

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

```
import java.io.*;
import java.net.*;
/* command line program to copy a file to another directory.
 * Author: Marco Schmid
 */
public class Copyfile {
    /* constant values for the override option
     * -O or --override
     */
    public static final int OVERRIDE = 1;
    public static final int OVERWRITE = 2;
    /* program options
     * -c or --checksum
     */
    private static final int CHECKSUM = 4;
    /* program options
     * -t or --timestamp
     */
    private static final int COPIESTAMPS = 8;
    /* program options
     * -h or --help
     */
    private static final int HELP = 16;
    /* command line program to copy a file to another directory.
     * Author: Marco Schmid
     */
    public static long copyFile(File source, File destDir) throws IOException {
        FileOutputStream fos = new FileOutputStream(destDir);
        long millis = System.currentTimeMillis();
        if (source.isDirectory()) {
            if (destDir.exists()) {
                checksum = new MD5();
                if (destDir.isDirectory()) {
                    for (File f : destDir.listFiles()) {
                        if (f.isDirectory())
                            checksum.update(f.getAbsolutePath());
                        else
                            fos.write(new FileInputStream(f));
                    }
                } else {
                    fos.write(new FileInputStream(destDir));
                }
            } else {
                destDir.mkdirs();
                if (destDir.exists())
                    checksum = new MD5();
                fos.write(new FileInputStream(source));
                while ((numRead = in.read(buffer)) >= 0) {
                    checksum.update(buffer, 0, numRead);
                    fos.write(buffer, 0, numRead);
                }
            }
        } else {
            if (destDir.exists()) {
                millis = System.currentTimeMillis();
                System.out.println(millis);
                if (destDir.isDirectory()) {
                    destDir.mkdirs();
                    if (destDir.exists())
                        return new Long(checksum.getvalue());
                } else {
                    return null;
                }
            }
        }
        public static long checksumOfFile(File file) throws IOException {
            FileInputStream fis = new FileInputStream(file);
            long millis = System.currentTimeMillis();
            if (fis.exists()) {
                byte[] buffer = new byte[bufferSize];
                while ((numRead = fis.read(buffer)) >= 0) {
                    checksum.update(buffer, 0, numRead);
                }
            } else {
                millis = System.currentTimeMillis();
                System.out.println(millis);
                return new Long(checksum.getvalue());
            }
        }
        /* determine if data is to be copied to given file
         * Take into consideration override option and
         * check if file exists and if file is a directory
         * If true, then file is to be copied to file, false if not
         */
        public static boolean relate(File file, String override) {
            if (file.exists() || file.isDirectory())
                return true;
            if (override == OVERRIDE)
                return false;
            if (override == OVERWRITE)
                throw new FileNotFoundException("File exists. -O");
            else
                throw new FileNotFoundException("Value for override: " + override);
        }
        public static void main(String[] args) throws IOException {
            if (args.length < 2)
                System.out.println("Usage: copyfile SRC-DIR-OR-FILE DEST-DIR-NAME");
            /* make sure the source file is in fact a readable file
             * file scrFile = new File(args[0]);
             * if (!scrFile.exists() || !scrFile.isFile() || !scrFile.canRead())
             *     System.out.println("Not a readable file: " + scrFile.getName());
            */
            /* make sure the destination is a directory
             * File destDir = new File(args[1]);
             * if (!destDir.exists() || !destDir.isDirectory())
             *     System.out.println("Not a directory: " + destDir.getName());
            */
            /* create file object for destination file
             * if (!destDir.exists())
             *     destDir.mkdirs();
            */
            /* check if copying in desired given overwrite option
             */
            /* copy file, optionally creating checksum
             */
            /* copy timestamp of last modification
             */
            if (copyFile(args[0], new File(args[1]))) {
                if (args.length > 2)
                    System.out.println("Timestamp of copied file.");
            }
            /* optionally verify file
             */
            if (args.length > 2)
                System.out.println("Verifying destination file...");
            if (args.length > 3)
                System.out.println("Comparing checksums with file to be compared by user");
            if (args.length > 4)
                System.out.println("Comparing timestamps with file to be compared by user");
            public static boolean recheckChecksum(String message) {
                BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
                boolean answer = null;
                String line;
                while ((line = in.readLine()) != null) {
                    if ("y".equals(line) || "yes".equals(line))
                        answer = true;
                    if ("n".equals(line) || "no".equals(line))
                        answer = false;
                    if ("q".equals(line) || "quit".equals(line))
                        System.out.println("Quitting");
                }
                if (answer == null)
                    throw new IOException("Unexpected end of input from stdin.");
            }
            in.close();
            return answer;
        }
        catch (IOException ioe)
            throw new InterruptedException("Cannot read from stdin or write to stdout.");
    }
}
```

```
/*
 * 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 + " (" + 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 >= 44 & address.length() - 1) {
        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]);
    }
}

public class HappyNewYear implements Runnable {
    /*
     * This class implements a simple GUI application for the new year.
     */
    private static NumberFormat formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYear;
    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, 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);
        newYear = cal.getTimeInMillis();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
    public static int determineFontSize(JFrame frame,
                                       int componentWidth, String fontName, int fontStyle,
                                       String text) {
        int fontCreate = componentWidth * 2 / text.length();
        Font font = new Font(fontName, fontStyle, fontCreate);
        FontMetrics fontMetrics = frame.getFontMetrics(font);
        int stringWidth = fontMetrics.stringWidth(text);
        return (int) (fontCreate * stringWidth / componentWidth);
    }
    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.setLayout(null);
        JLabel label = new JLabel("1");
        label.setForeground(Color.BLACK);
        label.setBackground(Color.WHITE);
        label.setHorizontalAlignment(SwingConstants.CENTER);
        frame.add(label);
        GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
        ge.getDefaultScreenDevice().setFullScreenWindow(frame);
        final String fontName = "SansSerif";
        int fontCreate = determineFontSize(frame,
                                           Toolkit.getDefaultToolkit().getScreenSize().width,
                                           Toolkit.getDefaultToolkit().getScreenSize().height,
                                           fontName);
        int fontCreateText = determineFontSize(frame,
                                              Toolkit.getDefaultToolkit().getScreenSize().width,
                                              Toolkit.getDefaultToolkit().getScreenSize().height,
                                              fontName);
        label.setFont(new Font(fontName, fontCreate,
                               fontCreateText, fontCreateText));
        new HappyNewYear(frame, label).run();
    }
    public void run() {
        boolean newYear = false;
        do {
            long time = System.currentTimeMillis();
            long remaining = (newYear - 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

```

    */
public class FileDownload {
    public static void download(String address, String localFileName) {
        try {
            URL url = new URL(address);
            URLConnection conn = url.openConnection();
            conn.connect();
            InputStream in = conn.getInputStream();
            OutputStream out = new FileOutputStream(localFileName);
            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 e) {
            e.printStackTrace();
        } finally {
            try {
                if (in != null)
                    in.close();
                if (out != null)
                    out.close();
            } catch (IOException e) {
            }
        }
    }

    public static void download(String address) {
        int lastSlashIndex = address.lastIndexOf('/');
        if (lastSlashIndex == -1)
            lastSlashIndex = address.length() - 1;
        else {
            System.err.println("Warning: 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) {
        // store argument GUI elements
        this.frame = frame;
        this.label = label;
        // compute beginning of next year
        Calendar cal = Calendar.getInstance();
        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.getTimeInMillis();
        // prepare message
        message = "Happy New Year at " + newYearMillis;
    }

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

    public static void main(String[] args) {
        frame = new JFrame();
        frame.addKeyListener(new KeyListener());
        frame.setVisible(true);
        public void keyPressed(KeyEvent event) {}
        public void keyReleased(KeyEvent event) {}
        public void keyTyped(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.setHorizontalAlignment(SwingConstants.CENTER);
    label.setFont(new Font("Times New Roman", Font.PLAIN, 16));
    GraphicsEnvironment.getLocalGraphicsEnvironment().createFontRenderingHint(RenderingHints.VALUE_RENDER_QUALITY);
    final int fontStyle = Font.BOLD;
    final String fontName = "Times New Roman";
    int fontWidth = determineFontSize(frame,
                                      Toolkit.getDefaultToolkit().getScreenSize().width,
                                      fontName, fontStyle, "Happy New Year");
    int fontHeight = determineFontSize(frame,
                                      Toolkit.getDefaultToolkit().getScreenSize().width,
                                      fontName, fontStyle, "Happy New Year");
    label.setFont(new Font(fontName, fontStyle, fontWidth));
    label.setText(message);
    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

Crosscutting Concerns

```
import java.io.*;
import java.net.*;
/*
 * command line program to copy a file to another directory.
 * Author: Marco Schmid
 */
public class Copyfile {
    /* constant values for the override option
     * -O or --override
     */
    public static final int OVERRIDE = 1;
    public static final int OVERWRITE = 2;
    /* progress option
     * -P or --progress
     */
    public static final int PROGRESS = 4;
    private static boolean copyTimestamps = true;
    private static int maxBufferSize = 1024;
    public static long copyFile(File sourceFile, File destFile) {
        /* constant values for the override option
         * -O or --override
         */
        final int OVERRIDE = 1;
        final int OVERWRITE = 2;
        /* progress option
         * -P or --progress
         */
        final int PROGRESS = 4;
        /* determine if data is to be copied to given file
         * Take into consideration override option and
         * check if file exists and if it is a file
         * If file exists, then check if it is a file
         * If true, then check if file is to be copied to file, false if not
         */
        final static boolean isFile(File file) {
            if (file.isDirectory()) {
                System.out.println("File " + file.getName() + " is a directory");
                return false;
            }
            if (file.exists()) {
                if (file.isDirectory()) {
                    System.out.println("File " + file.getName() + " exists and is a directory");
                    return true;
                }
                if (Override == OVERRIDE) {
                    System.out.println("Overwriting standard input? File exists. -O or --override");
                } else {
                    throw new IOException("Program error: result = " + "Value for override: " + "is undefined");
                }
            }
            return false;
        }
        public static void main(String[] args) {
            /* determine if data is to be copied to given file
             * Take into consideration override option and
             * check if file exists and if it is a file
             * If file exists, then check if it is a file
             * If true, then check if file is to be copied to file, false if not
             */
            if (args.length > 1) {
                System.out.println("Usage: copyfile src-dir-name dest-dir-name");
                /* make sure the source file is a valid readable file
                 * if (sourceFile.exists() & !sourceFile.isDirectory())
                 *     sourceFile.isFile();
                 *     System.out.println("Not a readable file: " + sourceFile.getName());
                 */
                /* make sure the destination is a directory
                 * File destDir = new File(destDirName);
                 * if (!destDir.exists() & !destDir.isDirectory())
                 *     destDir.mkdirs();
                 */
                /* create File object for destination file
                 * if (destDir.exists() & destDir.isDirectory())
                 *     destDir.mkdirs();
                 */
                /* check if copying in desired given overwrite option
                 */
                if (args[0].equals("-O") || args[0].equals("--override")) {
                    System.out.println("Overwriting standard input? File exists. -O or --override");
                } else {
                    System.out.println("Overwriting standard input? File exists. -O or --override");
                }
                /* copy file, optionally creating a checksum
                 */
                if (args[0].equals("-P") || args[0].equals("--progress")) {
                    System.out.println("Copying file with progress bar");
                }
                /* copy timestamp of last modification
                 */
                if (args[0].equals("-T") || args[0].equals("--timestamps")) {
                    System.out.println("Copying file with timestamps");
                }
                /* optionally verify file
                 */
                if (args[0].equals("-V") || args[0].equals("--verify")) {
                    System.out.println("Verifying destination file.");
                    if (sourceFile.equals(destFile)) {
                        System.out.println("Source and destination file are equal.");
                    } else {
                        System.out.println("Error: checksums differ.");
                    }
                }
            }
            /* Print a message to standard output and read lines from
             * standard input. The user can type a line and press enter
             * to print a message informative with to be answered by user
             * for example: "What is your name?" and "Hello, " + name
             */
            public static String readInput(String message) {
                System.out.print(message);
                BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
                String answer = null;
                while ((line = in.readLine()) != null) {
                    if ("quit".equals(line)) {
                        if ("yes".equals(line)) {
                            System.out.println("Goodbye!");
                            System.out.println("Goodbye!");
                        }
                    }
                    if ("yes".equals(line)) {
                        System.out.println("Goodbye!");
                        System.out.println("Goodbye!");
                    }
                    if ("no".equals(line)) {
                        System.out.println("Goodbye!");
                        System.out.println("Goodbye!");
                    }
                }
                if (answer == null) {
                    throw new IOException("Unexpected end of input from stdin.");
                }
                in.close();
            }
            catch (IOException ioe) {
                throw new IOException("Error: read from stdin or write to stdout.");
            }
        }
    }
}
```

```
/*
 * public class FileDownload {
 *     public static void download(String address, String localFileName) {
 *         OutputStream out = null;
 *         HttpURLConnection 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 + " (" + 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]);
    }
}

public class HappyNewYear implements Runnable {
    /*
     * private static NumberFormat formatter = NumberFormat.getInstance();
     * private JFrame frame;
     * private JLabel label;
     * private JButton button;
     * private String message;
     */
    public HappyNewYear(JFrame frame, JLabel label) {
        /*
         * store argument GUI elements
         */
        this.frame = frame;
        this.label = label;
        /*
         * complete beginning of next year
         */
        Calendar cal = new GregorianCalendar();
        int nextYear = cal.get(Calendar.YEAR) + 1;
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, 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);
        /*
         * get current time
         */
        Date now = new Date();
        message = "Happy " + nextYear + "\n";
        /*
         * prepare a message
         */
        frame.setTitle(message);
        public static int determineFontSize(JFrame frame,
                                         int componentWidth, String fontName, int fontStyle,
                                         String text) {
        int fontWidth = componentWidth * 2 / text.length();
        Font font = new Font(fontName, fontStyle, fontWidth);
        FontMetrics fontMetrics = font.createFontMetrics(frame.getGraphics());
        int stringWidth = fontMetrics.stringWidth(text);
        return (int) Math.floor((componentWidth - stringWidth) / 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.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        final String fontName = "SansSerif";
        final String fontStyle = "bold";
        final int componentWidth = 300;
        frame.setSize(componentWidth, componentWidth);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Dimension screenSize = toolkit.getScreenSize();
        int fontScaleText = determineFontSize(frame, toolkit.getScreenSize().width, toolkit.getScreenSize().height);
        int fontScaleFontSize = determineFontSize(frame, toolkit.getScreenSize().width, toolkit.getScreenSize().height);
        label.setFont(new Font(fontName, fontStyle, fontScaleFontSize));
        label.setText(message);
        frame.setContentPane(label);
        frame.setVisible(true);
        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);
                }
            }
        } while (!newYear);
    }
}
```

Synchronization

UI dependency

Crosscutting Concerns

```
import java.io.*;
import java.util.zip.*;
/*
 * command line program to copy a file to another directory.
 * Author: Marco Schmid
 */
public class Copyfile {
    /* constant values for the override option
     * -O or --override
     */
    public static final int OVERRIDE = 1;
    public static final int OVERWRITE = 2;
    /* progress option
     * -P or --progress
     */
    public static final int PROGRESS = 4;
    private static boolean copyTimestamps = true;
    private static int maxTimestamps = 1000000000;
    private static long copyTimestampsDeadline;
    public static long copyTimestampsDeadline() {
        return copyTimestampsDeadline;
    }
    /* constant values for the copy option
     * -C or --copy
     */
    public static final int COPY = 8;
    public static final int COPYFILE = 16;
    /* command line options
     * -f or --force
     */
    public static final int FORCE = 32;
    /* command line options
     * -S or --size
     */
    public static final int SIZE = 64;
    /* command line options
     * -H or --hard-links
     */
    public static final int HARDLINKS = 128;
    /* command line options
     * -L or --links
     */
    public static final int LINKS = 256;
    /* command line options
     * -R or --recursive
     */
    public static final int RECURSIVE = 512;
    /* command line options
     * -T or --timestamps
     */
    public static final int timestamps = 1024;
    /* command line options
     * -M or --max-timestamps
     */
    public static final int MAX_TIMESTAMP = 2048;
    /* command line options
     * -D or --deadline
     */
    public static final int DEADLINE = 4096;
    /* command line options
     * -V or --verbose
     */
    public static final int VERBOSE = 8192;
    /* command line options
     * -o or --output
     */
    public static final int OUTPUT = 16384;
    /* command line options
     * -p or --progress
     */
    public static final int PROGRESSIVE = 32768;
    /* command line options
     * -n or --no-progress
     */
    public static final int NO_PROGRESS = 65536;
    /* command line options
     * -q or --quiet
     */
    public static final int QUIET = 131072;
    /* command line options
     * -c or --copy-size
     */
    public static final int COPY_SIZE = 262144;
    /* command line options
     * -b or --block-size
     */
    public static final int BLOCK_SIZE = 524288;
    /* command line options
     * -m or --max-block-size
     */
    public static final int MAX_BLOCK_SIZE = 1048576;
    /* command line options
     * -s or --size-only
     */
    public static final int SIZE_ONLY = 2097152;
    /* command line options
     * -x or --no-copy
     */
    public static final int NO_COPY = 4194304;
    /* command line options
     * -r or --no-recurse
     */
    public static final int NO_RECURSE = 8388608;
    /* command line options
     * -d or --no-deadline
     */
    public static final int NO_DEADLINE = 16777216;
    /* command line options
     * -e or --no-force
     */
    public static final int NO_FORCE = 33554432;
    /* command line options
     * -f or --no-force
     */
    public static final int NO_FORCE_F = 67108864;
    /* command line options
     * -n or --no-force
     */
    public static final int NO_FORCE_N = 134217728;
    /* command line options
     * -p or --no-progress
     */
    public static final int NO_PROGRESS_P = 268435456;
    /* command line options
     * -q or --no-quiet
     */
    public static final int NO_QUIET_Q = 536870912;
    /* command line options
     * -c or --no-copy-size
     */
    public static final int NO_COPY_SIZE_C = 1073741840;
    /* command line options
     * -b or --no-block-size
     */
    public static final int NO_BLOCK_SIZE_B = 2147483680;
    /* command line options
     * -m or --no-max-block-size
     */
    public static final int NO_MAX_BLOCK_SIZE_M = 4294967360;
    /* command line options
     * -s or --no-size-only
     */
    public static final int NO_SIZE_ONLY_S = 8589934720;
    /* command line options
     * -x or --no-no-copy
     */
    public static final int NO_NO_COPY_X = 17179869440;
    /* command line options
     * -r or --no-no-recurse
     */
    public static final int NO_NO_RECURSE_R = 34359738880;
    /* command line options
     * -d or --no-no-deadline
     */
    public static final int NO_NO_DEADLINE_D = 68719477760;
    /* command line options
     * -e or --no-no-force
     */
    public static final int NO_NO_FORCE_E = 137438955520;
    /* command line options
     * -f or --no-no-force-f
     */
    public static final int NO_NO_FORCE_F_F = 274877911040;
    /* command line options
     * -n or --no-no-force-n
     */
    public static final int NO_NO_FORCE_N_N = 549755822080;
    /* command line options
     * -p or --no-no-progress-p
     */
    public static final int NO_NO_PROGRESS_P_P = 1099511644160;
    /* command line options
     * -q or --no-no-quiet-q
     */
    public static final int NO_NO_QUIET_Q_Q = 2199023288320;
    /* command line options
     * -c or --no-no-copy-size-c
     */
    public static final int NO_NO_COPY_SIZE_C_C = 4398046576640;
    /* command line options
     * -b or --no-no-block-size-b
     */
    public static final int NO_NO_BLOCK_SIZE_B_B = 8796093153280;
    /* command line options
     * -m or --no-no-max-block-size-m
     */
    public static final int NO_NO_MAX_BLOCK_SIZE_M_M = 17592186306560;
    /* command line options
     * -s or --no-no-size-only-s
     */
    public static final int NO_NO_SIZE_ONLY_S_S = 35184372613120;
    /* command line options
     * -x or --no-no-no-copy-x
     */
    public static final int NO_NO_NO_COPY_X_X = 70368745226240;
    /* command line options
     * -r or --no-no-no-recurse-r
     */
    public static final int NO_NO_NO_RECURSE_R_R = 140737490452480;
    /* command line options
     * -d or --no-no-no-deadline-d
     */
    public static final int NO_NO_NO_DEADLINE_D_D = 281474980904960;
    /* command line options
     * -e or --no-no-no-force-e
     */
    public static final int NO_NO_NO_FORCE_E_E = 562949961809920;
    /* command line options
     * -f or --no-no-no-force-f-f
     */
    public static final int NO_NO_NO_FORCE_F_F_F = 1125899923619840;
    /* command line options
     * -n or --no-no-no-force-n-n
     */
    public static final int NO_NO_NO_FORCE_N_N_N = 2251799847239680;
    /* command line options
     * -p or --no-no-no-progress-p-p
     */
    public static final int NO_NO_NO_PROGRESS_P_P_P = 4503599694479360;
    /* command line options
     * -q or --no-no-no-quiet-q-q
     */
    public static final int NO_NO_NO_QUIET_Q_Q_Q = 9007199388958720;
    /* command line options
     * -c or --no-no-no-copy-size-c-c
     */
    public static final int NO_NO_NO_COPY_SIZE_C_C_C = 18014398777917440;
    /* command line options
     * -b or --no-no-no-block-size-b-b
     */
    public static final int NO_NO_NO_BLOCK_SIZE_B_B_B = 36028797555834880;
    /* command line options
     * -m or --no-no-no-max-block-size-m-m
     */
    public static final int NO_NO_NO_MAX_BLOCK_SIZE_M_M_M = 72057595111669760;
    /* command line options
     * -s or --no-no-no-size-only-s-s
     */
    public static final int NO_NO_NO_SIZE_ONLY_S_S_S = 144115190223339520;
    /* command line options
     * -x or --no-no-no-no-copy-x-x
     */
    public static final int NO_NO_NO_NO_COPY_X_X_X = 288230380446679040;
    /* command line options
     * -r or --no-no-no-no-recurse-r-r
     */
    public static final int NO_NO_NO_NO_RECURSE_R_R_R = 576460760893358080;
    /* command line options
     * -d or --no-no-no-no-deadline-d-d
     */
    public static final int NO_NO_NO_NO_DEADLINE_D_D_D = 1152921521786716160;
    /* command line options
     * -e or --no-no-no-no-force-e-e
     */
    public static final int NO_NO_NO_NO_FORCE_E_E_E = 2305843043573432320;
    /* command line options
     * -f or --no-no-no-no-force-f-f-f
     */
    public static final int NO_NO_NO_NO_FORCE_F_F_F_F = 4611686087146864640;
    /* command line options
     * -n or --no-no-no-no-force-n-n-n
     */
    public static final int NO_NO_NO_NO_FORCE_N_N_N_N = 9223372174293729280;
    /* command line options
     * -p or --no-no-no-no-progress-p-p-p
     */
    public static final int NO_NO_NO_NO_PROGRESS_P_P_P_P = 18446744348587458560;
    /* command line options
     * -q or --no-no-no-no-quiet-q-q-q
     */
    public static final int NO_NO_NO_NO_QUIET_Q_Q_Q_Q = 36893488697174917120;
    /* command line options
     * -c or --no-no-no-no-copy-size-c-c-c
     */
    public static final int NO_NO_NO_NO_COPY_SIZE_C_C_C_C = 73786977394349834240;
    /* command line options
     * -b or --no-no-no-no-block-size-b-b-b
     */
    public static final int NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B = 147573954788699668480;
    /* command line options
     * -m or --no-no-no-no-max-block-size-m-m-m
     */
    public static final int NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M = 295147909577399336960;
    /* command line options
     * -s or --no-no-no-no-size-only-s-s-s
     */
    public static final int NO_NO_NO_NO_SIZE_ONLY_S_S_S_S = 590295819154798673920;
    /* command line options
     * -x or --no-no-no-no-no-copy-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_COPY_X_X_X_X = 1180591638309597347840;
    /* command line options
     * -r or --no-no-no-no-no-recurse-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_RECURSE_R_R_R_R = 2361183276619194695680;
    /* command line options
     * -d or --no-no-no-no-no-deadline-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_DEADLINE_D_D_D_D = 4722366553238389391360;
    /* command line options
     * -e or --no-no-no-no-no-force-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_FORCE_E_E_E_E = 9444733106476778782720;
    /* command line options
     * -f or --no-no-no-no-no-force-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_FORCE_F_F_F_F_F = 18889466212953557565440;
    /* command line options
     * -n or --no-no-no-no-no-force-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_FORCE_N_N_N_N_N = 37778932425907115130880;
    /* command line options
     * -p or --no-no-no-no-no-progress-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P = 75557864851814230261760;
    /* command line options
     * -q or --no-no-no-no-no-quiet-q-q-q-q
     */
    public static final int NO_NO_NO_NO_NO_QUIET_Q_Q_Q_Q_Q = 151115729703628460523520;
    /* command line options
     * -c or --no-no-no-no-no-copy-size-c-c-c-c
     */
    public static final int NO_NO_NO_NO_NO_COPY_SIZE_C_C_C_C_C = 302231459407256921047040;
    /* command line options
     * -b or --no-no-no-no-no-block-size-b-b-b-b
     */
    public static final int NO_NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B_B = 604462918814513842094080;
    /* command line options
     * -m or --no-no-no-no-no-max-block-size-m-m-m-m
     */
    public static final int NO_NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M_M = 1208925837629027684188160;
    /* command line options
     * -s or --no-no-no-no-no-size-only-s-s-s-s
     */
    public static final int NO_NO_NO_NO_NO_SIZE_ONLY_S_S_S_S_S = 2417851675258055368376320;
    /* command line options
     * -x or --no-no-no-no-no-no-copy-x-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_NO_COPY_X_X_X_X_X = 4835703350516110736752640;
    /* command line options
     * -r or --no-no-no-no-no-no-recurse-r-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_NO_RECURSE_R_R_R_R_R = 9671406701032221473505280;
    /* command line options
     * -d or --no-no-no-no-no-no-deadline-d-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_NO_DEADLINE_D_D_D_D_D = 19342813402064442947010560;
    /* command line options
     * -e or --no-no-no-no-no-no-force-e-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_NO_FORCE_E_E_E_E_E = 38685626804128885894021120;
    /* command line options
     * -f or --no-no-no-no-no-no-force-f-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_NO_FORCE_F_F_F_F_F_F = 77371253608257771788042240;
    /* command line options
     * -n or --no-no-no-no-no-no-force-n-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_NO_FORCE_N_N_N_N_N_N = 154742507216515543576084480;
    /* command line options
     * -p or --no-no-no-no-no-no-progress-p-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P_P = 309485014432031087152168960;
    /* command line options
     * -q or --no-no-no-no-no-no-quiet-q-q-q-q-q
     */
    public static final int NO_NO_NO_NO_NO_NO_QUIET_Q_Q_Q_Q_Q_Q = 618970028864062174304337760;
    /* command line options
     * -c or --no-no-no-no-no-no-copy-size-c-c-c-c-c
     */
    public static final int NO_NO_NO_NO_NO_NO_COPY_SIZE_C_C_C_C_C_C = 123794005712818434860167520;
    /* command line options
     * -b or --no-no-no-no-no-no-block-size-b-b-b-b-b
     */
    public static final int NO_NO_NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B_B_B = 247588011425636869720335040;
    /* command line options
     * -m or --no-no-no-no-no-no-max-block-size-m-m-m-m-m
     */
    public static final int NO_NO_NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M_M_M = 495176022851273739440670080;
    /* command line options
     * -s or --no-no-no-no-no-no-size-only-s-s-s-s-s
     */
    public static final int NO_NO_NO_NO_NO_NO_SIZE_ONLY_S_S_S_S_S_S = 990352045702547478881340160;
    /* command line options
     * -x or --no-no-no-no-no-no-no-copy-x-x-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_COPY_X_X_X_X_X_X = 1980704091405094957762680320;
    /* command line options
     * -r or --no-no-no-no-no-no-no-recurse-r-r-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_RECURSE_R_R_R_R_R_R = 3961408182810189915525360640;
    /* command line options
     * -d or --no-no-no-no-no-no-no-deadline-d-d-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_DEADLINE_D_D_D_D_D_D = 7922816365620379831050720000;
    /* command line options
     * -e or --no-no-no-no-no-no-no-force-e-e-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_FORCE_E_E_E_E_E_E = 15845632731240759662101440000;
    /* command line options
     * -f or --no-no-no-no-no-no-no-force-f-f-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_FORCE_F_F_F_F_F_F_F = 31691265462481519324202880000;
    /* command line options
     * -n or --no-no-no-no-no-no-no-force-n-n-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_FORCE_N_N_N_N_N_N_N = 63382530924963038648405760000;
    /* command line options
     * -p or --no-no-no-no-no-no-no-progress-p-p-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P_P_P = 126765061849926077296811520000;
    /* command line options
     * -q or --no-no-no-no-no-no-no-quiet-q-q-q-q-q-q
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_QUIET_Q_Q_Q_Q_Q_Q_Q = 253530123699852154593623040000;
    /* command line options
     * -c or --no-no-no-no-no-no-no-copy-size-c-c-c-c-c-c
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_COPY_SIZE_C_C_C_C_C_C_C = 507060247399704313187246080000;
    /* command line options
     * -b or --no-no-no-no-no-no-no-block-size-b-b-b-b-b-b
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B_B_B_B = 1014120494799408626374492160000;
    /* command line options
     * -m or --no-no-no-no-no-no-no-max-block-size-m-m-m-m-m-m
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M_M_M_M = 2028240989598817252748984320000;
    /* command line options
     * -s or --no-no-no-no-no-no-no-size-only-s-s-s-s-s-s
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_SIZE_ONLY_S_S_S_S_S_S_S = 4056481979197634505497968640000;
    /* command line options
     * -x or --no-no-no-no-no-no-no-no-copy-x-x-x-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_COPY_X_X_X_X_X_X_X = 8112963958395269010995937280000;
    /* command line options
     * -r or --no-no-no-no-no-no-no-no-recurse-r-r-r-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_RECURSE_R_R_R_R_R_R_R = 16225927916790538021991874560000;
    /* command line options
     * -d or --no-no-no-no-no-no-no-no-deadline-d-d-d-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_DEADLINE_D_D_D_D_D_D_D = 32451855833581076043983749120000;
    /* command line options
     * -e or --no-no-no-no-no-no-no-no-force-e-e-e-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_FORCE_E_E_E_E_E_E_E = 64903711667162152087967498240000;
    /* command line options
     * -f or --no-no-no-no-no-no-no-no-force-f-f-f-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_FORCE_F_F_F_F_F_F_F_F = 129807423334324304175934996480000;
    /* command line options
     * -n or --no-no-no-no-no-no-no-no-force-n-n-n-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_FORCE_N_N_N_N_N_N_N_N = 259614846668648608351869992960000;
    /* command line options
     * -p or --no-no-no-no-no-no-no-no-progress-p-p-p-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P_P_P_P = 519229693337297216703739985920000;
    /* command line options
     * -q or --no-no-no-no-no-no-no-no-quiet-q-q-q-q-q-q-q
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_QUIET_Q_Q_Q_Q_Q_Q_Q_Q = 1038459386674594433407479971840000;
    /* command line options
     * -c or --no-no-no-no-no-no-no-no-copy-size-c-c-c-c-c-c-c
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_COPY_SIZE_C_C_C_C_C_C_C_C = 2076918773349188866814959943680000;
    /* command line options
     * -b or --no-no-no-no-no-no-no-no-block-size-b-b-b-b-b-b-b
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B_B_B_B_B = 4153837546698377733629919887360000;
    /* command line options
     * -m or --no-no-no-no-no-no-no-no-max-block-size-m-m-m-m-m-m-m
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M_M_M_M_M = 8307675093396755467259839774720000;
    /* command line options
     * -s or --no-no-no-no-no-no-no-no-size-only-s-s-s-s-s-s-s
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_SIZE_ONLY_S_S_S_S_S_S_S_S = 16615350186793510934519679549440000;
    /* command line options
     * -x or --no-no-no-no-no-no-no-no-no-copy-x-x-x-x-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_COPY_X_X_X_X_X_X_X_X = 33230700373587021869039359098880000;
    /* command line options
     * -r or --no-no-no-no-no-no-no-no-no-recurse-r-r-r-r-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_RECURSE_R_R_R_R_R_R_R_R = 66461400747174043738078718197760000;
    /* command line options
     * -d or --no-no-no-no-no-no-no-no-no-deadline-d-d-d-d-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_DEADLINE_D_D_D_D_D_D_D_D = 13292280149434808747615743639520000;
    /* command line options
     * -e or --no-no-no-no-no-no-no-no-no-force-e-e-e-e-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_E_E_E_E_E_E_E_E = 26584560298869617495231487278560000;
    /* command line options
     * -f or --no-no-no-no-no-no-no-no-no-force-f-f-f-f-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_F_F_F_F_F_F_F_F_F = 53169120597739234985462954557120000;
    /* command line options
     * -n or --no-no-no-no-no-no-no-no-no-force-n-n-n-n-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_N_N_N_N_N_N_N_N_N = 10633824119547846997092585911440000;
    /* command line options
     * -p or --no-no-no-no-no-no-no-no-no-progress-p-p-p-p-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P_P_P_P_P = 21267648238995693994185171822880000;
    /* command line options
     * -q or --no-no-no-no-no-no-no-no-no-quiet-q-q-q-q-q-q-q-q
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_QUIET_Q_Q_Q_Q_Q_Q_Q_Q_Q = 42535296477991387988370343645760000;
    /* command line options
     * -c or --no-no-no-no-no-no-no-no-no-copy-size-c-c-c-c-c-c-c-c
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_COPY_SIZE_C_C_C_C_C_C_C_C_C = 85070592955982775976740687291520000;
    /* command line options
     * -b or --no-no-no-no-no-no-no-no-no-block-size-b-b-b-b-b-b-b-b
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B_B_B_B_B_B = 170141185911965551953481374583040000;
    /* command line options
     * -m or --no-no-no-no-no-no-no-no-no-max-block-size-m-m-m-m-m-m-m-m
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M_M_M_M_M_M = 340282371823931103906962749166080000;
    /* command line options
     * -s or --no-no-no-no-no-no-no-no-no-size-only-s-s-s-s-s-s-s-s
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_SIZE_ONLY_S_S_S_S_S_S_S_S_S_S = 680564743647862207813925498332160000;
    /* command line options
     * -x or --no-no-no-no-no-no-no-no-no-no-copy-x-x-x-x-x-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_COPY_X_X_X_X_X_X_X_X_X_X = 1361129487295724415627850996664320000;
    /* command line options
     * -r or --no-no-no-no-no-no-no-no-no-no-recurse-r-r-r-r-r-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_RECURSE_R_R_R_R_R_R_R_R_R = 2722258974591448831255701993328640000;
    /* command line options
     * -d or --no-no-no-no-no-no-no-no-no-no-deadline-d-d-d-d-d-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_DEADLINE_D_D_D_D_D_D_D_D_D = 54445179491828976625114039866560000;
    /* command line options
     * -e or --no-no-no-no-no-no-no-no-no-no-force-e-e-e-e-e-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_E_E_E_E_E_E_E_E_E_E = 108890358983657953250228079733120000;
    /* command line options
     * -f or --no-no-no-no-no-no-no-no-no-no-force-f-f-f-f-f-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_F_F_F_F_F_F_F_F_F_F = 217780717967315906500456159466240000;
    /* command line options
     * -n or --no-no-no-no-no-no-no-no-no-no-force-n-n-n-n-n-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_N_N_N_N_N_N_N_N_N_N_N = 435561435934631813000912318932480000;
    /* command line options
     * -p or --no-no-no-no-no-no-no-no-no-no-progress-p-p-p-p-p-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P_P_P_P_P_P = 871122871869263626001824637864960000;
    /* command line options
     * -q or --no-no-no-no-no-no-no-no-no-no-quiet-q-q-q-q-q-q-q-q-q
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_QUIET_Q_Q_Q_Q_Q_Q_Q_Q_Q_Q = 1742245743738527252003649275729920000;
    /* command line options
     * -c or --no-no-no-no-no-no-no-no-no-no-copy-size-c-c-c-c-c-c-c-c-c
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_COPY_SIZE_C_C_C_C_C_C_C_C_C_C = 3484491487477054504007298551459840000;
    /* command line options
     * -b or --no-no-no-no-no-no-no-no-no-no-block-size-b-b-b-b-b-b-b-b-b
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B_B_B_B_B_B_B = 6968982954954109008014597102919680000;
    /* command line options
     * -m or --no-no-no-no-no-no-no-no-no-no-max-block-size-m-m-m-m-m-m-m-m-m
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M_M_M_M_M_M_M = 13937965909908218016029194205839360000;
    /* command line options
     * -s or --no-no-no-no-no-no-no-no-no-no-size-only-s-s-s-s-s-s-s-s-s
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_SIZE_ONLY_S_S_S_S_S_S_S_S_S_S_S_S = 27875931819816436032058388411678720000;
    /* command line options
     * -x or --no-no-no-no-no-no-no-no-no-no-no-copy-x-x-x-x-x-x-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_COPY_X_X_X_X_X_X_X_X_X_X_X_X = 5575186363963287206411677782335760000;
    /* command line options
     * -r or --no-no-no-no-no-no-no-no-no-no-no-recurse-r-r-r-r-r-r-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_RECURSE_R_R_R_R_R_R_R_R_R_R = 11150372727926574412823355564671520000;
    /* command line options
     * -d or --no-no-no-no-no-no-no-no-no-no-no-deadline-d-d-d-d-d-d-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_DEADLINE_D_D_D_D_D_D_D_D_D_D = 2230074545585314882564671112934320000;
    /* command line options
     * -e or --no-no-no-no-no-no-no-no-no-no-no-force-e-e-e-e-e-e-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_E_E_E_E_E_E_E_E_E_E_E_E = 4460149091170629765129342225868640000;
    /* command line options
     * -f or --no-no-no-no-no-no-no-no-no-no-no-force-f-f-f-f-f-f-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_F_F_F_F_F_F_F_F_F_F_F = 8920298182341259530258684451737280000;
    /* command line options
     * -n or --no-no-no-no-no-no-no-no-no-no-no-force-n-n-n-n-n-n-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_N_N_N_N_N_N_N_N_N_N_N_N_N = 17840596364682518604517368903544640000;
    /* command line options
     * -p or --no-no-no-no-no-no-no-no-no-no-no-progress-p-p-p-p-p-p-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P_P_P_P_P_P_P = 3568119272936503720903473780708960000;
    /* command line options
     * -q or --no-no-no-no-no-no-no-no-no-no-no-quiet-q-q-q-q-q-q-q-q-q-q
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_QUIET_Q_Q_Q_Q_Q_Q_Q_Q_Q_Q_Q_Q = 713623854587300744040694796142480000;
    /* command line options
     * -c or --no-no-no-no-no-no-no-no-no-no-no-copy-size-c-c-c-c-c-c-c-c-c-c
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_COPY_SIZE_C_C_C_C_C_C_C_C_C_C_C_C = 142724770917450148810138959233120000;
    /* command line options
     * -b or --no-no-no-no-no-no-no-no-no-no-no-block-size-b-b-b-b-b-b-b-b-b-b
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_BLOCK_SIZE_B_B_B_B_B_B_B_B_B_B_B_B = 285449541834900297620277918466240000;
    /* command line options
     * -m or --no-no-no-no-no-no-no-no-no-no-no-max-block-size-m-m-m-m-m-m-m-m-m-m
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_MAX_BLOCK_SIZE_M_M_M_M_M_M_M_M_M_M_M_M = 570898983669800595240555836932480000;
    /* command line options
     * -s or --no-no-no-no-no-no-no-no-no-no-no-size-only-s-s-s-s-s-s-s-s-s-s
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_SIZE_ONLY_S_S_S_S_S_S_S_S_S_S_S_S_S_S = 1141797967339601190481111673864960000;
    /* command line options
     * -x or --no-no-no-no-no-no-no-no-no-no-no-no-copy-x-x-x-x-x-x-x-x-x-x
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_COPY_X_X_X_X_X_X_X_X_X_X_X_X_X_X = 2283595934679202380962223347729920000;
    /* command line options
     * -r or --no-no-no-no-no-no-no-no-no-no-no-no-recurse-r-r-r-r-r-r-r-r-r-r
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_RECURSE_R_R_R_R_R_R_R_R_R_R_R_R = 4567191869358404761924446695459840000;
    /* command line options
     * -d or --no-no-no-no-no-no-no-no-no-no-no-no-deadline-d-d-d-d-d-d-d-d-d-d
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_DEADLINE_D_D_D_D_D_D_D_D_D_D_D_D = 9134383738716809523848893390919680000;
    /* command line options
     * -e or --no-no-no-no-no-no-no-no-no-no-no-no-force-e-e-e-e-e-e-e-e-e-e
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_E_E_E_E_E_E_E_E_E_E_E_E_E_E = 1826876747743361904769776795183920000;
    /* command line options
     * -f or --no-no-no-no-no-no-no-no-no-no-no-no-force-f-f-f-f-f-f-f-f-f-f-f
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_F_F_F_F_F_F_F_F_F_F_F_F = 3653753495486723809539553590367840000;
    /* command line options
     * -n or --no-no-no-no-no-no-no-no-no-no-no-no-force-n-n-n-n-n-n-n-n-n-n-n
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_FORCE_N_N_N_N_N_N_N_N_N_N_N_N_N_N_N = 7307506985973447618798867180735680000;
    /* command line options
     * -p or --no-no-no-no-no-no-no-no-no-no-no-no-progress-p-p-p-p-p-p-p-p-p-p-p
     */
    public static final int NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_NO_PROGRESS_P_P_P_P_P_P_P_P_P_P_P_P = 1461501397194689523759774036147120000;
    /* command line options
     * -q or --no-no-no-no-no-no
```

AOP

```

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

/* Command line program to copy a file to another directory.
 * Usage: copyfile [options] sourcefile destfile
 */
public class copyfile {
    /* constant values for the recursive option
     * public static final int OVERWRITE_ASCII = 1;
     * public static final int OVERWRITE_BINARY = 2;
     * public static final int COPY_BINARY = 3;
     */
    // static options initialized to default values
    private static int overwrite = 0; // default
    private static int copyBinary = 0; // default
    public static long copyFile(File sourcefile, File destfile)
        throws IOException, FileNotFoundException {
        byte[] buffer = new byte[1024];
        int bytesread;
        while ((bytesread = in.read(buffer)) >= 0) {
            if (bytesread < 0) {
                checksum.update(buffer, 0, bytesread);
            } else {
                out.write(buffer, 0, bytesread);
            }
        }
        return new Long(checksum.getValue());
    }
    public static long checksum(File file) throws IOException {
        long milli = System.currentTimeMillis();
        byte[] buffer = new byte[1024];
        int bytesread;
        while ((bytesread = in.read(buffer)) >= 0) {
            if (bytesread < 0) {
                checksum.update(buffer, 0, bytesread);
            } else {
                out.write(buffer, 0, bytesread);
            }
        }
        in.close();
        milli = System.currentTimeMillis() - milli;
        System.out.println("seconds=" + milli/1000);
        return new Long(checksum.getValue());
    }
    /*
     * determine if data is to be copied to given file.
     * If source file is object for potential destination file
     * then we must copy it to a temporary file and then copy it to
     * public static boolean copyFile(File file, String destName) {
        if (destName.equals(file.getName())) {
            if (overwrite == OVERWRITE_ASCII || !exists) {
                if (!exists) {
                    throw new InternalError("Program error: invalid "
                        + "value for overwrite - " + overwrite);
                }
            }
        }
        public static void download(String address, String localFileName) {
            try {
                URL url = new URL(address);
                URLConnection conn = url.openConnection();
                InputStream in = null;
                OutputStream out = null;
                byte[] buffer = new byte[1024];
                int bytesread;
                long numWritten = 0;
                while ((bytesread = in.read(buffer)) >= 0) {
                    if (bytesread < 0) {
                        out.write(buffer, 0, bytesread);
                        numWritten += bytesread;
                    } else {
                        in.read(buffer);
                        bytesread = new Long(buffer);
                    }
                }
            } catch (IOException ioe) {
                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 Timer timer;
    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);
        message = "Happy New Year " + cal.getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
    public static int determineFontSize(JFrame frame,
        int componentWidth, String fontName, int fontStyle,
        String text) {
        return (int)(fontStyle * 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.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    label = new JLabel(" ");
    label.setBackground(Color.BLACK);
    label.setForeground(Color.WHITE);
    label.setOpaque(true);
    label.setHorizontalAlign(SwingConstants.CENTER);
    frame.getContentPane().add(label);
    GraphicsEnvironment.getLocalGraphicsEnvironment();
    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 class Filedownload {
 *     public static void download(String address, String localFileName) {
 *         OutputStream out = null;
 *         URL url = new URL(address);
 *         InputStream in = null;
 *         try {
 *             in = conn.getInputStream();
 *             byte[] buffer = new byte[1024];
 *             int bytesRead;
 *             long numWritten = 0;
 *             while ((bytesRead = in.read(buffer)) >= 0) {
 *                 if (bytesRead < 0) {
 *                     out.write(buffer, 0, numWritten);
 *                     numWritten += bytesRead;
 *                 } else {
 *                     in.read(buffer);
 *                     bytesRead = new Long(buffer);
 *                 }
 *             }
 *         } finally {
 *             try {
 *                 if (in != null)
 *                     in.close();
 *                 if (out != null)
 *                     out.close();
 *             } catch (IOException ioe) {
 *             }
 *         }
 *     }
 * }
 */

public static long checksum(File file) throws IOException {
    long milli = System.currentTimeMillis();
    byte[] buffer = new byte[1024];
    int bytesread;
    long numWritten = 0;
    while ((bytesread = in.read(buffer)) >= 0) {
        if (bytesread < 0) {
            checksum.update(buffer, 0, bytesread);
        } else {
            out.write(buffer, 0, bytesread);
            numWritten += bytesread;
        }
    }
    in.close();
    milli = System.currentTimeMillis() - milli;
    System.out.println("seconds=" + milli/1000);
    return new Long(checksum.getValue());
}

```

```

public static long checksum(File file) throws IOException {
    long milli = System.currentTimeMillis();
    byte[] buffer = new byte[1024];
    int bytesread;
    long numWritten = 0;
    while ((bytesread = in.read(buffer)) >= 0) {
        if (bytesread < 0) {
            checksum.update(buffer, 0, bytesread);
        } else {
            out.write(buffer, 0, bytesread);
            numWritten += bytesread;
        }
    }
    in.close();
    if (checksum != null) {
        milli = System.currentTimeMillis() - milli;
        System.out.println("seconds=" + milli/1000);
    }
    return new Long(checksum.getValue());
}

```

```

public static long checksum(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);
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
}

```

Modularized

Pointcuts

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

```
import java.io.*;
import java.util.*;
/* Command line program to copy a file to another directory.
 * Author: Marco Schmid.
 */
public class Copyfile {
    /* computed values for the override option
     * static final int OVERRIDE_WRITE = 1;
     * static final int OVERRIDE_READ = 2;
     * static final int OVERRIDE_BOTH = 3;
     * program options initialized to default values
     * static final int OVERRIDE_NONE = 0;
     * private static boolean strict = true;
     */
    public static long copy(String source, String destination) {
        try {
            byte[] buffer = new byte[1024];
            while ((numRead = in.read(buffer)) >= 0) {
                checksum.update(buffer, 0, numRead);
            }
            return new Long(checksum.getvalue());
        } catch (IOException ioe) {
            return null;
        }
    }
    public static long checksumOfFile(File sourceFile) {
        long millis = System.currentTimeMillis();
        FileInputStream in = null;
        byte[] buffer = new byte[1024];
        URLConnection conn = null;
        Inputstream in = null;
        try {
            in = conn.getInputStream();
            byte[] buffer = new byte[1024];
            int numWritten = 0;
            while ((numRead = in.read(buffer)) >= 0) {
                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 downloadString(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]);
        }
    }
}
// Assume if data is to be copied to given file.
// Return file Path or null for generic destination file
// Returns true if data is to be copied to file, false if not
public static boolean isFileOrDir(String path) {
    File file = new File(path);
    if (file.exists() && file.isFile()) {
        return true;
    } else if (file.exists() && file.isDirectory()) {
        return true;
    } else {
        throw new InternalError("Program error. Invalid " +
                               "value for override: " + override);
    }
}
public static void main(String[] args) throws IOException {
    // parse command line arguments
    if (args.length > 2) {
        System.out.println("Usage: Copyfile SRC-FILE-NAME DEST-FILE-NAME");
    }
    // make sure the source file exists and is readable file
    File source = new File(args[0]);
    if (!source.exists() || !source.isFile() || !source.isReadable()) {
        System.exit(1);
    }
    // make sure the second argument is a directory
    if (!args[1].exists() || !args[1].isDirectory()) {
        System.out.println("Not a directory: " + args[1].getAbsolutePath());
    }
    // create file object for destination file
    File dest = new File(args[1] + File.separator + source.getName());
    if (!dest.exists() || !dest.isWritable()) {
        System.exit(1);
    }
    // check if copying is desired given override option
    if (override != null) {
        return;
    }
    // copy timestamp of last modification
}
// Forces user answer, true for yes, false for no.
public static boolean askForAnswer(String message) {
    System.out.print(message);
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    String answer = null;
    try {
        while (true) {
            answer = reader.readLine();
            if ("y".equals(answer) || "Y".equals(answer)) {
                break;
            } else if ("n".equals(answer) || "N".equals(answer)) {
                break;
            } else {
                System.out.println("Could not understand answer ('" +
                                   answer + "') . Please type y for yes or n for no.");
            }
        }
    } catch (IOException ioe) {
    }
}
}

```

```
/*
 * public class FileDownload {
 *     public static void downloadString(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 numWritten = 0;
 *             while ((numRead = in.read(buffer)) >= 0) {
 *                 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 downloadStringIndex(int index, 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.getTimeInMillis();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
    public void run() {
        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.getTimeInMillis();
        // prepare a message
        message = "Happy " + nextYear + "!";
        public static int determineFontSize(JFrame frame,
                                           int componentWidth, String fontName,
                                           int fontStyle,
                                           String text) {
        return (int)(fontStyle * 0.35 *
                  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(message);
        label.setFont(new Font("Times New Roman", Font.PLAIN, 20));
        label.setForeground(Color.WHITE);
        label.setOpaque(true);
        label.setHorizontalAlignment(SwingConstants.CENTER);
        frame.getContentPane().add(label);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getDesktopScreenDevice().setFullScreenWindow(frame);
        final int fontStyle = Font.PLAIN;
        label.setFont(new Font(fontName, fontStyle,
                             Math.min(componentWidth, fontName.length())));
        new HappyNewYear(frame, label).run();
    }
    public void run() {
        boolean newYear = false;
        do {
            newYear = true;
            output = message;
        } while (true);
        else {
            // make a String from the number of seconds
            output = formatter.format(remaining);
            label.setText(output);
            if (label.getText().length() > 10) {
                Thread.sleep(1000);
            }
        }
    }
}

```

```
public static long createTimeStampFromFile(File file, Throwable exception) {
    long millis = System.currentTimeMillis();
    if (exception != null) {
        millis -= exception.getStackTrace().length * 4000;
    }
    byte[] buffer = new byte[1024];
    int numWritten = 0;
    while ((bytread = in.read(buffer)) >= 0) {
        numWritten += bytread;
    }
    in.close();
    if (closed) {
        millis = System.currentTimeMillis() - millis;
    }
    System.out.println("second=" + (millis / 1000));
    return new Long(checksum.getvalue());
}
}

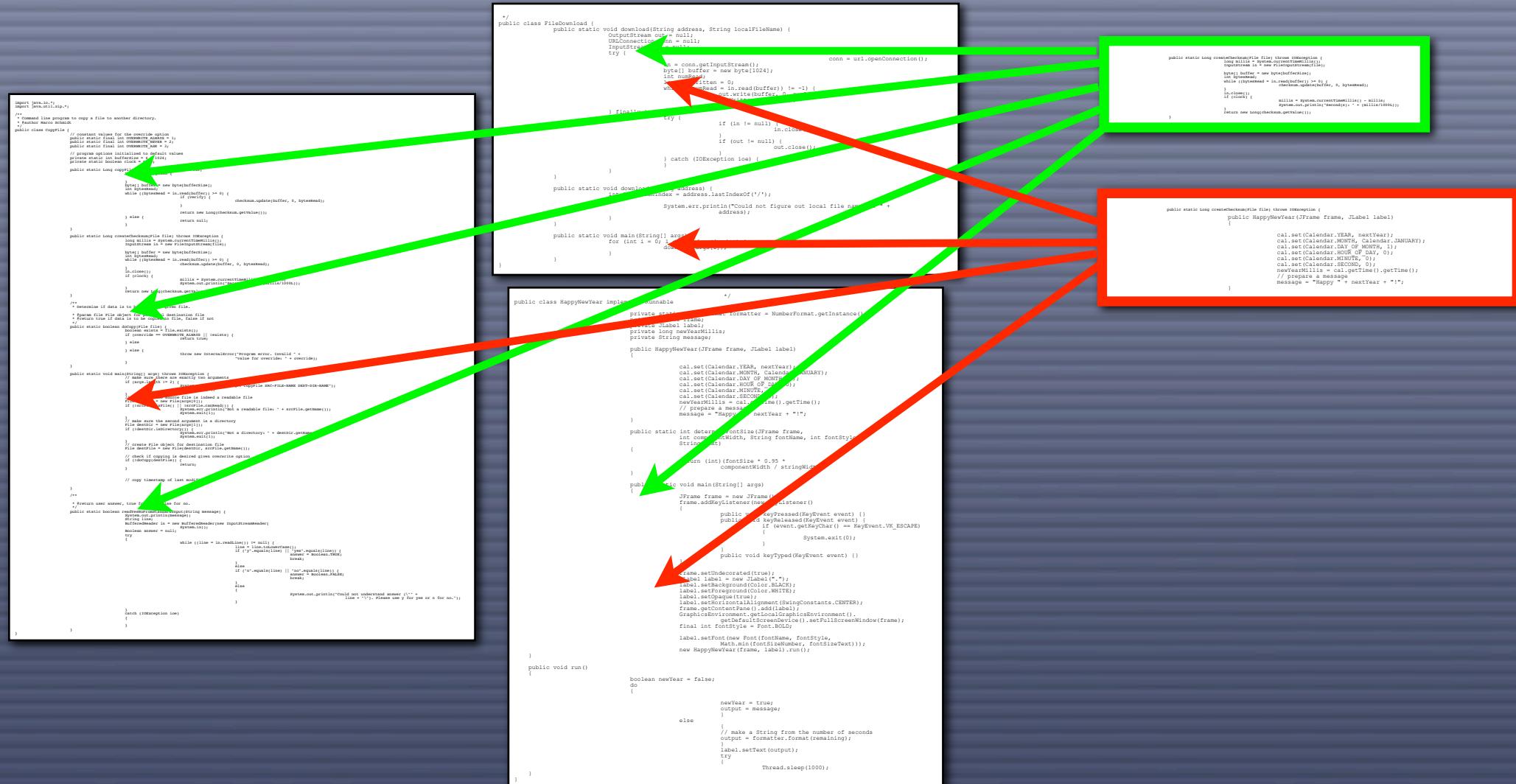
```

```
public static long createTimeStampFromFile() 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.getTimeInMillis();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
    public void run() {
        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.getTimeInMillis();
        // prepare a message
        message = "Happy " + nextYear + "!";
        public static int determineFontSize(JFrame frame,
                                           int componentWidth, String fontName,
                                           int fontStyle,
                                           String text) {
        return (int)(fontStyle * 0.35 *
                  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(message);
        label.setFont(new Font("Times New Roman", Font.PLAIN, 20));
        label.setForeground(Color.WHITE);
        label.setOpaque(true);
        label.setHorizontalAlignment(SwingConstants.CENTER);
        frame.getContentPane().add(label);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getDesktopScreenDevice().setFullScreenWindow(frame);
        final int fontStyle = Font.PLAIN;
        label.setFont(new Font(fontName, fontStyle,
                             Math.min(componentWidth, fontName.length())));
        new HappyNewYear(frame, label).run();
    }
    public void run() {
        boolean newYear = false;
        do {
            newYear = true;
            output = message;
        } while (true);
        else {
            // make a String from the number of seconds
            output = formatter.format(remaining);
            label.setText(output);
            if (label.getText().length() > 10) {
                Thread.sleep(1000);
            }
        }
    }
}

```

Pointcuts

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



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

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

Pointcut

Advice

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

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

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

AsCARMABeforeAfterAdvice

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

Pointcut

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

AsCARMABeforeAfterAdvice

```
qualifier:...
pointcutQuery: 'reception(?jp, #eventDoubleClick
    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

Person>>name

^name

Person>>name: anObject

name := anObject

- 1.accessing protocol
- 2.selector corresponds to variable name

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

~~Person>>name~~

~~^name~~

~~Person>>name: anObject~~

~~name := anObject~~

- 1.accessing protocol
- 2.selector corresponds to variable name

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]

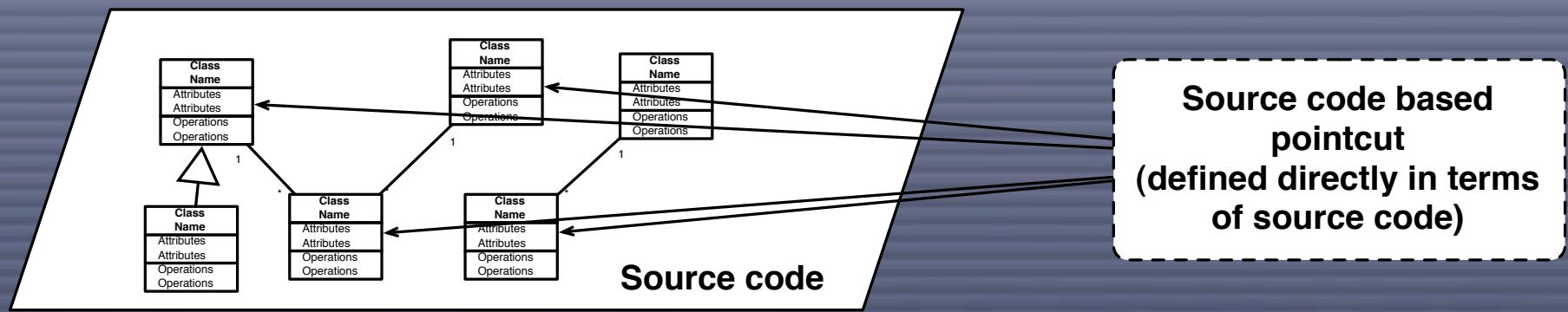
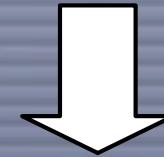
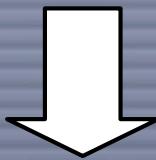
Synchronization

```
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

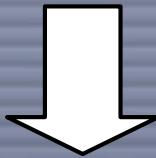
Base program developer

Aspect developer

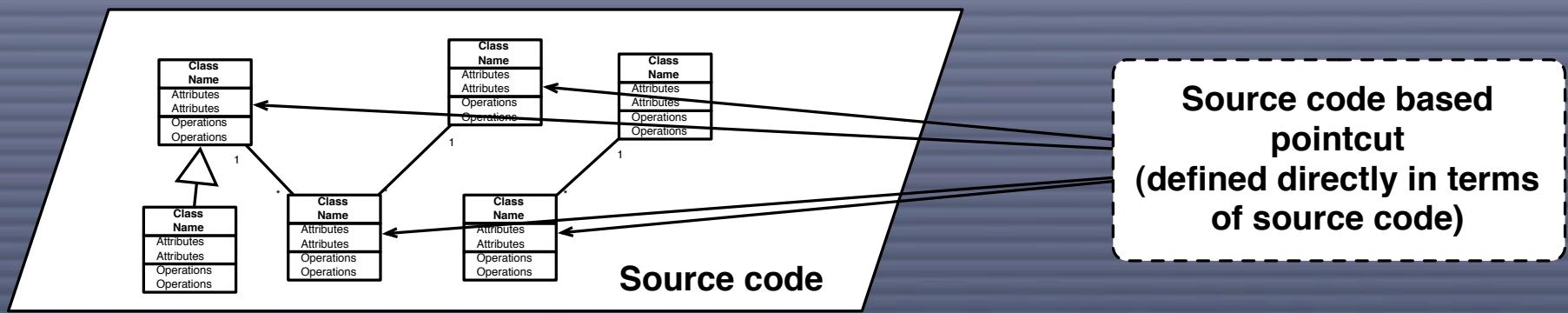
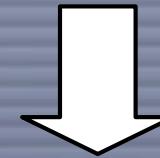


Problem Analysis

Base program developer



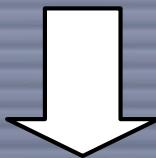
Aspect developer



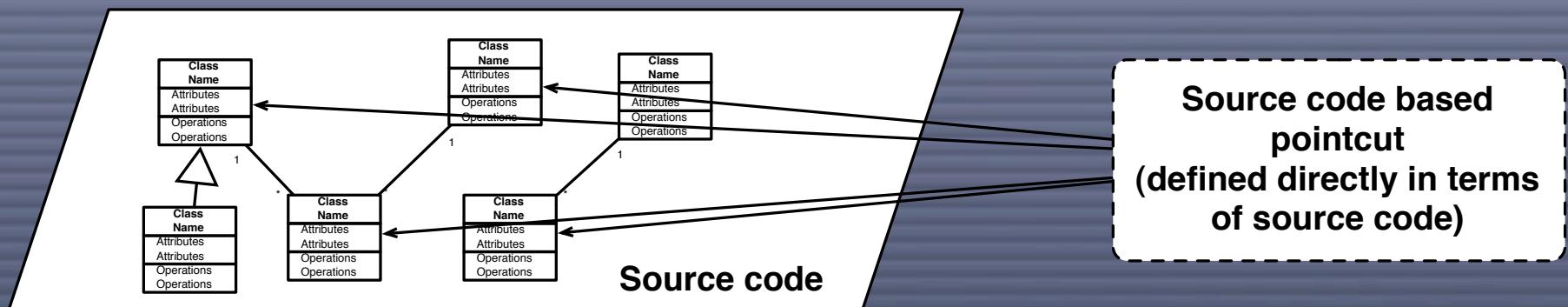
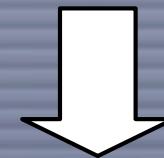
Tight
coupling

Problem Analysis

Base program developer



Aspect developer

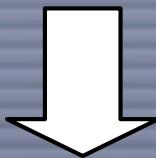


Tight
coupling

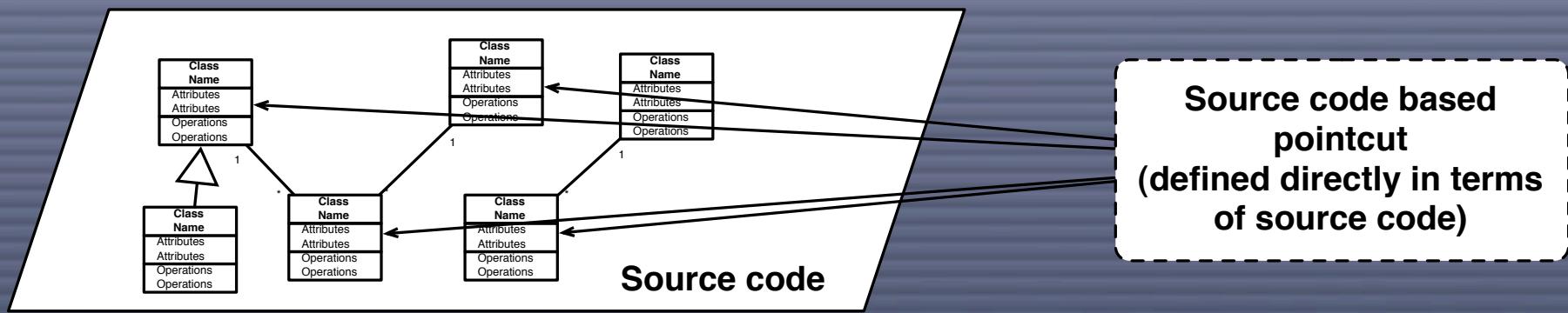
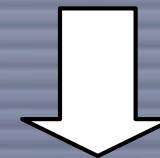
Complex

Problem Analysis

Base program developer



Aspect developer



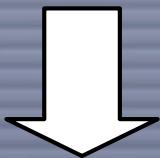
Tight
coupling

Complex

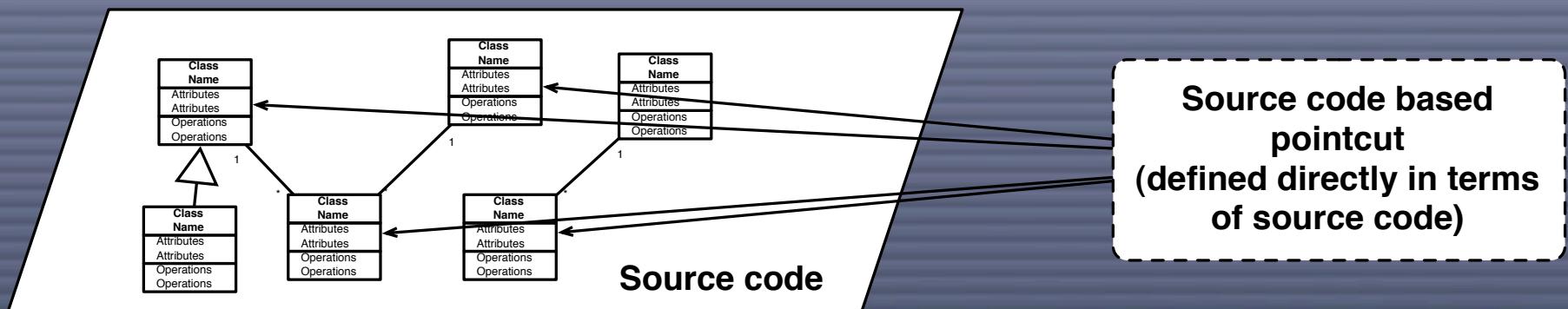
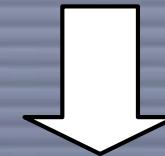
Fragile

Application-specific

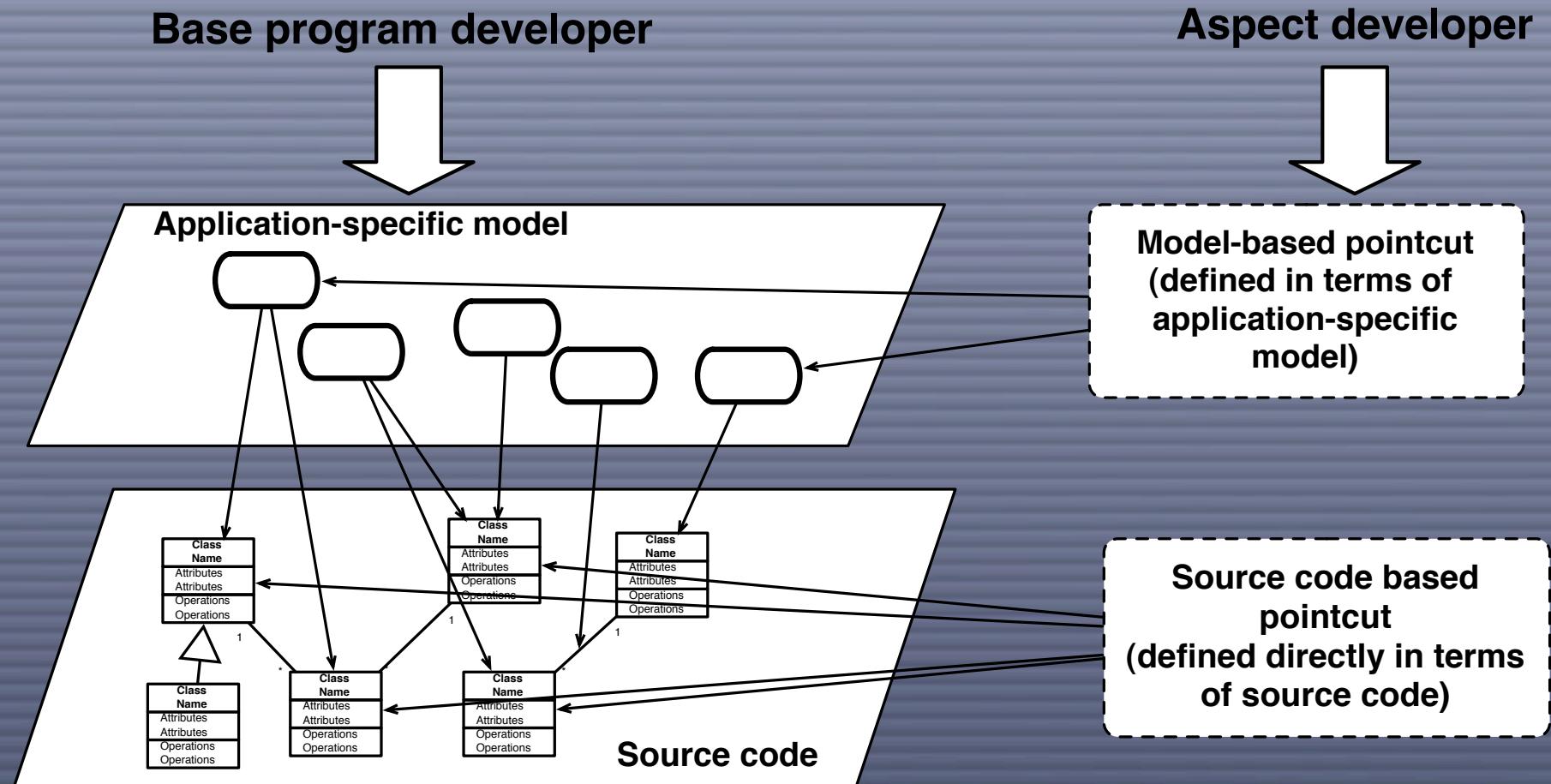
Base program developer



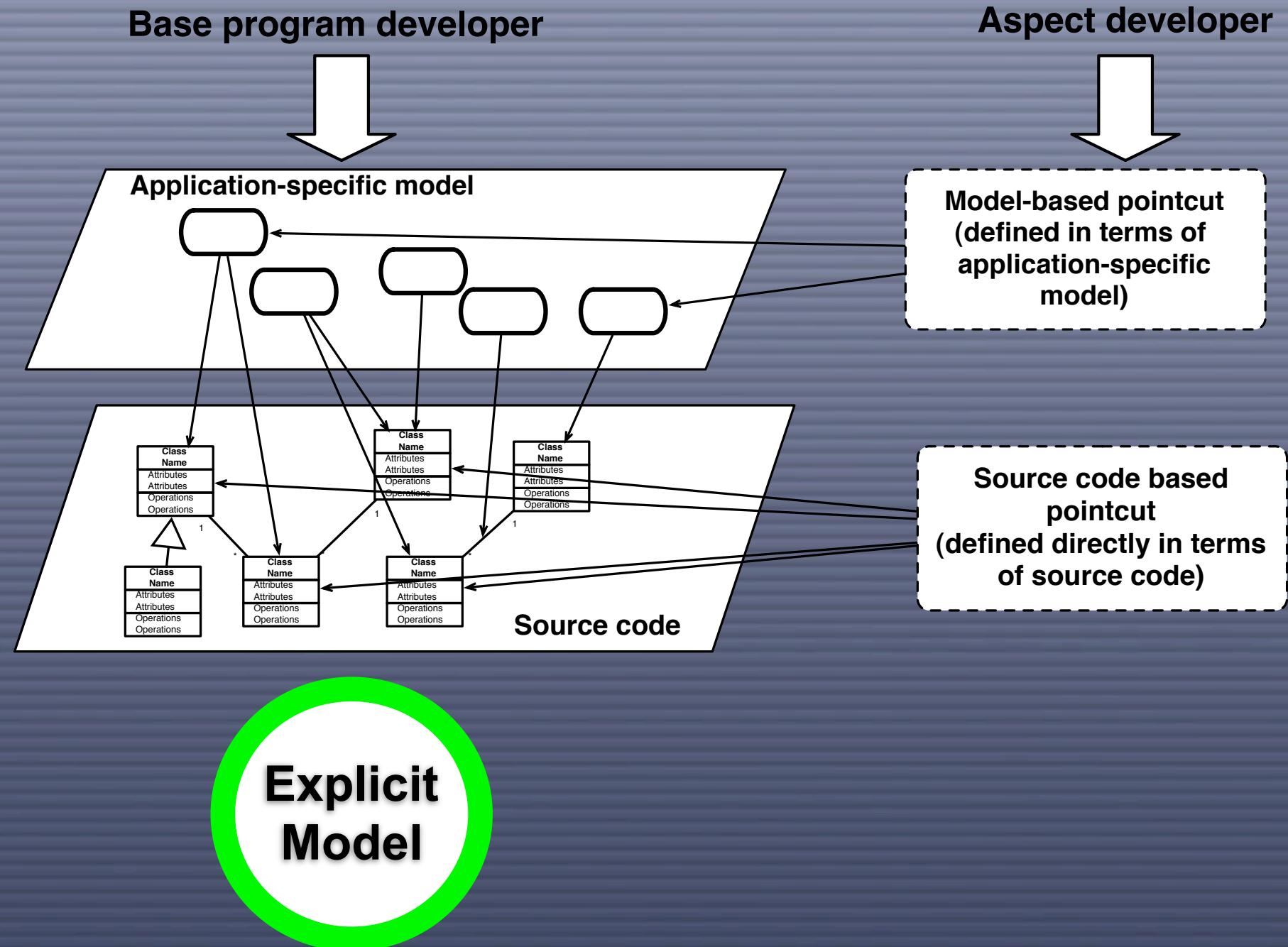
Aspect developer



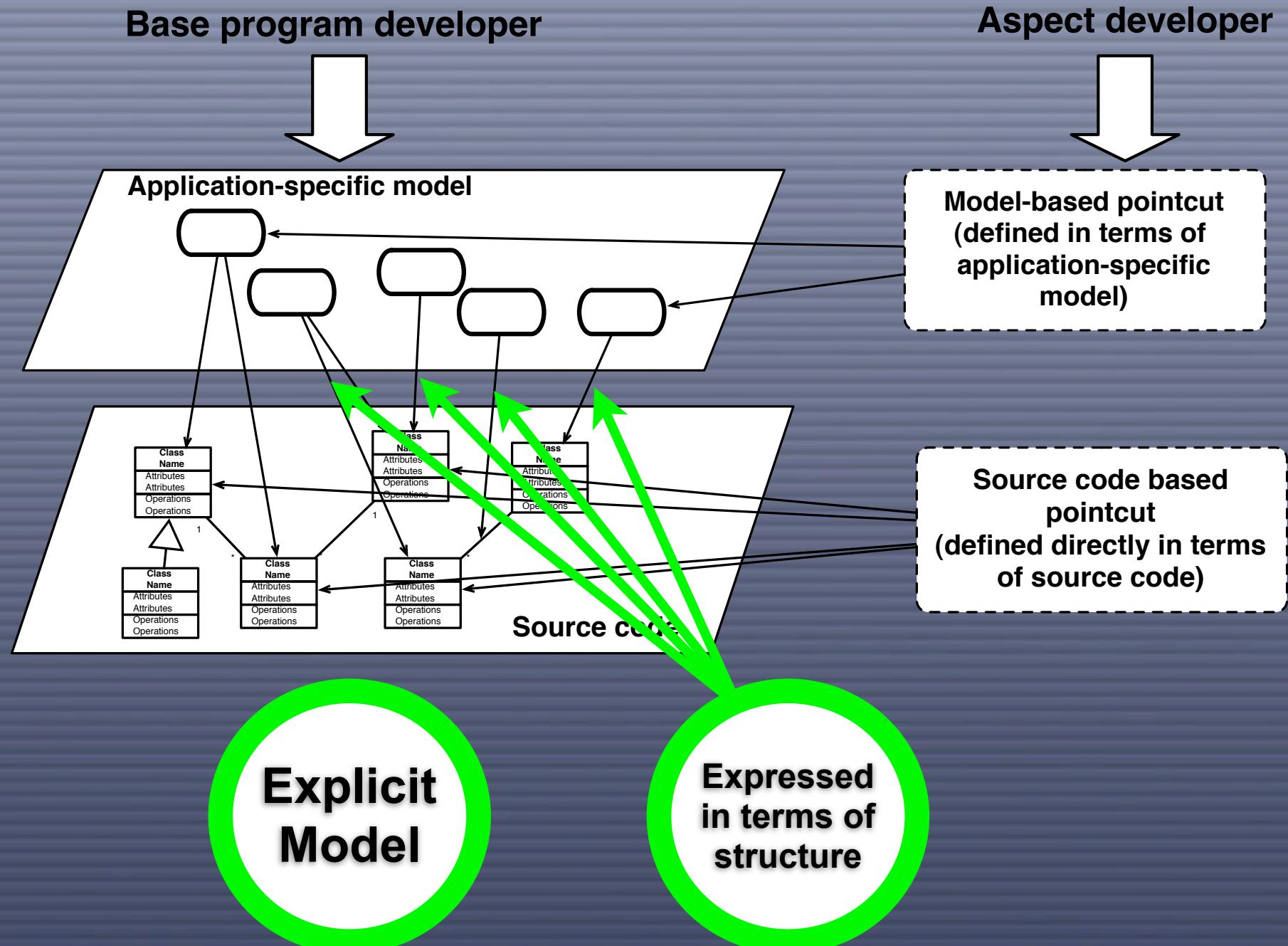
Application-specific



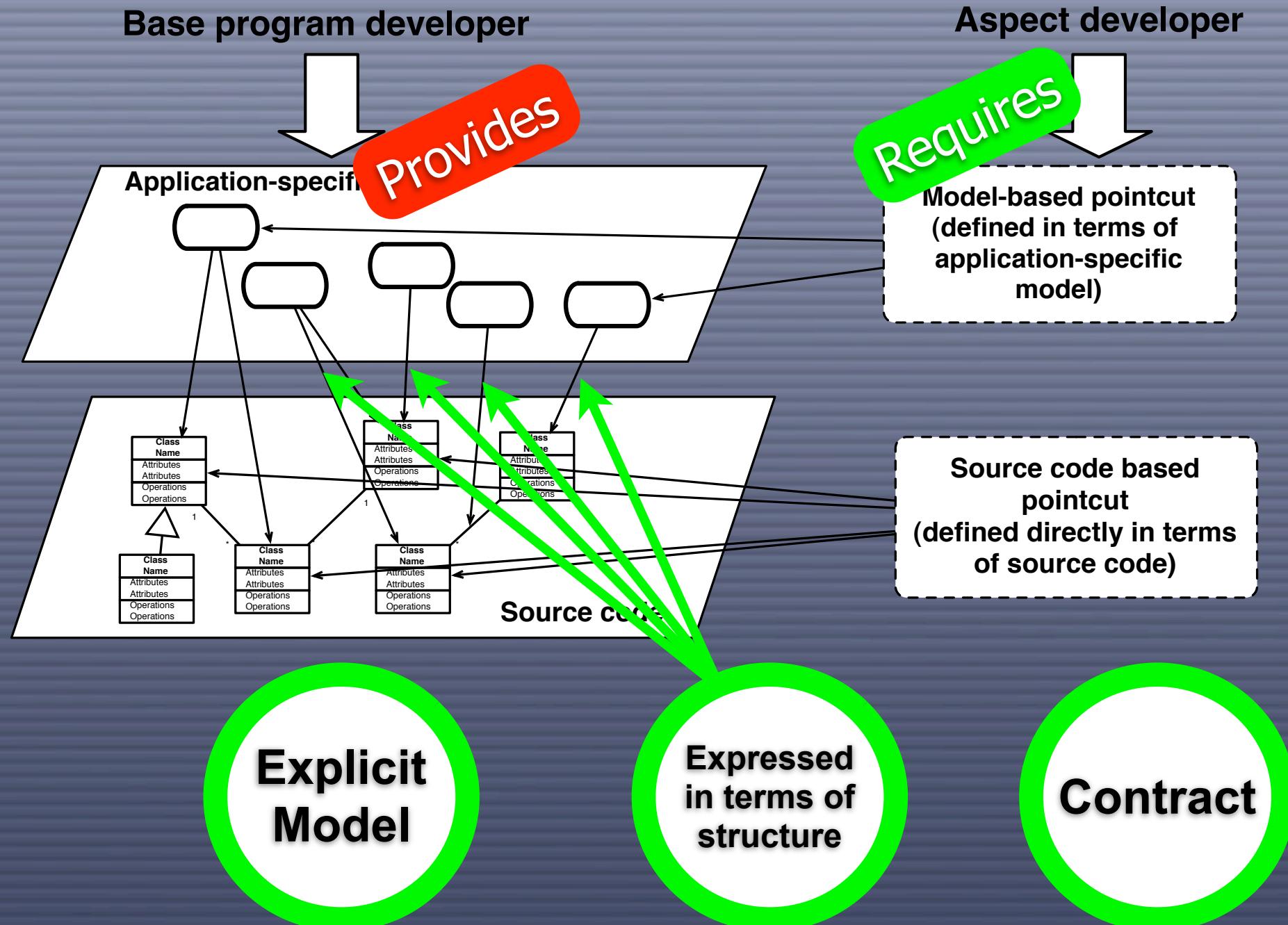
Application-specific



Application-specific



Application-specific



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
```

Pointcut

Synchronization

```
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)

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)

Model

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)
```

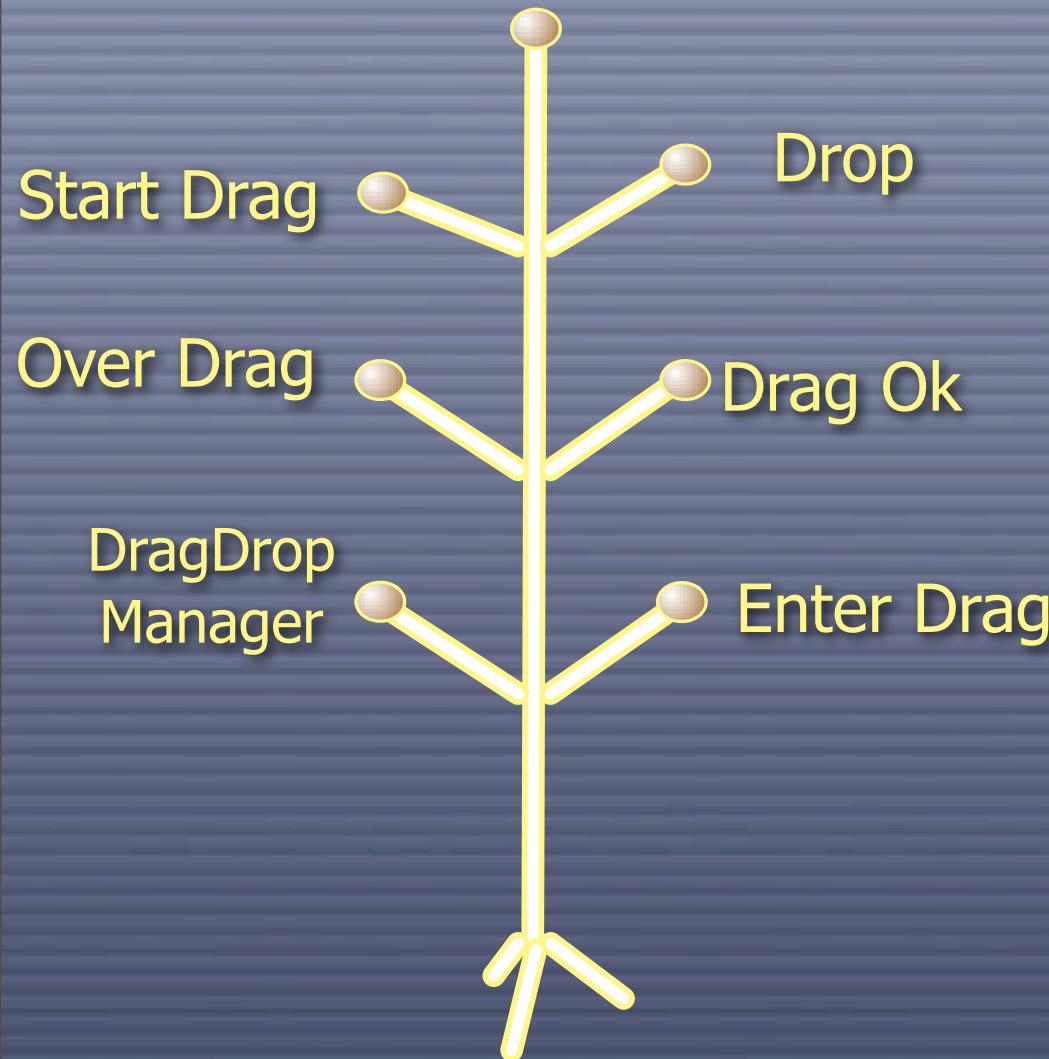
Synchronization

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

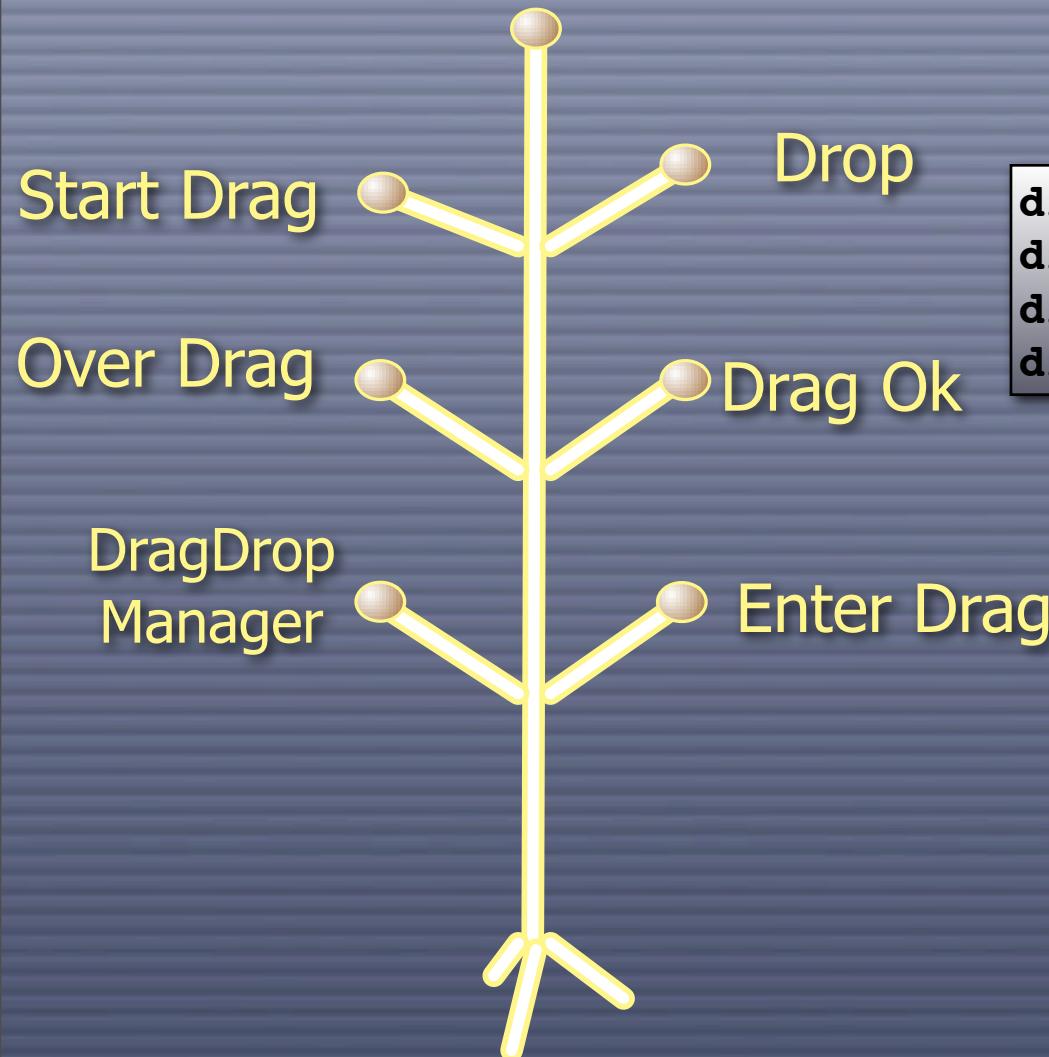
Synchronization

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

Drag & Drop



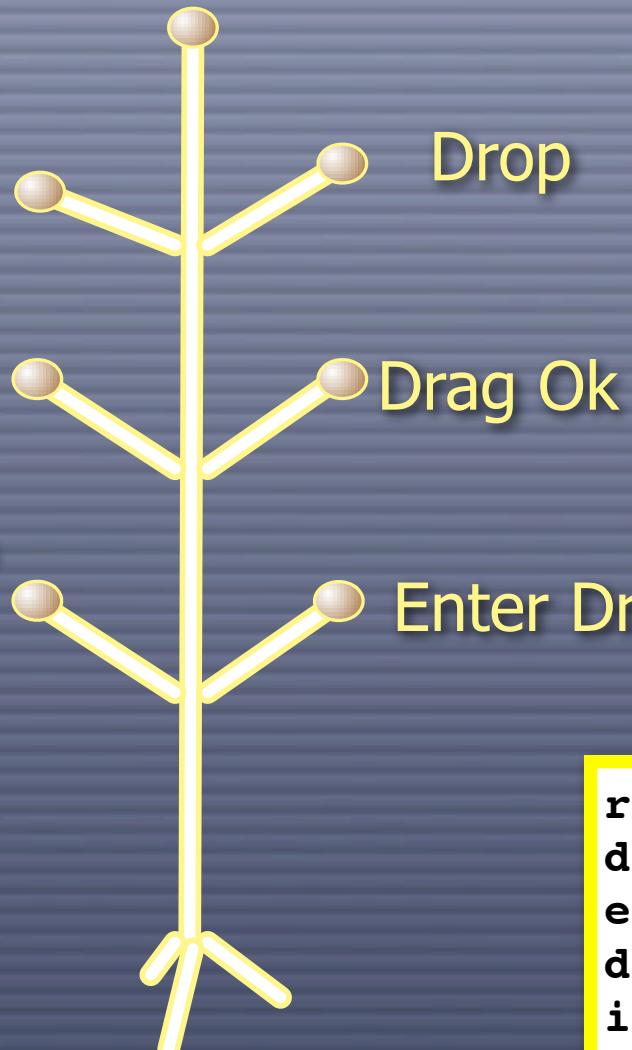
Drag & Drop



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

Drag & Drop

Start Drag



Drop

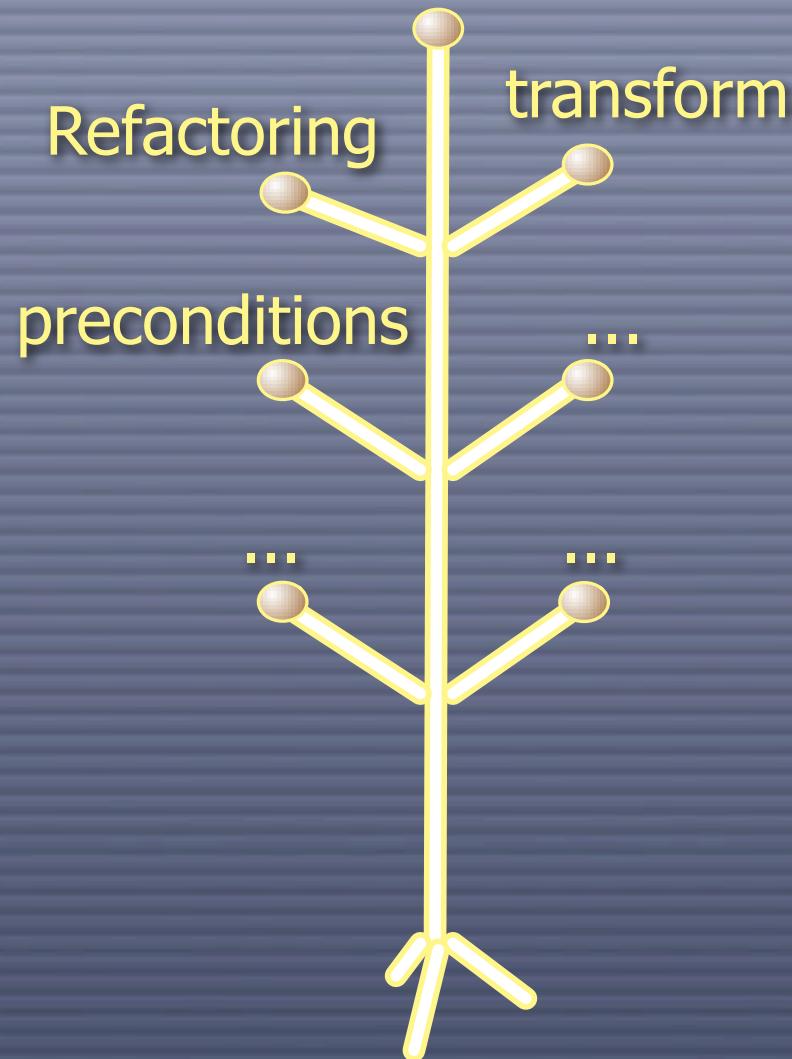
Drag Ok

Enter Drag

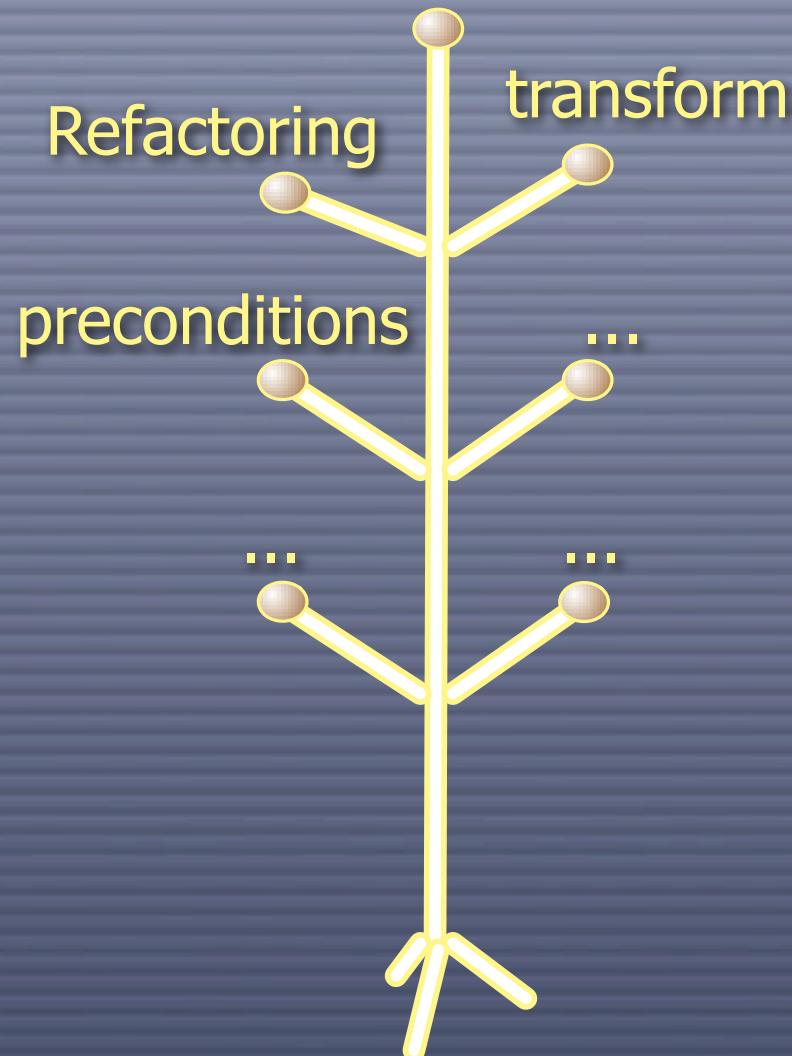
```
dragOkMethod(?class,?sel,?component)  
dragEnterMethod(?class,?sel,?component)  
dragSource(?dragdropmanager,?source)  
draggedObject(?dragdropmanager,?object)
```

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

Refactoring

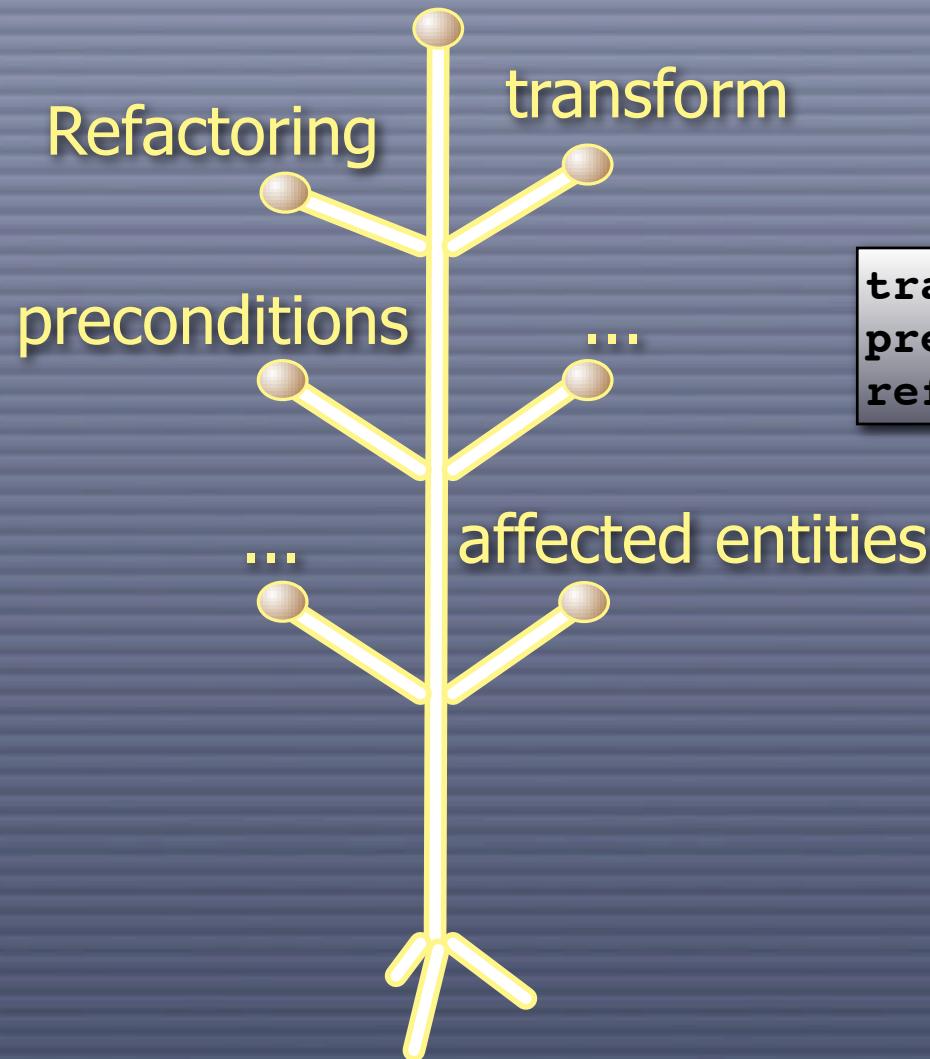


Refactoring



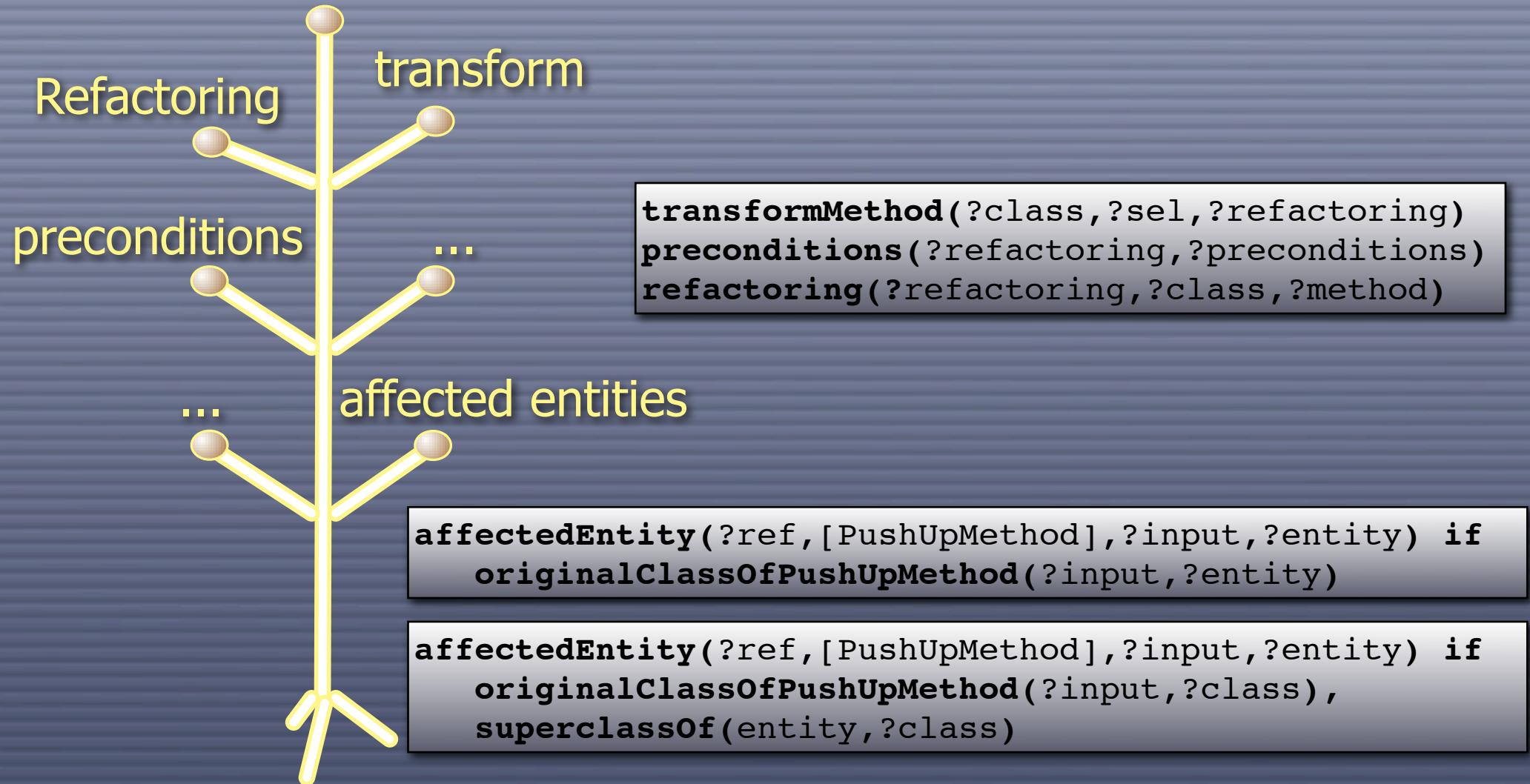
```
transformMethod(?class,?sel,?refactoring)
preconditions(?refactoring,?preconditions)
refactoring(?refactoring,?class,?method)
```

Refactoring



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

Refactoring



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

UifL



HPI



More information:

<http://prog.vub.ac.be/carma/>