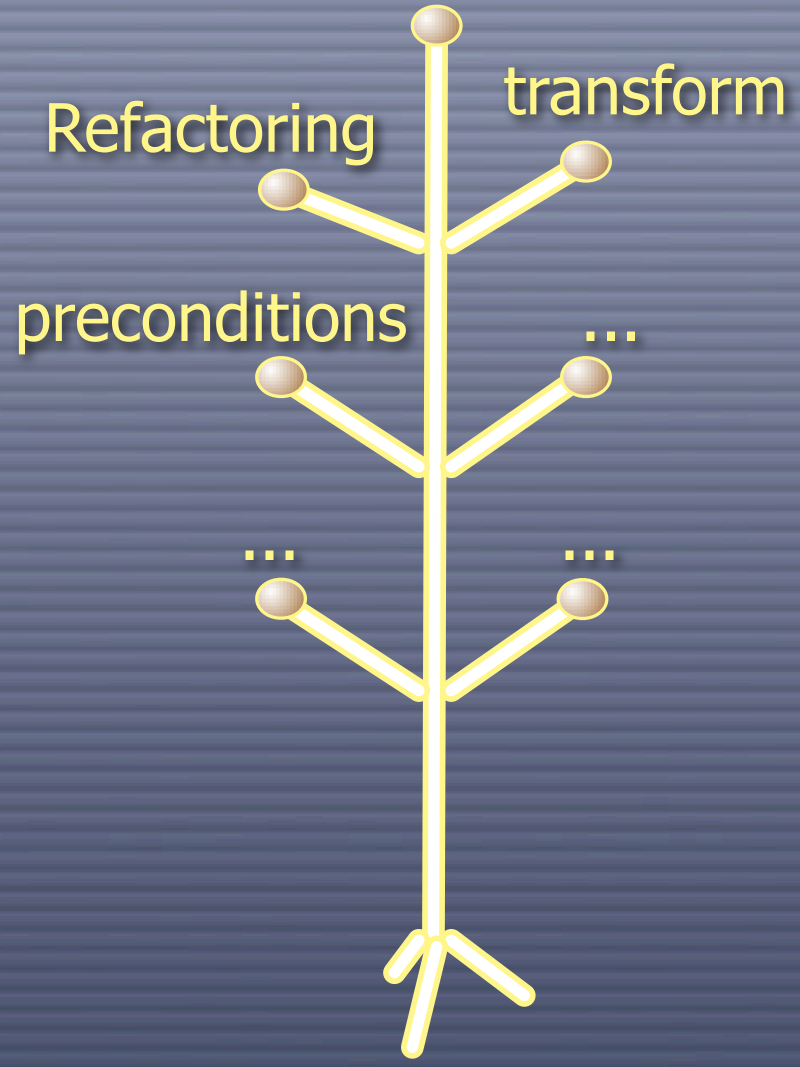
```
dragOkMethod(?class,?sel,?component)
dragEnterMethod(?class,?sel,?component)
dragSource(?dragdropmanager,?source)
draggedObject(?dragdropmanager,?object)
```

```
reception(?jp,?sel,?args),
dragEnterMethod(?class,?sel,?component),
equals(?args,<?dragdropmanager>),
dragSource(?dragdropmanager,?source),
instanceOf(?source,FigureManager),
draggedObject(?dragdropmanager,?object),
instanceOf(?object,Line)
```

Refactoring

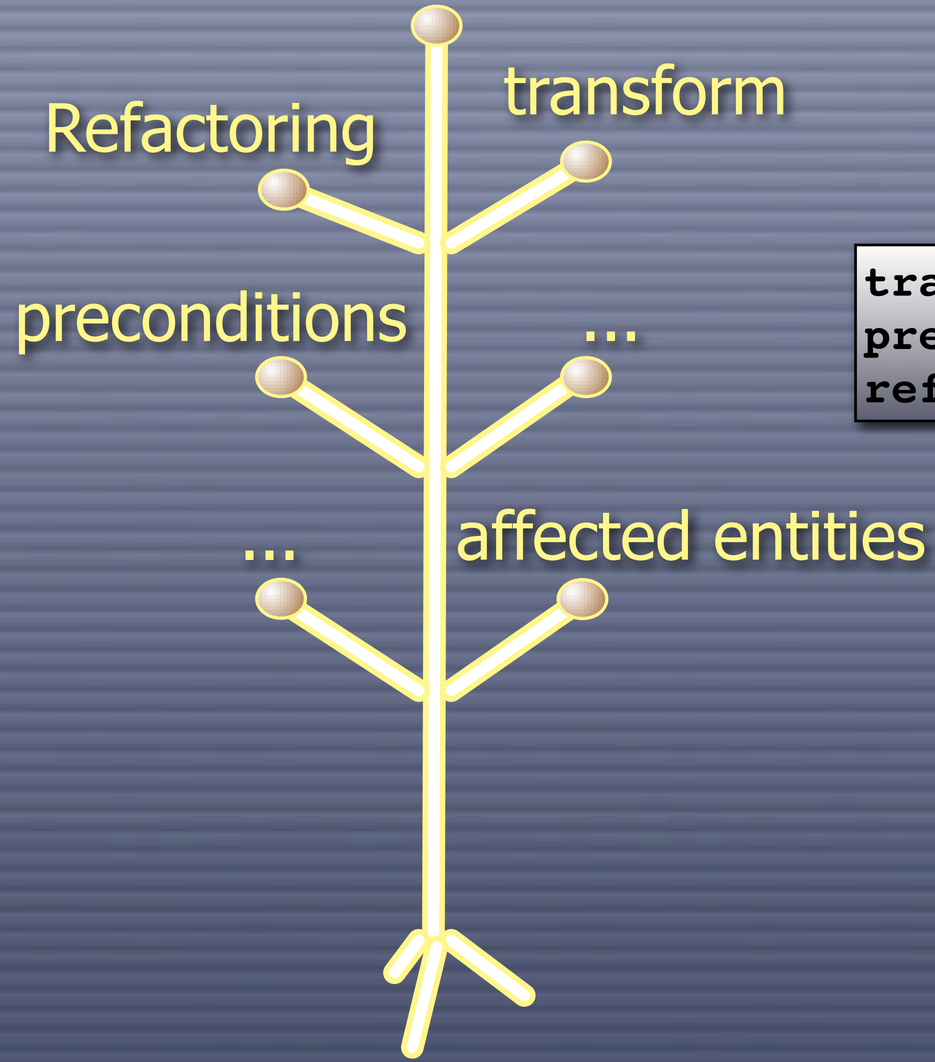


Refactoring



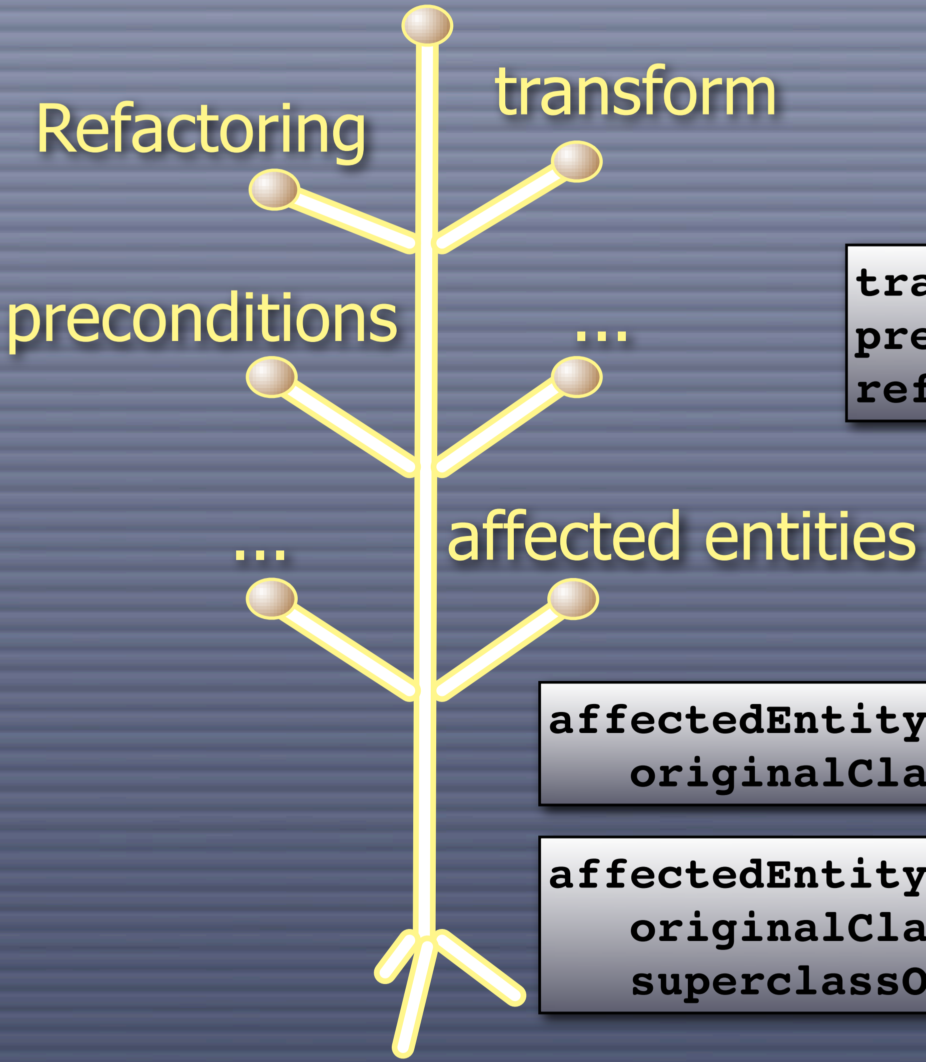
```
transformMethod(?class,?sel,?refactoring)
preconditions(?refactoring,?preconditions)
refactoring(?refactoring,?class,?method)
```

Refactoring



```
transformMethod(?class,?sel,?refactoring)
preconditions(?refactoring,?preconditions)
refactoring(?refactoring,?class,?method)
```

Refactoring



```
transformMethod(?class,?sel,?refactoring)
preconditions(?refactoring,?preconditions)
refactoring(?refactoring,?class,?method)
```

```
affectedEntity(?ref,[PushUpMethod],?input,?entity) if
   originalClassOfPushUpMethod(?input,?entity)
```

```
affectedEntity(?ref,[PushUpMethod],?input,?entity) if
   originalClassOfPushUpMethod(?input,?class),
   superclassOf(entity,?class)
```

Summary

- AspectS + Carma = AspectSOUL
- Pointcut in terms of application-specific model
- Extensible pointcut language
- Logic language to express model

- Further integration of AspectS/Carma
- Integration with support for fragile pointcuts

Questions

?

- More information:
  - http://prog.vub.ac.be/carma/