

## Exception:

The screenshot shows two instances of the Eclipse IDE interface, each displaying a Java code editor and various toolbars and panes.

**Top Window (APIException.java):**

- Project Explorer:** Shows the package structure under com.ebay.mauli.exception. The APIException class is selected.
- Code Editor:** Displays the source code for APIException.java. The code defines a class that extends RuntimeException. It includes constructors for String, Throwable, int, and String/Throwable, along with a getErrorCode() method.
- Outline View:** Shows the class hierarchy for APIException.
- Call Hierarchy:** Shows methods calling the constructor of WebPageSection.

```

1 package com.ebay.mauli.exception;
2
3 public class APIException extends RuntimeException {
4     private static final long serialVersionUID = -7173883151125380826L;
5     private int errorCode;
6
7     public APIException() {
8    }
9
10    public APIException(String message) {
11        super(message);
12    }
13
14    public APIException(Throwable cause) {
15        super(cause);
16    }
17
18    public APIException(String message, Throwable cause) {
19        super(message, cause);
20    }
21
22    public APIException(int errorCode, String message, Throwable cause) {
23        super(message, cause);
24        this.errorCode = errorCode;
25    }
26
27    public int getErrorCode() {
28        return this.errorCode;
29    }
30
31
32
33 }
34

```

**Bottom Window (MauiException.java):**

- Project Explorer:** Shows the package structure under com.ebay.mauli.exception. The MauiException class is selected.
- Code Editor:** Displays the source code for MauiException.java. The code defines a class that extends Exception. It includes constructors for String, Throwable, and String/Throwable.
- Outline View:** Shows the class hierarchy for MauiException.
- Call Hierarchy:** Shows methods calling the constructor of WebPageSection.

```

1 package com.ebay.mauli.exception;
2
3 public class MauiException extends Exception {
4     private static final long serialVersionUID = -2757992149932989353L;
5
6     public MauiException() {
7    }
8
9     public MauiException(String message) {
10        super(message);
11    }
12
13     public MauiException(Throwable cause) {
14        super(cause);
15    }
16
17     public MauiException(String message, Throwable cause) {
18        super(message, cause);
19    }
20
21
22 }
23

```

**Screenshot 1: IntelliJ IDEA IDE showing the code for DBException class.**

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Explorer:** Shows the project structure with packages like com.ebay.mau.exception, com.ebay.mau.helper, and com.ebay.mau.reporter.
- Code Editor:** Displays the code for the DBException class. The code is as follows:

```

1 package com.ebay.mau.exception;
2
3 public class DBException extends MauiException {
4     private static final long serialVersionUID = -8417471836653675063L;
5
6     public DBException() {
7     }
8
9     public DBException(String message) {
10        super(message);
11    }
12
13    public DBException(Throwable cause) {
14        super(cause);
15    }
16
17    public DBException(String message, Throwable cause) {
18        super(message, cause);
19    }
20
21}

```

- Toolbars and Menus:** Standard IntelliJ IDEA toolbars and menus are visible at the top.
- Right-hand pane:** Shows a tree view of the DBException class, including its methods and fields.
- Bottom:** Shows the Java EE tab selected, along with other tabs like Out, Task, and Call Hierar.

  

**Screenshot 2: IntelliJ IDEA IDE showing the code for WebResponseException class.**

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Explorer:** Shows the project structure with packages like com.ebay.mau.exception, com.ebay.mau.helper, and com.ebay.mau.reporter.
- Code Editor:** Displays the code for the WebResponseException class. The code is as follows:

```

1 package com.ebay.mau.exception;
2
3 import com.ebay.mau.driver.web.element.IPage;
4
5 public class WebResponseException extends RuntimeException {
6     private static final long serialVersionUID = -5365732347362556776L;
7
8     private IPage page = null;
9
10    public WebResponseException() {
11    }
12
13    public WebResponseException(String message) {
14        super(message);
15    }
16
17    public WebResponseException(String message, Throwable cause) {
18        super(message, cause);
19    }
20
21    public WebResponseException(String message, Throwable cause, IPage page) {
22        super(message, cause);
23        this.page = page;
24    }
25
26    public WebResponseException(Throwable cause) {
27        super(cause);
28    }
29
30    public IPage getPage() {
31        return page;
32    }
33
34}

```

- Toolbars and Menus:** Standard IntelliJ IDEA toolbars and menus are visible at the top.
- Right-hand pane:** Shows a tree view of the WebResponseException class, including its methods and fields.
- Bottom:** Shows the Java EE tab selected, along with other tabs like Out, Task, and Call Hierar.

**Screenshot 1: EmailException.java**

```

1 package com.ebay.maul.exception;
2
3 public class EmailException extends MaulException {
4     private static final long serialVersionUID = -27236405695729535L;
5
6     public EmailException() {
7
8     }
9
10    public EmailException(String message) {
11        super(message);
12    }
13
14    public EmailException(Throwable cause) {
15        super(cause);
16    }
17
18    public EmailException(String message, Throwable cause) {
19        super(message, cause);
20    }
21}

```

**Screenshot 2: WebSessionTerminatedException.java**

```

24 * Author: limjishi, created on Aug 11, 2013
25 package com.ebay.maul.exception;
26
27 import org.openqa.selenium.WebDriverException;
28
29 public class WebSessionTerminatedException extends WebDriverException {
30
31     /**
32      */
33     private static final long serialVersionUID = -4870889731005206071L;
34
35     public WebSessionTerminatedException() {
36         super();
37     }
38
39     public WebSessionTerminatedException(Throwable ex) {
40         super(ex);
41     }
42
43 }

```

**Project Explorer**

- com.ebay.maul.exception
  - AbstractWebDriverFactory
  - AndroidCapabilitiesFactory.class
  - ChromeCapabilitiesFactory.class
  - ChromeDriverFactory.class
  - FirefoxCapabilitiesFactory.class
  - FirefoxDriverFactory.class
  - HTMLUnitCapabilitiesFactory.class
  - HTMLUnitDriverFactory.class
  - ICapabilitiesFactory.class
  - IECapabilitiesFactory.class
  - IDriverFactory.class
  - IWebDriverFactory.class
  - OperaCapabilitiesFactory.class
  - OperaDriverFactory.class
  - PhantomJSCapabilitiesFactory.class
  - RemoteDriverFactory.class
  - SafariCapabilitiesFactory.class
  - SafariDriverFactory.class
- com.ebay.maul.exception
  - APIException
  - APIException.class
  - DataCreationException
  - DBException.class
  - DBFinderException.class
  - EmailException
  - MauException
  - PageNotCurrentException
  - SSHException
  - WebResponseException
  - WebSessionTerminatedException
  - XMLException
- com.ebay.maul.helper
  - CALHelper.class
  - ClassContextHelper.class
  - EscapeHelper.class
  - FileHelper.class
  - OSHelper.class
  - StringHelper.class
  - ThreadHelper.class
  - URLHelper.class
  - WaitHelper.class
  - XMLHelper.class
- com.ebay.maul.reporter
  - DetailedLog.class
  - HTMLReporter.class
  - MiniTestResult.class
  - PluginUtil.class

**Members calling constructors of 'WebPageSection' - in workspace**

- WebPageSection(String, By) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String, String) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String) - com.ebay.maul.driver.web.element.WebPageSection

## Helper

**Project Explorer**

- com.ebay.maul.exception
  - DataCreationException.class
  - DBException.class
  - DBFinderException.class
  - EmailException.class
  - MauException.class
  - PageNotCurrentException.class
  - SSHException.class
  - WebResponseException.class
  - WebSessionTerminatedException.class
  - XMLException.class
- com.ebay.maul.helper
  - CALHelper.class
  - ClassContextHelper.class
    - EscapeHelper.class
    - FileHelper.class
    - OSHelper.class
    - StringHelper.class
    - ThreadHelper.class
    - URLHelper.class
    - WaitHelper.class
    - XMLHelper.class
- com.ebay.maul.reporter
  - DetailedLog.class
  - HTMLReporter.class
  - MiniTestResult.class
  - PluginUtil.class
  - XMLReporter.class
- com.ebay.maul.reporter.pluginmodel
  - Mauplugins.class
  - Method.class
  - ObjectFactory.class
  - Page.class
  - Plugin.class
  - Test.class
  - reporter.css
  - reporter.images.lightbox
  - reporter.images.mktree
  - reporter.images.yukontoolbox
  - reporter.js
  - templates
  - META-INF
  - tmp
  - guice-3.0.jar - /Users/gaurshukla.m2/raptor2/com/google/guice/guice-3.0.jar
  - javax.inject-1.jar - /Users/gaurshukla.m2/raptor2/javax/inject/javax/inject-1.jar
  - aopalience-1.0.jar - /Users/gaurshukla.m2/raptor2/aopalience-1.0.jar

**Members calling constructors of 'WebPageSection' - in workspace**

- WebPageSection(String, By) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String, String) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String) - com.ebay.maul.driver.web.element.WebPageSection

```

34 import org.apache.commons.io.FileUtils;
35 import org.apache.log4j.Logger;
36
37 public class FileHelper {
38     static Logger logger = Logger.getLogger(FileHelper.class);
39     final static int BUFFER = 2048;
40
41     private static Map<Integer, FileLock> fileLockMap = new HashMap<Integer, FileLock>();
42
43     public static void extractJar(String storeLocation, Class<?> clazz) throws IOException {
44         File ffpProfile = new File(storeLocation);
45         String location = clazz.getProtectionDomain().getCodeSource()
46             .getLocation().getURI().getLocalPath();
47         location = FileHelper.decodePath(location);
48         Jarfile jar = new Jarfile(location);
49         System.out.println("Extracting " + location);
50         Jarfile.out.println("Extracting " + location);
51         Jarfile.out.println("Directory");
52         Ffprofile.mkdir();
53         // System.out.println("FirefoxProfile: " + profilePath +
54         // System.out.println("FileInputStream");
55         Enumeration<JarEntry> jarEntries = jar.entries();
56         while (jarEntries.hasMoreElements()) {
57             ZipEntry entry = (ZipEntry) jarEntries.nextElement();
58             String currentPath = entry.getName();
59             File destFile = new File(storeLocation, currentEntry);
60             destinationParent = destFile.getParentFile();
61
62             // create the parent directory structure if needed
63             destinationParent.mkdirs();
64             if (entry.isDirectory()) {
65                 BufferedInputStream is = new BufferedInputStream(
66                     jar.getInputStream(entry));
67                 int currentByte;
68                 // establish buffer for writing file
69                 byte data[] = new byte[BUFFER];
70
71                 // write the current file to disk
72                 FileOutputStream fos = new FileOutputStream(destFile);
73                 BufferedOutputStream dest = new BufferedOutputStream(fos,
74                     BUFFER);
75
76                 // read and write until last byte is encountered
77                 while ((currentByte = is.read(data, 0, BUFFER)) != -1) {
78                     dest.write(data, 0, currentByte);
79                 }
80                 dest.flush();
81                 dest.close();
82                 is.close();
83             }
84         }
85         FileUtils.deleteDirectory(new File(storeLocation + "\\META-INF"));
86         if (osHelper.isWindows()) {
87             new File(storeLocation + "\\"
88                 + clazz.getCanonicalName().replaceAll("\\.", "\\")
89                 + ".class").delete();
90         } else {
91             new File(storeLocation + "\\"
92                 + clazz.getCanonicalName().replaceAll("\\.", "/") + ".class")
93                 .delete();
94         }
95     }
96     public static void copyfile(File srcPath, File dstPath) throws IOException {
97         if (srcPath.isDirectory()) {
98             if (dstPath.exists())
99                 dstPath.mkdir();
100
101             String files[] = srcPath.list();
102             for (int i = 0; i < files.length; i++) {
103                 copyfile(new File(srcPath, files[i]), new File(dstPath,
104                     files[i]));
105             }
106         } else {
107             if (!srcPath.exists())
108                 throw new IOException("DIR_NOT_FOUND: " + srcPath);
109             else {
110                 InputStream in = null;
111                 OutputStream out = null;
112                 try {
113                     if (dstPath.getParentFile().exists())
114                         dstPath.getParentFile().mkdirs();
115
116                     in = new FileInputStream(srcPath);
117                     out = new FileOutputStream(dstPath);
118
119                     // Transfer bytes from in to out
120                     byte[] buf = new byte[1024];
121                     int len;
122
123                     while ((len = in.read(buf)) > 0) {
124                         out.write(buf, 0, len);
125                     }
126                 } finally {
127                     if (in != null)
128                         try {
129                             in.close();
130                         } catch (Exception e) {
131                         }
132                     if (out != null)
133                         try {
134                             out.close();
135                         } catch (Exception e) {
136                         }
137                 }
138             }
139         }
140     }
141 }
142
143 public static void copyfile(String srcPath, String dstPath)
144     throws IOException {
145     copyfile(new File(srcPath), new File(dstPath));
146 }
147 */
148 * Deletes a Directory with Files
149 * @param path
150 * @return
151 */
152 public static boolean deletedirectory(File path) {
153     if (path.exists()) {
154         File[] files = path.listFiles();
155         if (files == null) return true;
156         for (int i = 0; i < files.length; i++) {
157
158

```

```

97
98     if (srcPath.isDirectory()) {
99         if (!dstPath.exists()) {
100             dstPath.mkdir();
101
102             String files[] = srcPath.list();
103             for (int i = 0; i < files.length; i++) {
104                 copyfile(new File(srcPath, files[i]), new File(dstPath,
105                     files[i]));
106             }
107         } else {
108             if (!srcPath.exists())
109                 throw new IOException("DIR_NOT_FOUND: " + srcPath);
110             else {
111                 InputStream in = null;
112                 OutputStream out = null;
113                 try {
114                     if (dstPath.getParentFile().exists())
115                         dstPath.getParentFile().mkdirs();
116
117                     in = new FileInputStream(srcPath);
118                     out = new FileOutputStream(dstPath);
119
120                     // Transfer bytes from in to out
121                     byte[] buf = new byte[1024];
122                     int len;
123
124                     while ((len = in.read(buf)) > 0) {
125                         out.write(buf, 0, len);
126                     }
127                 } finally {
128                     if (in != null)
129                         try {
130                             in.close();
131                         } catch (Exception e) {
132                         }
133                     if (out != null)
134                         try {
135                             out.close();
136                         } catch (Exception e) {
137                         }
138                 }
139             }
140         }
141     }
142 }
143
144 public static void copyfile(String srcPath, String dstPath)
145     throws IOException {
146     copyfile(new File(srcPath), new File(dstPath));
147 }
148 */
149 * Deletes a Directory with Files
150 * @param path
151 * @return
152 */
153 public static boolean deletedirectory(File path) {
154     if (path.exists()) {
155         File[] files = path.listFiles();
156         if (files == null) return true;
157         for (int i = 0; i < files.length; i++) {
158
159

```

The screenshot shows an IDE interface with multiple tabs open. The main tab displays Java code for a class named `FileHelper`. The code includes methods for deleting files and directories, performing file locks, and reading file contents. The code is annotated with Javadoc-style comments.

```
161     for (int i = 0; i < files.length; i++) {
162         if (files[i].isDirectory()) {
163             deletedDirectory(files[i]);
164         } else {
165             files[i].delete();
166         }
167     }
168     return path.delete();
169 }
170 */
171 public static boolean deleteDirectory(String dir) {
172     return deleteDirectory(new File(dir));
173 }
174 */
175 /**
176 * Locks a profile to use
177 * @param profilePath
178 * @param max
179 * @return
180 * @throws IOException
181 */
182 public static int performFileLock(String profilePath, int max)
183     throws IOException {
184     FileLock lock = null;
185     File dir = new File(profilePath);
186     if (!dir.exists()) {
187         dir.mkdirs();
188     }
189     RandomAccessFile raf = null;
190     FileChannel fileChannel = null;
191     for (int i = 1; i <= max; i++) {
192         try {
193             logger.info("trying to lock profile..." + i);
194             raf = new RandomAccessFile(profilePath + "/" + i + ".lock", "rw");
195             fileChannel = raf.getChannel();
196             lock = fileChannel.tryLock();
197             if (lock != null && lock.isValid() && lock.isShared()) {
198                 File pdir = new File(profilePath + "/profile_" + i);
199                 if (!pdir.exists())
200                     pdir.mkdirs();
201                 fileLockMap.put(Integer.valueOf(i), lock);
202                 logger.info("Successfully locked profile..." + i);
203             } else {
204                 try {
205                     lock = null;
206                     fileChannel.close();
207                     raf.close();
208                 } catch (Exception e) {
209                     // KEEPME
210                 }
211             }
212         } catch (Exception e) {
213             // no.printStackTrace();
214             try {
215                 lock = null;
216                 fileChannel.close();
217             } catch (Exception e) {
218                 // KEEPME
219             }
220         }
221     }
222     logger.info("UNABLE_TO_LOCK_PROFILE");
223 }
224 */
225 /**
226 * Locking the port
227 * @param startPort
228 * @param max
229 * @return
230 */
231 public static int performPortLock(int startPort, int max) {
232     for (int i = startPort; i <= startPort + max; i++) {
233         try {
234             final ServerSocket ss = new ServerSocket(i);
235             Runnable rble = new Runnable() {
236                 public void run() {
237                     try {
238                         ss.accept();
239                     } catch (Exception e) {
240                     } finally {
241                         try {
242                             logger.info("Releasing lock on port "
243                                 + ss.getLocalPort());
244                             ss.close();
245                             logger.info("Done");
246                         } catch (Exception e) { // KEEPME
247                         }
248                     }
249                 }
250             };
251             Thread thr = new Thread(rble);
252             thr.start();
253             return i;
254         } catch (Exception e) {
255             }
256     }
257     return 0;
258 }
259 */
260 /**
261 * Read contents of a file
262 * @param path
263 * @returns content
264 * @throws IOException
265 */
266 public static String readFromFile(File path) throws IOException {
267     FileInputStream fis = null;
268     try {
269         fis = new FileInputStream(path);
270         return readFromFile(fis);
271     } finally {
272         if (fis != null)
273             fis.close();
274     }
275 }
```

The screenshot shows an IDE interface with multiple tabs open. The main tab displays Java code for a class named `FileHelper`. The code includes methods for performing file locks and port locks. The port lock method creates a server socket on a specified port range and releases the lock once the connection is closed. The code is annotated with Javadoc-style comments.

```
122     try {
123         lock = null;
124         fileChannel.close();
125         raf.close();
126     } catch (Exception e) {
127     }
128 }
129 }
130 */
131 /**
132 * Locking the port
133 * @param startPort
134 * @param max
135 * @return
136 */
137 public static int performPortLock(int startPort, int max) {
138     for (int i = startPort; i <= startPort + max; i++) {
139         try {
140             final ServerSocket ss = new ServerSocket(i);
141             Runnable rble = new Runnable() {
142                 public void run() {
143                     try {
144                         ss.accept();
145                     } catch (Exception e) {
146                     } finally {
147                         try {
148                             logger.info("Releasing lock on port "
149                                 + ss.getLocalPort());
150                             ss.close();
151                             logger.info("Done");
152                         } catch (Exception e) { // KEEPME
153                         }
154                     }
155                 }
156             };
157             Thread thr = new Thread(rble);
158             thr.start();
159             return i;
160         } catch (Exception e) {
161             }
162     }
163     return 0;
164 }
165 */
166 /**
167 * Read contents of a file
168 * @param path
169 * @returns content
170 * @throws IOException
171 */
172 public static String readFromFile(File path) throws IOException {
173     FileInputStream fis = null;
174     try {
175         fis = new FileInputStream(path);
176         return readFromFile(fis);
177     } finally {
178         if (fis != null)
179             fis.close();
180     }
181 }
```

The screenshot shows a Java code editor with the following code:

```
297    public static String readFromFile(InputStream path) throws IOException {
298        InputStreamReader fr = null;
299        BufferedReader br = null;
300        StringBuffer sbContent = new StringBuffer();
301
302        try {
303            fr = new InputStreamReader(path);
304            br = new BufferedReader(fr);
305
306            String line = null;
307            while ((line = br.readLine()) != null)
308                sbContent.append(line).append("\n");
309        } finally {
310            if (br != null) {
311                try {
312                    br.close();
313                } catch (IOException e) {
314                }
315            }
316            if (fr != null) {
317                try {
318                    fr.close();
319                } catch (IOException e) {
320                }
321            }
322        }
323
324        return sbContent.toString();
325    }
326
327    /**
328     * Releases filelock given a profile number
329     *
330     * @param i
331     */
332    public static void releaseFilelock(int i) {
333        Filelock lock = filelockMap.get(Integer.valueOf(i));
334        logger.info("Releasing profile_" + i + "----> " + lock);
335        if (lock != null) {
336            try {
337                FileChannel channel = lock.channel();
338                lock.release();
339                channel.close();
340
341                filelockMap.remove(Integer.valueOf(i));
342                logger.info("Successfully released profile_" + i);
343            } catch (IOException e) {
344                logger.error("Ex", e);
345            }
346        }
347    }
348
349    public static void releasePortlock(int port) {
350        Socket socket = null;
351        OutputStream os = null;
352        try {
353            socket = new Socket("127.0.0.1", port);
354            os = socket.getOutputStream();
355        } catch (Exception e) {
356            logger.warn("Ex", e);
357        } finally {
358            try {
359                os.close();
360            } catch (Exception e) {
361            }
362        }
363    }
364
365    /**
366     * Creates Image from bytes and stores it
367     *
368     * @param path
369     * @param screenShot
370     */
371    public static synchronized void writeImage(String path, byte[] byteArray) {
372        if (byteArray.length == 0)
373            return;
374        System.gc(); // KEEPME
375
376        InputStream in = null;
377        FileOutputStream fos = null;
378        try {
379            File parentDir = new File(path).getParentFile();
380            if (!parentDir.exists())
381                parentDir.mkdirs();
382            by = Base64.decodeBase64(byteArray);
383            in = new ByteArrayInputStream(decodedBy);
384            BufferedImage img = ImageIO.read(in);
385            fos = new FileOutputStream(path);
386            ImageIO.write(img, "png", fos);
387            img = null;
388        } catch (Exception e) {
389            logger.warn(e.getMessage());
390        } finally {
391            if (in != null)
392                try {
393                    in.close();
394                } catch (Exception e) {
395                }
396            if (fos != null)
397                try {
398                    fos.close();
399                } catch (Exception e) {
400                }
401        }
402    }
403
404    public static byte[] readBinaryFile(String inputFileName) {
405        logger.info("Reading in binary file named :" + inputFileName);
406        File file = new File(inputFileName);
407        logger.info("File size : " + file.length());
408        byte[] result = new byte[(int) file.length()];
409        try {
410            InputStream input = null;
411            try {
412                int totalBytesRead = 0;
413                input = new BufferedInputStream(new FileInputStream(file));
414                while (totalBytesRead < result.length) {
415                    int bytesRead = input.read(result, totalBytesRead,
416                        result.length - totalBytesRead);
417                    if (input.read() == -1 || bytesRead < 0)
418                        break;
419                    totalBytesRead += bytesRead;
420                }
421            } catch (IOException e) {
422                logger.error("Error reading file " + file.getName());
423            }
424        } catch (IOException e) {
425            logger.error("Error reading file " + file.getName());
426        }
427    }
428}
```

The screenshot shows a Java code editor with the following code:

```
361
362
363    /**
364     * Creates Image from bytes and stores it
365     *
366     * @param path
367     * @param screenShot
368     */
369    public static synchronized void writeImage(String path, byte[] byteArray) {
370        if (byteArray.length == 0)
371            return;
372        System.gc(); // KEEPME
373
374        InputStream in = null;
375        FileOutputStream fos = null;
376        try {
377            File parentDir = new File(path).getParentFile();
378            if (!parentDir.exists())
379                parentDir.mkdirs();
380            by = Base64.decodeBase64(byteArray);
381            in = new ByteArrayInputStream(decodedBy);
382            BufferedImage img = ImageIO.read(in);
383            fos = new FileOutputStream(path);
384            ImageIO.write(img, "png", fos);
385            img = null;
386        } catch (Exception e) {
387            logger.warn(e.getMessage());
388        } finally {
389            if (in != null)
390                try {
391                    in.close();
392                } catch (Exception e) {
393                }
394            if (fos != null)
395                try {
396                    fos.close();
397                } catch (Exception e) {
398                }
399        }
400    }
401
402    public static byte[] readBinaryFile(String inputFileName) {
403        logger.info("Reading in binary file named :" + inputFileName);
404        File file = new File(inputFileName);
405        logger.info("File size : " + file.length());
406        byte[] result = new byte[(int) file.length()];
407        try {
408            InputStream input = null;
409            try {
410                int totalBytesRead = 0;
411                input = new BufferedInputStream(new FileInputStream(file));
412                while (totalBytesRead < result.length) {
413                    int bytesRead = input.read(result, totalBytesRead,
414                        result.length - totalBytesRead);
415                    if (input.read() == -1 || bytesRead < 0)
416                        break;
417                    totalBytesRead += bytesRead;
418                }
419            } catch (IOException e) {
420                logger.error("Error reading file " + file.getName());
421            }
422        } catch (IOException e) {
423            logger.error("Error reading file " + file.getName());
424        }
425    }
426}
```

The screenshot shows an IDE interface with multiple tabs open. The main tab displays the Java code for the `FileHelper` class. The code is annotated with various annotations such as `@Param`, `@Param is`, and `@throws IOException`. The code implements methods like `readFromInputStream` and `writeToFile`.

```
424     if (bytesRead > 0) {
425         totalBytesRead = totalBytesRead + bytesRead;
426     }
427 }
428 */
429 * the above style is a bit tricky: it places bytes into the
430 * 'result' array; 'result' is an output parameter; the while
431 * loop usually has a single iteration only.
432 *
433 * logger.info("num bytes read: " + totalBytesRead);
434 } finally {
435     logger.info("closing input stream.");
436     if (in != null) {
437         in.close();
438     }
439 } catch (FileNotFoundException e) {
440     logger.info("file not found.");
441 } catch (IOException e) {
442     logger.info(e);
443 }
444 }
445 return result;
446 }
447 */
448 /**
449 * Write to file
450 * @param path
451 * @param is
452 * @throws IOException
453 */
454 public static void writeToFile(String path, InputStream is)
455 throws IOException {
456     System.gc(); // KEEPME
457
458     BufferedInputStream bis = new BufferedInputStream(is);
459     BufferedOutputStream bos = new BufferedOutputStream(
460             new FileOutputStream(path));
461
462     try {
463         File parentDir = new File(path).getParentFile();
464         if (!parentDir.exists()) {
465             parentDir.mkdirs();
466         }
467         byte[] bArr = new byte[4096];
468         int nRead = 0;
469         while ((nRead = bis.read(bArr)) != -1) {
470             bos.write(bArr, 0, nRead);
471         }
472     } finally {
473         try {
474             bos.close();
475         } catch (Exception e) {
476         }
477         try {
478             bis.close();
479         } catch (Exception e) {
480         }
481         try {
482             is.close();
483         } catch (Exception e) {
484         }
485     };
486 }
487 }
```

The screenshot shows an IDE interface with multiple tabs open. The main tab displays the Java code for the `FileHelper` class. The code includes methods for writing to files and getting the latest file in a folder. It uses `FileInputStream`, `FileOutputStream`, and `BufferedWriter` classes.

```
498     public static void writeToFile(String path, String content)
499     throws IOException {
500
501     System.gc(); // KEEPME
502
503     FileInputStream fos = null;
504     OutputStreamWriter osw = null;
505     BufferedWriter bw = null;
506     try {
507         File parentDir = new File(path).getParentFile();
508         try {
509             if (!parentDir.exists()) {
510                 parentDir.mkdirs();
511             }
512         } catch (Exception ignore) {
513         }
514         fos = new FileOutputStream(path);
515         osw = new OutputStreamWriter(fos, "UTF-8");
516         bw = new BufferedWriter(osw);
517         bw.write(content);
518     } finally {
519         if (bw != null) {
520             try {
521                 bw.close();
522             } catch (Exception e) {
523                 // KEEPME
524             }
525         }
526         if (osw != null) {
527             try {
528                 osw.close();
529             } catch (Exception e) {
530                 // KEEPME
531             }
532         }
533         if (fos != null) {
534             try {
535                 fos.close();
536             } catch (Exception e) {
537                 // KEEPME
538             }
539         }
540     }
541     public static String getLatestFile(String folder) {
542         String file = null;
543         File folderFile = new File(folder);
544         if (!folderFile.exists() && folderFile.isDirectory()) {
545             File[] files = folderFile.listFiles();
546             long date = 0;
547             for (int i = 0; i < files.length; i++) {
548                 if (files[i].lastModified() > date) {
549                     date = files[i].lastModified();
550                     file = files[i].getAbsolutePath();
551                 }
552             }
553         }
554         return file;
555     }
556     public static String decodePath(String path)
557     throws UnsupportedEncodingException {
558         return URLDecoder.decode(path, "UTF-8");
559     }
560 }
```

OSHelper.java

```

8  public class OSHelper {
9
10     public static String getOSName(){
11         return System.getProperty("os.name");
12     }
13
14     public static boolean isMac(){
15         return getOSName().startsWith("Mac");
16     }
17
18     public static boolean isWindows(){
19         return getOSName().startsWith("Win");
20     }
21
22     public static String getOSBits(){
23         return System.getProperty("os.arch");
24     }
25
26
27     public static boolean is32(){
28         return getOSBits().equals("x86");
29     }
30
31     public static boolean is64(){
32         if (!isWindows())
33         {
34             return (System.getenv("ProgramW6432") != null);
35         }
36         else
37             return !getOSBits().equals("x86");
38     }
39
40     private static List<String> executeCommand(String cmd){
41         List<String> output = new ArrayList<String>();
42         Process p;
43         try {
44             p = Runtime.getRuntime().exec(cmd);
45         } catch (IOException e) {
46             return output;
47         }
48         BufferedReader stdInput = new BufferedReader(new InputStreamReader(p.getInputStream()),8*1024);
49         String s = null;
50         try {
51             while ((s = stdInput.readLine()) != null)
52                 output.add(s);
53         } catch (IOException e) {
54             return output;
55         }
56         return output;
57     }
58     public static int getIEVersion(){
59         List<String> output;
60         output = executeCommand("reg query \"HKEY\Software\Microsoft\Internet Explorer\" /v svcVersion");
61         if(output.size()>3)
62         {
63             output = executeCommand("reg query \"HKEY\Software\Microsoft\Internet Explorer\" /v Version");
64         }
65         String internet_explorer_value = Output.get(2);
66         String version = internet_explorer_value.trim().split("\\\\")[-1];
67         version = version.trim().split("\\\\")[-1];
68         return Integer.parseInt(version);
69     }
70
71     public static String getSlash(){
72         if(!isWindows())
73             return "\\\\";
74         else
75             return "/";
76     }
77
78     public static void main(String[] args){
79         System.out.println(getIEVersion());
80     }
81 }
82 
```

DBException.java

```

19     public static boolean isWindows(){
20         return getOSName().startsWith("Win");
21     }
22
23     public static String getOSBits(){
24         return System.getProperty("os.arch");
25     }
26
27     public static boolean is32(){
28         return getOSBits().equals("x86");
29     }
30
31     public static boolean is64(){
32         if (!isWindows())
33         {
34             return (System.getenv("ProgramW6432") != null);
35         }
36         else
37             return !getOSBits().equals("x86");
38     }
39
40     private static List<String> executeCommand(String cmd){
41         List<String> output = new ArrayList<String>();
42         Process p;
43         try {
44             p = Runtime.getRuntime().exec(cmd);
45         } catch (IOException e) {
46             return output;
47         }
48         BufferedReader stdInput = new BufferedReader(new InputStreamReader(p.getInputStream()),8*1024);
49         String s = null;
50         try {
51             while ((s = stdInput.readLine()) != null)
52                 output.add(s);
53         } catch (IOException e) {
54             return output;
55         }
56         return output;
57     }
58     public static int getIEVersion(){
59         List<String> output;
60         output = executeCommand("reg query \"HKEY\Software\Microsoft\Internet Explorer\" /v svcVersion");
61         if(output.size()>3)
62         {
63             output = executeCommand("reg query \"HKEY\Software\Microsoft\Internet Explorer\" /v Version");
64         }
65         String internet_explorer_value = Output.get(2);
66         String version = internet_explorer_value.trim().split("\\\\")[-1];
67         version = version.trim().split("\\\\")[-1];
68         return Integer.parseInt(version);
69     }
70
71     public static String getSlash(){
72         if(isWindows())
73             return "\\\\";
74         else
75             return "/";
76     }
77
78     public static void main(String[] args){
79         System.out.println(getIEVersion());
80     }
81 }
82 
```

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The top menu bar includes 'File', 'Edit', 'Select', 'Run', 'View', 'Search', 'Help', 'Quick Access', 'Java EE', and 'Debug'. The left sidebar contains 'Project Explorer', 'Navigator', 'Type Hierarchy', and 'Outline' tabs. The main editor area displays the code for `StringHelper.java`, which is part of the `com.ebay.maul.helper` package. The code implements methods for constructing method signatures and generating random UUIDs.

```

1 package com.ebay.maul.helper;
2
3 import java.lang.reflect.Method;
4 import java.math.BigInteger;
5 import java.security.MessageDigest;
6 import java.util.UUID;
7
8 import com.ebay.maul.controller.ContextManager;
9
10 public class StringHelper {
11
12     public static String constructMethodSignature(Method method, Object[] parameters) {
13         return method.getDeclaringClass().getCanonicalName() + "." + method.getName() + "(" + constructParameterString(parameters) + ")";
14     }
15
16     public static String constructParameterString(Object[] parameters) {
17         StringBuilder sbParam = new StringBuilder();
18
19         if (parameters != null) {
20             for (int i = 0; i < parameters.length; i++) {
21                 if (parameters[i] instanceof String) {
22                     sbParam.append("'");
23                 } else if (parameters[i] instanceof java.lang.String) {
24                     sbParam.append("\"").append(parameters[i]).append("\", ");
25                 } else {
26                     sbParam.append(parameters[i]).append(", ");
27                 }
28             }
29         }
30
31         if (sbParam.length() > 0)
32             sbParam.delete(sbParam.length() - 2, sbParam.length() - 1);
33
34         return sbParam.toString();
35     }
36
37     public static String getRandomHashCode(String seed) {
38         byte[] data = (ContextManager.getThreadContext().getTestMethodSignature() + UUID.randomUUID().getLeastSignificantBits() + seed).getBytes();
39         try {
40             MessageDigest digest = MessageDigest.getInstance("MD5");
41             return new BigInteger(1, digest.digest(data)).toString(16);
42         } catch (Exception e2) {
43             // Ok, we won't digest
44             return new String(data);
45         }
46     }
47 }

```

This screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The top menu bar and sidebar are identical to the previous screenshot. The main editor area displays the code for `ThreadHelper.java`, which is part of the `com.ebay.maul.helper` package. The code defines a static method `waitForSeconds` that waits for a specified number of seconds.

```

1 package com.ebay.maul.helper;
2
3 public class ThreadHelper {
4     /**
5      * Wait for seconds
6      * @param seconds
7      */
8     public static void waitForSeconds(int seconds) {
9         long timeout = 1000 * 60 * 3;
10        if(ContextManager.getThreadContext().isWait()) timeout =
11            ContextManager.getThreadContext().getWebSessionTimeout();
12        if (seconds < 1 || timeout < 1) throw new RuntimeException("Can not wait for " +
13            seconds + " seconds. Because that is longer than web session timeout of " +
14            timeout / 1000 +
15            " seconds which is defined in the testing configuration file.");
16        try {
17            Thread.sleep(seconds * 1000); // KEEPME
18        } catch (InterruptedException ignore) {
19        }
20    }
21 }

```

The bottom of the screen shows the 'Call Hierar' tab selected in the 'Members calling constructors of 'WebPageSection' - in workspace' section, listing three entries:

- `WebPageSection(String, By) - com.ebay.maul.driver.web.element.WebPageSection`
- `WebPageSection(String, String) - com.ebay.maul.driver.web.element.WebPageSection`
- `WebPageSection(String) - com.ebay.maul.driver.web.element.WebPageSection`

```

40 public class URLHelper {
41     ...
42     private static final Logger logger = Logger.getLogger(URLHelper.class);
43     private static final String HTTPS_PROTOCOL = "https";
44     ...
45     // private static final int HTTPS_PORT = 443;
46
47     public static String convert(String url) {
48         String urlString;
49         ContextManager contextManager = ContextManager.getThreadContext();
50         URLConvertClass converterClass = contextManager.getURLConvertClass();
51         if (converterClass != null) {
52             try {
53                 Class<?> converterClass = Class.forName(urlConvertClass);
54                 Object converter = converterClass.newInstance();
55                 Method convert = converterClass.getMethod("convertURL",
56                     String.class);
57                 return (String) convert.invoke(converter, url);
58             } catch (Exception e) {
59                 logger.warn("Convert URL failed", e);
60             }
61         }
62     }
63     return urlString;
64 }
65
66 public static String getRandomHashCode(String seed) {
67     String signature;
68     if (ContextManager.getThreadContext() == null)
69         signature = ContextManager.getContext().getThreadContext()
70             .getTestMethodSignature();
71     else
72         signature = "";
73     byte[] data = (signature + UUID.randomUUID().getLeastSignificantBits() + seed)
74         .getBytes();
75     try {
76         MessageDigest digest = MessageDigest.getInstance("MD5");
77         return new BigInteger(1, digest.digest(data)).toString(16);
78     } catch (Exception e2) {
79         // OK, we won't digest
80     }
81     return new String(data);
82 }
83
84 public static void main(String[] args) throws Exception {
85     //open("http://besbuilt.qa.ebay.com/odm/v3/console?ConsumerType=SGFConsumerForBDS&eventTypes=SGF.BDX.JOB.NER&eventTypes=SGF.BDX.RECURRING.JOB&eventTypes=SGF.BDX.SURJOB.NEW&reportType=ConsumerHostRep
86
87     //String urlString= "http://www.mess.stratus.qa.ebay.com/messsvc/EBay/users/1192676159?paymentInstrumentRefId=7409694308&creditCardType=MASTER&paymentInstrumentType=CreditCard&client=EBP&lostFour=0424&expiration
88
89 }
90
91
92 public static String encode(String urlString) throws MalformedURLException, UnsupportedEncodingException {
93     URL url = null;
94     if(urlString==null) return urlString;
95     if(urlString.indexOf('?')>-1) return urlString;
96     String[] params = url.getParameter().split("&");
97     String encodedQueryString = "";
98     for(int i=0;i<params.length;i++) {
99         if(params[i].contains("=")) {
100             String key = params[i].split("=")[0];
101             String value = params[i].substring(key.length()+1,params[i].length());
102             encodedQueryString = encodedQueryString + URLencoder.encode(key,"UTF-8")+"="+URLencoder.encode(value,"UTF-8") +"&";
103         }
104     }
105     if(encodedQueryString.endsWith("&"))encodedQueryString=encodedQueryString.substring(0,encodedQueryString.length()-1);
106     String encodedUrl = urlString.replace(url.getParameter(), encodedQueryString);
107
108     //System.out.println(encodedUrl);
109     return encodedUrl;
110 }
111
112 /**
113 * Open URL using URLConnection
114 * @param url
115 * @return content of the page
116 * @throws Exception
117 */
118 public static String open(String url) throws Exception {
119     return open(url, false);
120 }
121
122 public static String open(String url, boolean validate) throws Exception {
123     return open(url, validate, 30 * 1000, true);
124 }
125
126 /**
127 * @param url
128 * @param validate : default: false
129 * @param useProxy : true - will use webProxyAddress in context, default:true
130 * @return
131 * @throws Exception
132 */
133 public static String open(String url, boolean validate, boolean useProxy) throws Exception {
134     return open(url, validate, 30 * 1000, useProxy);
135 }
136
137 /**
138 * Open URL using URLConnection
139 * @param url
140 * @return
141 * @throws Exception
142 */
143 public static String open(String url, boolean validate, int timeout,boolean useProxy)
144 throws Exception {
145
146     InputStream is = null;
147     BufferedReader br = null;
148     InputStreamReader isr = null;
149
150     try {
151         is = openInputStream(url, validate, timeout,useProxy);
152         is = new InputStreamReader(is, "UTF-8");
153         br = new BufferedReader(is);
154         StringBuffer sb = new StringBuffer();
155         String line = null;
156         while ((line = br.readLine()) != null) {
157             sb.append(line).append("\n");
158         }
159     }

```

```

160     catch (IOException e) {
161         logger.error("Error reading from URL "+url,e);
162     }
163     finally {
164         if(br!=null)try{br.close();}catch(IOException e){logger.error("Error closing BufferedReader "+br,e);}
165         if(isr!=null)try{isr.close();}catch(IOException e){logger.error("Error closing InputStreamReader "+isr,e);}
166         if(is!=null)try{is.close();}catch(IOException e){logger.error("Error closing InputStream "+is,e);}
167     }
168     return sb.toString();
169 }
170
171 /**
172 * Open URL using URLConnection
173 * @param url
174 * @param validate : default: false
175 * @param useProxy : true - will use webProxyAddress in context, default:true
176 * @return
177 * @throws Exception
178 */
179 public static String open(String url, boolean validate, int timeout,boolean useProxy)
180 throws Exception {
181
182     InputStream is = null;
183     BufferedReader br = null;
184     InputStreamReader isr = null;
185
186     try {
187         is = openInputStream(url, validate, timeout,useProxy);
188         is = new InputStreamReader(is, "UTF-8");
189         br = new BufferedReader(is);
190         StringBuffer sb = new StringBuffer();
191         String line = null;
192         while ((line = br.readLine()) != null) {
193             sb.append(line).append("\n");
194         }
195     }

```

```

178     return sb.toString();
179   } catch (Throwable e) {
180     throw new MauiException(e.getMessage(), e);
181   } finally {
182     if (br != null) {
183       try {
184         br.close();
185       } catch (IOException e) {
186       }
187     }
188     if (isr != null) {
189       try {
190         isr.close();
191       } catch (IOException e) {
192       }
193     }
194   }
195 }
196 /**
197 * Open the URL, remember close the input stream after usage
198 * @param url
199 * @return InputStream
200 * @throws Exception
201 */
202 public static InputStream openInputStream(String url) throws Exception {
203   return openInputStream(url, false, 30 * 1000, true);
204 }
205 /**
206 * Open the URL, remember close the input stream after usage
207 * @param url
208 * @param convert
209 * @param convert
210 * @return InputStream
211 * @throws Exception
212 */
213 public static InputStream openAsStream(String url, boolean convert, boolean useProxy)
214   throws Exception {
215   return openAsStream(url, convert, 30 * 1000, useProxy);
216 }
217 /**
218 * Open the URL, remember close the input stream after usage
219 * @param url
220 * @param convert
221 * @param convert
222 * @param timeout
223 * @param useProxy
224 * @return InputStream
225 * @throws Exception
226 */
227 private static InputStream openAsStream(String url, boolean convert,
228   int timeout, boolean useProxy) throws Exception {
229   // validate URL
230   if (convert)
231     convert(url);
232
233   if (url.startsWith("HTTPS_PROTOCOL"))
234     return openHttpsUrlAsStream(url, useProxy);
235
236   if (url.startsWith("http://")) {
237     InputStream is = null;
238     URLConnection connection = null;
239     try {
240       Context context = ContextManager.getThreadContext();
241
242       if (context.getWebProxyAddress() != null) {
243         String host = context.getWebProxyAddress().split(":")[0];
244         int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
245         Proxy proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress(host, port));
246         connection = new URL(url).openConnection(proxy);
247       } else {
248         connection = new URL(url).openConnection();
249       }
250
251       connection.setConnectTimeout(timeout);
252       connection.setReadTimeout(timeout);
253       is = connection.getInputStream();
254       return is;
255     } catch (Throwable e) {
256       // e.printStackTrace();
257       throw new MauiException(e.getMessage(), e);
258     }
259   }
260
261   @SuppressWarnings("deprecation")
262   public static String openHttpUrl(String url) throws IOException {
263     URL newUrl = new URL(url);
264     Protocol myhttps = new Protocol("https",
265       new EasySLProtocolSocketFactory(), newUrl.getPort());
266     Protocol.registerProtocol("https", myhttps);
267
268     HttpClient httpClient = new HttpClient();
269     String response = null;
270     GetMethod httpget = new GetMethod(url);
271     try {
272       // httpClient.getHostConfiguration().setHost(newUrl.getHost());
273       // myhttps.getProtocolProtocol("https", new
274       // EasySLProtocolSocketFactory());
275       httpClient.executeMethod(httpget);
276       response = httpget.getResponseBodyAsString();
277     } finally {
278       httpget.releaseConnection();
279     }
280     return response;
281   }
282
283   @SuppressWarnings("deprecation")
284   public static InputStream openHttpsUrlAsStream(String url, boolean useProxy)
285   throws IOException {
286     URL newUrl = new URL(url);
287     Protocol myhttps = new Protocol("https",
288       new EasySLProtocolSocketFactory(), newUrl.getDefaultPort());
289     Protocol.registerProtocol("https", myhttps);
290     HttpClient httpClient = new HttpClient();
291     Context context = ContextManager.getThreadContext();
292     if (useProxy && context.getWebProxyAddress() != null) {
293       String host = context.getWebProxyAddress().split(":")[0];
294       int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
295       httpClient.getHostConfiguration().setProxy(host, port);
296     }
297
298     InputStream response = null;
299     GetMethod httpget = new GetMethod(url);
300
301     httpClient.executeMethod(httpget);
302     response = httpget.getResponseBodyAsStream();
303
304     return response;
305   }

```

Screenshot of two Java code editors showing WaitHelper class implementations.

**Top Editor (mauicomponent-ebayIndia package):**

```

26 public class WaitHelper {
27     static long TIMEOUT_S = 60;
28     static int INT_TIMEOUT_S = Integer.parseInt(Long.toString(TIMEOUT_S));
29     static long WAIT_TIMEOUT = 30L;
30     protected static final Logger logger = Logging.getLogger(WebElement.class);
31
32     protected static Function<WebDriver, WebElement> presenceOfElementLocated(
33         final By locator) {
34         return new Function<WebDriver, WebElement>() {
35             public WebElement apply(WebDriver driver) {
36                 return driver.findElement(locator);
37             }
38         };
39     }
40
41     /**
42      * Waits for the element to be visible until a timeout of 30 secs.
43      *
44      * @param driver
45      * @param locator
46      */
47     public static void waitForElementToBeVisible(final WebDriver driver,
48         final By locator) throws RuntimeException {
49         Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
50         try {
51             wait.until(new ExpectedCondition<WebElement>() {
52                 public WebElement apply(WebDriver driver) {
53                     // driver.switchTo().defaultContent();
54                     WebElement element = driver.findElement(locator);
55                     if (element.isDisplayed()) {
56                         return element;
57                     }
58                 }
59             });
60         } catch (Exception e) {
61             throw new RuntimeException("Exception while waiting for " + locator
62                 + ". Exception:" + e + " on " + driver.getCurrentUrl());
63         }
64     }
65
66     /**
67      * Waits for the element to be visible and return element
68      *
69      * @param driver
70      * @param locator
71      * @param timeout
72      * @param text
73      */
74     /**
75      * Waits for the element to be visible until the specified timeout.
76      *
77      * @param driver
78      * @param locator
79      * @param timeout
80      * @param text
81      */
82     public static void waitForElementAndReturnElement(
83         final WebDriver driver, final By locator) throws RuntimeException {
84         waitForElementToBeVisible(driver, locator);
85         return driver.findElement(locator);
86     }
87
88     /**
89      * Waits for the element to be visible until the specified timeout.
90      *
91      * @param driver
92      * @param locator
93      * @param timeout
94      */
95     public static void waitForElementToBeVisible(final WebDriver driver,
96         final By locator, long timeout) throws RuntimeException {
97         Wait<WebDriver> wait = new WebDriverWait(driver, timeout);
98         try {
99             wait.until(new ExpectedCondition<WebElement>() {
100                 public WebElement apply(WebDriver driver) {
101                     // driver.switchTo().defaultContent();
102                 }
103             });
104         } catch (Exception e) {
105             throw new RuntimeException("Exception while waiting for " + locator
106                 + ". Exception:" + e);
107         }
108     }
109
110     /**
111      * Waits for the given text until timing out at 30 secs.
112      *
113      * @param driver
114      * @param locator
115      * @param text
116      */
117     public static void waitUntilText(final WebDriver driver, final By locator,
118         final String text) {
119         Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
120         try {
121             wait.until(new ExpectedCondition<Boolean>() {
122                 public Boolean apply(WebDriver webDriver) {
123                     String currentText = "";
124                     try {
125                         currentText = driver.findElement(locator).getText();
126                     } catch (Exception e) {
127                         ignoreIfElementIsNotPresent();
128                     }
129                     logger.info("Waiting for:" + text + " Found:" + currentText);
130                     return currentText.contains(text);
131                 }
132             });
133         } catch (Exception e) {
134             throw new RuntimeException("Exception while waiting for text "
135                 + text + " in " + locator + ". Exception:" + e);
136         }
137     }
138
139     /**
140      * Waits until the given element is either hidden or deleted.
141      *
142      * @param locator
143      * @param timeout
144      */
145     public static void waitUntilElementDisappears(final WebDriver driver,
146         final By locator) {
147         Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
148         try {
149             wait.until(new ExpectedCondition<Boolean>() {
150                 public Boolean apply(WebDriver driver) {
151                     try {
152                         WebElement element = driver.findElement(locator);
153                         logger.info(locator + " spotted.");
154                         return Boolean.valueOf(element.isDisplayed());
155                     } catch (NoSuchElementException e) {
156                         logger.info(locator + " disappeared.");
157                         return Boolean.TRUE;
158                     } catch (StaleElementReferenceException se) {
159                         logger.info(locator + " disappeared.");
160                     }
161                 }
162             });
163         } catch (Exception e) {
164             throw new RuntimeException("Exception while waiting for element "
165                 + locator + ". Exception:" + e);
166         }
167     }
168
169 }

```

**Bottom Editor (mauicomponent-ebayIndia package):**

```

90     // driver.switchTo().defaultContent();
91     WebElement element = driver.findElement(locator);
92     if (element.isDisplayed()) {
93         return element;
94     }
95     return null;
96 }
97 } catch (Exception e) {
98     throw new RuntimeException("Exception while waiting for " + locator
99         + ". Exception:" + e);
100 }
101
102 /**
103  * Waits for the given text until timing out at 30 secs.
104  *
105  * @param driver
106  * @param locator
107  * @param text
108  */
109 public static void waitUntilText(final WebDriver driver, final By locator,
110     final String text) {
111     Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
112     try {
113         wait.until(new ExpectedCondition<Boolean>() {
114             public Boolean apply(WebDriver webDriver) {
115                 String currentText = "";
116                 try {
117                     currentText = driver.findElement(locator).getText();
118                 } catch (Exception e) {
119                     ignoreIfElementIsNotPresent();
120                 }
121                 logger.info("Waiting for:" + text + " Found:" + currentText);
122                 return currentText.contains(text);
123             }
124         });
125     } catch (Exception e) {
126         throw new RuntimeException("Exception while waiting for text "
127             + text + " in " + locator + ". Exception:" + e);
128     }
129 }
130
131 /**
132  * Waits until the given element is either hidden or deleted.
133  *
134  * @param locator
135  * @param timeout
136  */
137 public static void waitUntilElementDisappears(final WebDriver driver,
138     final By locator) {
139     Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
140     try {
141         wait.until(new ExpectedCondition<Boolean>() {
142             public Boolean apply(WebDriver driver) {
143                 try {
144                     WebElement element = driver.findElement(locator);
145                     logger.info(locator + " spotted.");
146                     return Boolean.valueOf(element.isDisplayed());
147                 } catch (NoSuchElementException e) {
148                     logger.info(locator + " disappeared.");
149                     return Boolean.TRUE;
150                 } catch (StaleElementReferenceException se) {
151                     logger.info(locator + " disappeared.");
152                 }
153             }
154         });
155     } catch (Exception e) {
156         throw new RuntimeException("Exception while waiting for element "
157             + locator + ". Exception:" + e);
158     }
159 }

```

The screenshot shows an IDE interface with multiple tabs open. The active tab is 'WaitHelper.class'. The code is a Java class named 'WaitHelper' with several static methods for waiting on web driver elements. The methods include 'waitForElementVisible', 'waitForElementStartingWithString', 'waitForElementContainingString', 'waitForCurrentUrlToContainString', and 'waitForCurrentUrlToMatchString'. Each method uses WebDriverWait to apply a condition and handle exceptions.

```
153     } catch (StaleElementReferenceException se) {
154         logger.info(locator + " disappeared.");
155         return Boolean.TRUE;
156     }
157 }
158 } catch (Exception e) {
159     throw new RuntimeException(locator + " did not disappear.");
160 }
161 }
162 }
163
164 public static void waitForTitleStartingWithString(Final WebDriver driver,
165     final String title) {
166     try {
167         (new WebDriverWait(driver, WAIT_TIMEOUT))
168             .until(new ExpectedCondition<Boolean>() {
169                 public Boolean apply(WebDriver d) {
170                     return d.getTitle().startsWith(title);
171                 }
172             });
173     } catch (Exception e) {
174         throw new RuntimeException(title
175             + " did not show up after polling for " + WAIT_TIMEOUT
176             + " secs.");
177     }
178 }
179
180 public static void waitForTitleContainingString(Final WebDriver driver,
181     final String title) {
182     try {
183         (new WebDriverWait(driver, WAIT_TIMEOUT))
184             .until(new ExpectedCondition<Boolean>() {
185                 public Boolean apply(WebDriver d) {
186                     return d.getTitle().contains(title);
187                 }
188             });
189     } catch (Exception e) {
190         throw new RuntimeException(title
191             + " did not show up after polling for " + WAIT_TIMEOUT
192             + " secs.");
193     }
194 }
195
196 public static void waitForCurrentUrlToContainString(Final WebDriver driver,
197     final String url) {
198     try {
199         (new WebDriverWait(driver, WAIT_TIMEOUT))
200             .until(new ExpectedCondition<Boolean>() {
201                 public Boolean apply(WebDriver d) {
202                     return d.getCurrentUrl().contains(url);
203                 }
204             });
205     } catch (Exception e) {
206         throw new RuntimeException(url
207             + " was not present in current url: "
208             + driver.getCurrentUrl() + " after polling for "
209             + WAIT_TIMEOUT + " secs.");
210     }
211 }
212
213 public static void waitForCurrentUrlToMatchString(Final WebDriver driver,
214     final String url) {
215     try {
216         (new WebDriverWait(driver, WAIT_TIMEOUT))
217             .until(new ExpectedCondition<Boolean>() {
218                 public Boolean apply(WebDriver d) {
219                     return d.getCurrentUrl().trim().equals(url.trim());
220                 }
221             });
222     } catch (Exception e) {
223         throw new RuntimeException(url
224             + " was not present in current url: "
225             + driver.getCurrentUrl() + " after polling for "
226             + WAIT_TIMEOUT + " secs.");
227     }
228 }
229
230 /**
231 * Explicitly waits for the specified milliseconds. Use this method only if
232 * absolutely necessary.
233 */
234
235 /**
236 * @param milliseconds
237 */
238 public static void holdUntil(long milliseconds) {
239     long waitUntilTime = System.currentTimeMillis() + milliseconds;
240     while (System.currentTimeMillis() < waitUntilTime) {
241         // do nothing just wait for seconds.
242     }
243 }
244
245 public static void waitForSeconds(WebDriver driver, int waitTime) {
246     driver.manage().timeouts().implicitlyWait(waitTime, TimeUnit.SECONDS);
247 }
248 }
```

This screenshot shows the same IDE interface as the first one, but the active tab is now 'WaitHelper.class'. The code is identical to the first screenshot, representing the Java implementation of the 'WaitHelper' class with its various static methods for waiting on web driver elements.

```
182     try {
183         (new WebDriverWait(driver, WAIT_TIMEOUT))
184             .until(new ExpectedCondition<Boolean>() {
185                 public Boolean apply(WebDriver d) {
186                     return d.getTitle().contains(title);
187                 }
188             });
189     } catch (Exception e) {
190         throw new RuntimeException(title
191             + " did not show up after polling for " + WAIT_TIMEOUT
192             + " secs.");
193     }
194 }
195
196 public static void waitForCurrentUrlToContainString(Final WebDriver driver,
197     final String url) {
198     try {
199         (new WebDriverWait(driver, WAIT_TIMEOUT))
200             .until(new ExpectedCondition<Boolean>() {
201                 public Boolean apply(WebDriver d) {
202                     return d.getCurrentUrl().contains(url);
203                 }
204             });
205     } catch (Exception e) {
206         throw new RuntimeException(url
207             + " was not present in current url: "
208             + driver.getCurrentUrl() + " after polling for "
209             + WAIT_TIMEOUT + " secs.");
210     }
211 }
212
213 public static void waitForCurrentUrlToMatchString(Final WebDriver driver,
214     final String url) {
215     try {
216         (new WebDriverWait(driver, WAIT_TIMEOUT))
217             .until(new ExpectedCondition<Boolean>() {
218                 public Boolean apply(WebDriver d) {
219                     return d.getCurrentUrl().trim().equals(url.trim());
220                 }
221             });
222     } catch (Exception e) {
223         throw new RuntimeException(url
224             + " did not match current url: "
225             + driver.getCurrentUrl() + " after polling for "
226             + WAIT_TIMEOUT + " secs.");
227     }
228 }
229
230 /**
231 * Explicitly waits for the specified milliseconds. Use this method only if
232 * absolutely necessary.
233 */
234
235 /**
236 * @param milliseconds
237 */
238 public static void holdUntil(long milliseconds) {
239     long waitUntilTime = System.currentTimeMillis() + milliseconds;
240     while (System.currentTimeMillis() < waitUntilTime) {
241         // do nothing just wait for seconds.
242     }
243 }
244
245 public static void waitForSeconds(WebDriver driver, int waitTime) {
246     driver.manage().timeouts().implicitlyWait(waitTime, TimeUnit.SECONDS);
247 }
248 }
```

Screenshot of the Eclipse IDE interface showing the Java EE perspective. The left side features the Project Explorer, Navigator, and Type Hierarchy views. The center is dominated by the code editor displaying the XMLHelper.java file. The right side includes the Java EE tools view, which shows the XMLHelper class definition with methods getXMLNodes and getXmlNode.

```

1 package com.ebay.maul.helper;
2
3 import java.io.File;
4 import java.io.StringWriter;
5
6 import javax.xml.XMLConstants;
7 import javax.xml.bind.JAXBContext;
8 import javax.xml.bind.Marshaller;
9 import javax.xml.parsers.DocumentBuilder;
10 import javax.xml.parsers.DocumentBuilderFactory;
11
12 import org.w3c.dom.Document;
13 import org.w3c.dom.NodeList;
14
15 public class XMLHelper {
16     public static NodeList getXMLNodes(String xmlFileName, String tagName) {
17         NodeList nList = null;
18         try {
19             File configFile = new File(xmlFileName);
20             DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
21             DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
22             Document doc = dbBuilder.parse(configFile);
23             doc.getDocumentElement().normalize();
24             nList = doc.getElementsByTagName(tagName);
25         } catch (Exception e) {
26             e.printStackTrace();
27         }
28     }
29 }
30
31     public static <T> String getXml(T response) {
32     try {
33         JAXBContext jaxbContext = JAXBContext.newInstance(response.getClass());
34         Marshaller jaxbM = jaxbContext.createMarshaller();
35
36         // output pretty printed
37         jaxbM.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
38
39         StringWriter writer = new StringWriter();
40
41         jaxbM.marshal(response, writer);
42
43         return writer.toString();
44     } catch (JAXBException e) {
45         e.printStackTrace();
46     }
47     return "";
48 }
49
50 }
51

```

The code editor displays the XMLHelper.java file with the following content:

- Imports: java.io.File, java.io.StringWriter, javax.xml.XMLConstants, javax.xml.bind.JAXBContext, javax.xml.bind.Marshaller, javax.xml.parsers.DocumentBuilder, javax.xml.parsers.DocumentBuilderFactory, org.w3c.dom.Document, org.w3c.dom.NodeList.
- Class: XMLHelper.
- Constructor: None.
- Method 1: getXMLNodes(String xmlFileName, String tagName). This method reads an XML file from the specified file name and returns a NodeList of elements matching the given tag name.
- Method 2: getXml(T response). This method takes a response object, creates a JAXBContext for its class, and uses a Marshaller to convert it to a pretty-printed XML string.