AutoComment User Study

Information Letter for Quality of Source Code Comments

You are invited to participate in a research study conducted by Edmund Wong and Jinqiu Yang, under the supervision of Prof. Lin Tan of the Electrical and Computer Engineering Department at University of Waterloo, Canada. The objective of the research study is to assess the quality of source code comments.

If you decide to volunteer, you will be asked to complete a 1 hour session. 10 minutes will be spent on training, and the remainder on some tasks we ask you to perform. As a participant, you would be asked to write comments for source code segments. After that, we will provide you with comments that describe the given code segments, and you will rate the given comments' accuracy, adequacy, conciseness and usefulness based on a five-point scale. An example question sheet will be provided prior to the study.

Participants must have at least one year of programming experience in Java. Participation in this study is voluntary. You may decline to answer any questions that you do not wish to answer and you can withdraw your participation at any time by advising the researcher. There are no known or anticipated risk from participating in this study. No direct benefit is anticipated from this study.

This is an in-person study that will take approximately 1 hour at DC 3573. In appreciation, we will remunerate you $10. If you withdraw participation you will receive $5 per half hour. The amount received is taxable. It is your responsibility to report the amount received for income tax purposes.

Any information about you will be kept confidential. All of the data will be aggregated and no individual will be identifiable from the aggregated results. The data, with no personal identifiers, collected from this study will be maintained on a password-protected computer database in a restricted access area of the University. As well, the data will be electronically archived after completion of the study and maintained for two years and then erased.

Should you have any questions about the study, please contact either Edmund Wong (e32wong@uwaterloo.ca), Jinqiu Yang (j223yang@uwaterloo.ca) or Lin Tan (lintan@uwaterloo.ca). Further, if you would like to receive a copy of the results of this study, please contact either investigator.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin at 519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.

====

There will be a total of 15 questions. The steps for answering each question is of the following:

1. Read the marked source code that is highlighted and try to understand it
2. Write down a short description of what you think the code is about
   (without looking at the next question, it is okay if you don't know the answer)
3. Read the provided comment, this comment describes the marked source code that you just read
4. Rate the provided comment on its accuracy, adequacy, conciseness and usefulness.

I had read the above information and would like to participate in the user study. *

◉ I would like to participate into the user study.

- ○ I would not like to participate into the user study.

What is your education level? *

- ○ Undergraduate Student
- ◉ Graduate Student
- ○ Faculty Staff

Do you have experience in the software development industry? *

- ◉ Yes
- ○ No

Do you have experience in Android software development? *

- ○ Yes
- ◉ No

How many years of programming experience do you have? *

- ○ 1
- ○ 2
- ◉ 3
- ○ 4
- ○ 5
- ○ 6
- ○ 7
- ○ 8
- ○ 9
- ○

10+

Tutorial Question

Please read the marked code between "===="

```
private static InetAddress getFirstNonLoopbackAddress(boolean preferIpv4, boolean preferIPv6) throws SocketException {
  ####
  Enumeration en = NetworkInterface.getNetworkInterfaces();
  while (en.hasMoreElements()) {
    NetworkInterface i = (NetworkInterface) en.nextElement();
    for (Enumeration en2 = i.getInetAddresses(); en2.hasMoreElements();) {
      InetAddress addr = (InetAddress) en2.nextElement();
      if (!addr.isLoopbackAddress()) {
        if (addr instanceof Inet4Address) {
          if (preferIPv6) {
            continue;
          }
          return addr;
        }
        if (addr instanceof Inet6Address) {
          if (preferIpv4) {
            continue;
  ####
          }
          return addr;
        }
      }
    }
  }
  return null;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: It obtains the first non loopback address on the network interface. It returns an IPV6 address if desired or and IPV4 address.

Please read this comment: "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q1

Please read the marked code between "===="

```
@SuppressWarnings({"unchecked"})
public static <T> T serializeDeserialize(T o) throws Exception {
  if ( o == null ) {
    return null;
  }

  ====
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  ObjectOutputStream oos = new ObjectOutputStream( baos );
  oos.writeObject( o );
  byte[] buffer = baos.toByteArray();
  ====
  baos.close();

  ByteArrayInputStream bais = new ByteArrayInputStream( buffer );
  ObjectInputStream ois = new ObjectInputStream( bais );
  return (T) ois.readObject();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: It deserializes an object by first converting it to a byte array, and then returns it.

Please read this comment: "Serialize object with outputstream." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
	○	○	○	○	○	Strongly

Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q2

Please read the marked code between "===="

```
final BufferedImage cached = scaledImages.get(d);
if (cached != null) return cached;

// Directly scaling to less than half size would ignore some pixels.
// Prevent that by halving the base image size as often as needed.
while(wNew*2 <= w && hNew*2 <= h) {
    w = (w+1)/2;
    h = (h+1)/2;
    ====
    BufferedImage halved = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = halved.createGraphics();
    // For halving bilinear should most correctly average 2x2 pixels.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    g.drawImage(im, 0, 0, w, h, null);
    g.dispose();
    ====
    im = halved;
}

if(wNew != w || hNew != h) {
    BufferedImage scaled = new BufferedImage(wNew, hNew,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = scaled.createGraphics();
    // Bicubic should give best quality for odd scaling factors.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    g.drawImage(im, 0, 0, wNew, hNew, null);
    g.dispose();
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Continuously scale down the image size using bilinear interpolation until it's within the specified size. Then if the scaling factor was odd, use bicubic interpolation to improve image quality.

Please read this comment: "Creating a scaled version of an image." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q3

Please read the marked code between "===="

```
/**
 * @param node
 * @return
 */
public IDocumentElementNode clone(IDocumentElementNode node) {
  IDocumentElementNode clone = null;
  try {
    // Serialize
    ####
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bout);
    out.writeObject(node);
    out.flush();
    out.close();
    byte[] bytes = bout.toByteArray();
    // Deserialize
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream in = new ObjectInputStream(bin);
    ####
    clone = (IDocumentElementNode) in.readObject();
    in.close();
    // Reconnect
    clone.reconnect(this, fModel);
  } catch (IOException e) {
    clone = null;
  } catch (ClassNotFoundException e) {
    clone = null;
  }

  return clone;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Deserializes an output stream node.

Please read this comment: "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q4

Please read the marked code between "===="

```
/**
 * Deletes a directory or file
 * <p/>
 * Taken from
 * http://forum.java.sun.com/thread.jspa?threadID=470197&messageID=2169110
 * <p/>
 * Author: jfbriere
 *
 * @param file
 */
public static void deleteRecursive(File file) {
    if (file.isDirectory()) {
    ####
        File[] fileArray = file.listFiles();

        if (fileArray != null) {
            for (File aFileArray : fileArray) {
                deleteRecursive(aFileArray);
            }
        }
    }
    file.delete();
    ####
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Recuresively deletes all the files and folders in a directory.

Please read this comment: "Delete a folder with files using Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the

marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q5

Please read the marked code between "===="

```
Map result;
if ( scratch_file_is == null ){
  result = new LightHashMap();
}else{
    // System.out.println( "read cache file " + scratch_file_name + " for " + this );
  ####
  Properties p = new Properties();
  InputStream fis = scratch_file_is;
  try{
    p.load( fis );
    fis.close();
  ####
    scratch_file_is = new FileInputStream( scratch_file_name );
    messages = new LightHashMap();
    messages.putAll( p );
    result = messages;
  }catch( Throwable e ){
    if ( fis != null ){
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Tries to see if the properties of a file can be read.

Please read this comment: "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q6

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  ####
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
      cal.setTime(dateValue);
    cal.set(Calendar.YEAR, year);
    cal.set(Calendar.MONTH, month);
    cal.set(Calendar.DAY_OF_MONTH, day);
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Gets the current time.

Please read this comment: "If you want to extract more human-readable date/time information from that Calendar object, you can do something like." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q7

Please read the marked code between "===="

```
/**
 * Returns the contents of the given file as a char array.
 * When encoding is null, then the platform default one is used
 * @throws IOException if a problem occured reading the file.
 */
public static char[] getFileCharContent(File file, String encoding) throws IOException {
  ####
  InputStream stream = null;
  try {
    stream = new FileInputStream(file);
    return getInputStreamAsCharArray(stream, (int) file.length(), encoding);
  } finally {
    if (stream != null) {
      try {
        stream.close();
      } catch (IOException e) {
  ####
        // ignore
      }
    }
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Tries to see if the file can be read.

Please read this comment: "Close an established network connection." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q8

Please read the marked code between "===="

```
public static String getMetaModelSourceAsString(Class<?> clazz) {
  File sourceFile = getMetaModelSourceFileFor( clazz );
  StringBuilder contents = new StringBuilder();

  try {
    BufferedReader input = new BufferedReader( new FileReader( sourceFile ) );
    ####
    try {
      String line;
      /*
          * readLine is a bit quirky :
          * it returns the content of a line MINUS the newline.
          * it returns null only for the END of the stream.
          * it returns an empty String if two newlines appear in a row.
          */
      while ( ( line = input.readLine() ) != null ) {
        contents.append( line );
        contents.append( System.getProperty( "line.separator" ) );
      }
    }
    finally {
      input.close();
    ####
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
  }

  return contents.toString();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Tries to properly read in all the content from the input file.

Please read this comment: "Here is a function to read the file." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1	2	3	4	5
				Strongly

Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q9

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);
  ####

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
```

Please write your comment that describes the functionality of the above marked code segment *

- I do not have an answer

Other: Gets the current date and time down to the millisecond.

Please read this comment: "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q10

Please read the marked code between "===="

```
if ( scratch_file_is == null ){

  result = new LightHashMap();

}else{

    // System.out.println( "read cache file " + scratch_file_name + " for " + this );

  ####
  Properties p = new Properties();

  InputStream fis = scratch_file_is;

  try{

    p.load( fis );

    fis.close();
    ####
    scratch_file_is = new FileInputStream( scratch_file_name );

    messages = new LightHashMap();

    messages.putAll( p );

    result = messages;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Tries to read the properties of a file.

Please read this comment: "try to load a Properties object first." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q11

Please read the marked code between "===="

```
private int getBitmapSampleSize(BitmapFactory.Options options, int idealWidth, int idealHeight) {
    int width = options.outWidth;
    ####
    int height = options.outHeight;
    int inSampleSize = 1;
    if (height > idealHeight || width > idealWidth) {
        if (width > height) {
            inSampleSize = Math.round((float)height / (float)idealHeight);
        } else {
            inSampleSize = Math.round((float)width / (float)idealWidth);
    ###
        }
    }
    return inSampleSize;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Determines the number of sample Bitmaps in the input.

Please read this comment: "Calculate sample size." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q12

Please read the marked code between "===="

```
public void checkExternalStorage(){
  #####
  String state = Environment.getExternalStorageState();
  if(Environment.MEDIA_MOUNTED.equals(state)){
    // ok
  } else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    AccessibleToast.makeText(this, R.string.sd_mounted_ro, Toast.LENGTH_LONG).show();
  } else {
  ######
    AccessibleToast.makeText(this, R.string.sd_unmounted, Toast.LENGTH_LONG).show();
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: If the mounted media is in a read only state, read in a toast of a specified length.

Please read this comment: "Get path to secondary external directory for Camera files." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q13

Please read the marked code between "===="

```
/**
 * Returns true if the given Activity has hardware acceleration enabled
 * in its manifest, or in its foreground window.
 *
 * TODO(husky): Remove when initialize() is refactored (see TODO there)
 * TODO(dtrainor) This is still used by other classes.  Make sure to pull some version of this
 * out before removing it.
 */
public static boolean hasHardwareAcceleration(Activity activity) {
    // Has HW acceleration been enabled manually in the current window?
    #####
    Window window = activity.getWindow();
    if (window != null) {
        if ((window.getAttributes().flags
                & WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    }

    // Has HW acceleration been enabled in the manifest?
    try {
        ActivityInfo info = activity.getPackageManager().getActivityInfo(
                activity.getComponentName(), 0);
        if ((info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {
        Log.e("Chrome", "getActivityInfo(self) should not fail");
    ####
    }

    return false;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Determines if hardware acceleration is enabled by checking the current window or the system settings.

Please read this comment: "Detect Hardware Acceleration at Runtime." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q14

Please read the marked code between "===="

```
final void append(String itemID, String source, String[] newTexts, String linkURL) {

  StringBuilder newTextCombined = new StringBuilder();

  if (source != null) {
    newTextCombined.append(source).append(" : ");
  }

  int linkStart = newTextCombined.length();

  ####
  boolean first = true;
  for (String newText : newTexts) {
    if (first) {
      newTextCombined.append(newText);
      first = false;
    } else {
      newTextCombined.append(" [");
      newTextCombined.append(newText);
  ###
      newTextCombined.append(']');
    }
  }

  int linkEnd = newTextCombined.length();

  String newText = newTextCombined.toString();
  Spannable content = new SpannableString(newText + "\n\n");
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Appends all of the text to create a combined string.

Please read this comment: "Is often best to use StringBuilder to concatenate strings." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q15

Please read the marked code between "===="

```
public View getView(int position, View convertView, ViewGroup parent)
{
  ####
  View currView;
  if(convertView == null)
  {
    LayoutInflater li = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    currView = li.inflate(R.layout.icon, null);
  }
  else
  {
    currView = convertView;
  }

  TextView tv = (TextView) currView.findViewById(R.id.icon_text);
  tv.setText("" + position);
  ImageView iv = (ImageView) currView.findViewById(R.id.icon_image);
  ####
  iv.setImageResource(Icons.iconToResId(position));

  return currView;
 }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Changes the current view, and replaces the text and images.

Please read this comment: "In the code example under the referred link the author sets values for the view only at creation time, so each time the framework is reusing the view, it has the same properties." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

1	2	3	4	5

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Post Study

All the comments that were shown were generated using an automated tool. Do you think it is helpful to have comments like that? *

- ○ Yes
- ○ No
- ◉ Maybe
- ○ Other: _____

Will you use the automated tool if it is made available? *

- ○ Yes
- ○ No
- ◉

Maybe

Other: .....

AutoComment User Study

Information Letter for Quality of Source Code Comments

You are invited to participate in a research study conducted by Edmund Wong and Jinqiu Yang, under the supervision of Prof. Lin Tan of the Electrical and Computer Engineering Department at University of Waterloo, Canada. The objective of the research study is to assess the quality of source code comments.

If you decide to volunteer, you will be asked to complete a 1 hour session. 10 minutes will be spent on training, and the remainder on some tasks we ask you to perform. As a participant, you would be asked to write comments for source code segments. After that, we will provide you with comments that describe the given code segments, and you will rate the given comments' accuracy, adequacy, conciseness and usefulness based on a five-point scale. An example question sheet will be provided prior to the study.

Participants must have at least one year of programming experience in Java. Participation in this study is voluntary. You may decline to answer any questions that you do not wish to answer and you can withdraw your participation at any time by advising the researcher. There are no known or anticipated risk from participating in this study. No direct benefit is anticipated from this study.

This is an in-person study that will take approximately 1 hour at DC 3573. In appreciation, we will remunerate you $10. If you withdraw participation you will receive $5 per half hour. The amount received is taxable. It is your responsibility to report the amount received for income tax purposes.

Any information about you will be kept confidential. All of the data will be aggregated and no individual will be identifiable from the aggregated results. The data, with no personal identifiers, collected from this study will be maintained on a password-protected computer database in a restricted access area of the University. As well, the data will be electronically archived after completion of the study and maintained for two years and then erased.

Should you have any questions about the study, please contact either Edmund Wong (e32wong@uwaterloo.ca), Jinqiu Yang (j223yang@uwaterloo.ca) or Lin Tan (lintan@uwaterloo.ca). Further, if you would like to receive a copy of the results of this study, please contact either investigator.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin at 519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.

====

There will be a total of 15 questions. The steps for answering each question is of the following:

1. Read the marked source code that is highlighted and try to understand it
2. Write down a short description of what you think the code is about
   (without looking at the next question, it is okay if you don't know the answer)
3. Read the provided comment, this comment describes the marked source code that you just read
4. Rate the provided comment on its accuracy, adequacy, conciseness and usefulness.

I had read the above information and would like to participate in the user study. *

◉ I would like to participate into the user study.

- ○ I would not like to participate into the user study.

What is your education level? *

- ◉ Undergraduate Student
- ○ Graduate Student
- ○ Faculty Staff

Do you have experience in the software development industry? *

- ◉ Yes
- ○ No

Do you have experience in Android software development? *

- ◉ Yes
- ○ No

How many years of programming experience do you have? *

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5
- ○ 6
- ○ 7
- ◉ 8
- ○ 9
- ○

10+

Tutorial Question

Please read the marked code between "===="

```
private static InetAddress getFirstNonLoopbackAddress(boolean preferIpv4, boolean preferIPv6) throws SocketException {
  ####
  Enumeration en = NetworkInterface.getNetworkInterfaces();
  while (en.hasMoreElements()) {
    NetworkInterface i = (NetworkInterface) en.nextElement();
    for (Enumeration en2 = i.getInetAddresses(); en2.hasMoreElements();) {
      InetAddress addr = (InetAddress) en2.nextElement();
      if (!addr.isLoopbackAddress()) {
        if (addr instanceof Inet4Address) {
          if (preferIPv6) {
            continue;
          }
          return addr;
        }
        if (addr instanceof Inet6Address) {
          if (preferIpv4) {
            continue;
  ####
          }
          return addr;
        }
      }
    }
  }
  return null;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: returns the fist valid ip address of the prefered standard (IPv4 or IPv6) provided by the getNetworkInterfaces and getInetAddresses functions

Please read this comment: "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q1

Please read the marked code between "===="

```
@SuppressWarnings({"unchecked"})
public static <T> T serializeDeserialize(T o) throws Exception {
  if ( o == null ) {
    return null;
  }

  ====
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  ObjectOutputStream oos = new ObjectOutputStream( baos );
  oos.writeObject( o );
  byte[] buffer = baos.toByteArray();
  ====
  baos.close();

  ByteArrayInputStream bais = new ByteArrayInputStream( buffer );
  ObjectInputStream ois = new ObjectInputStream( bais );
  return (T) ois.readObject();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: initializes a byte array output stream and object output stream, then writes the input object o to the output stream with a buffer at the end.

Please read this comment: "Serialize object with outputstream." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

1	2	3	4	5

Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q2

Please read the marked code between "===="

```
final BufferedImage cached = scaledImages.get(d);
if (cached != null) return cached;

// Directly scaling to less than half size would ignore some pixels.
// Prevent that by halving the base image size as often as needed.
while(wNew*2 <= w && hNew*2 <= h) {
    w = (w+1)/2;
    h = (h+1)/2;
    ====
    BufferedImage halved = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = halved.createGraphics();
    // For halving bilinear should most correctly average 2x2 pixels.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    g.drawImage(im, 0, 0, w, h, null);
    g.dispose();
    ====
    im = halved;
}

if(wNew != w || hNew != h) {
    BufferedImage scaled = new BufferedImage(wNew, hNew,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = scaled.createGraphics();
    // Bicubic should give best quality for odd scaling factors.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    g.drawImage(im, 0, 0, wNew, hNew, null);
    g.dispose();
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: scale input image to half of its original size

Please read this comment: "Creating a scaled version of an image." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q3

Please read the marked code between "===="

```
/**
 * @param node
 * @return
 */
public IDocumentElementNode clone(IDocumentElementNode node) {
  IDocumentElementNode clone = null;
  try {
    // Serialize
    ####
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bout);
    out.writeObject(node);
    out.flush();
    out.close();
    byte[] bytes = bout.toByteArray();
    // Deserialize
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream in = new ObjectInputStream(bin);
    ####
    clone = (IDocumentElementNode) in.readObject();
    in.close();
    // Reconnect
    clone.reconnect(this, fModel);
  } catch (IOException e) {
    clone = null;
  } catch (ClassNotFoundException e) {
    clone = null;
  }

  return clone;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: writes node object to output stream then creates new input stream

Please read this comment: "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q4

Please read the marked code between "===="

```
/**
 * Deletes a directory or file
 * <p/>
 * Taken from
 * http://forum.java.sun.com/thread.jspa?threadID=470197&messageID=2169110
 * <p/>
 * Author: jfbriere
 *
 * @param file
 */
public static void deleteRecursive(File file) {
    if (file.isDirectory()) {
    ####
        File[] fileArray = file.listFiles();

        if (fileArray != null) {
            for (File aFileArray : fileArray) {
                deleteRecursive(aFileArray);
            }
        }
    }
    file.delete();
    ####
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: deletes a file or directory and all sub-directories

Please read this comment: "Delete a folder with files using Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the

marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q5

Please read the marked code between "===="

```
Map result;
if ( scratch_file_is == null ){
  result = new LightHashMap();
}else{
    // System.out.println( "read cache file " + scratch_file_name + " for " + this );
  ####
  Properties p = new Properties();
  InputStream fis = scratch_file_is;
  try{
    p.load( fis );
    fis.close();
  ####
    scratch_file_is = new FileInputStream( scratch_file_name );
    messages = new LightHashMap();
    messages.putAll( p );
    result = messages;
  }catch( Throwable e ){
    if ( fis != null ){
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: attempt to load the properties of scratch_file_is

Please read this comment: "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q6

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  ####
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
      cal.setTime(dateValue);
    cal.set(Calendar.YEAR, year);
    cal.set(Calendar.MONTH, month);
    cal.set(Calendar.DAY_OF_MONTH, day);
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: set time variables using cal.get() functions

Please read this comment: "If you want to extract more human-readable date/time information from that Calendar object, you can do something like." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q7

Please read the marked code between "===="

```
/**
 * Returns the contents of the given file as a char array.
 * When encoding is null, then the platform default one is used
 * @throws IOException if a problem occured reading the file.
 */
public static char[] getFileCharContent(File file, String encoding) throws IOException {
  ####
  InputStream stream = null;
  try {
    stream = new FileInputStream(file);
    return getInputStreamAsCharArray(stream, (int) file.length(), encoding);
  } finally {
    if (stream != null) {
      try {
        stream.close();
      } catch (IOException e) {
  ####
        // ignore
      }
    }
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: attempt to open new input stream from file and catch IOExceptions

Please read this comment: "Close an established network connection." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q8

Please read the marked code between "===="

```
public static String getMetaModelSourceAsString(Class<?> clazz) {
  File sourceFile = getMetaModelSourceFileFor( clazz );
  StringBuilder contents = new StringBuilder();

  try {
    BufferedReader input = new BufferedReader( new FileReader( sourceFile ) );
    ####
    try {
      String line;
      /*
          * readLine is a bit quirky :
          * it returns the content of a line MINUS the newline.
          * it returns null only for the END of the stream.
          * it returns an empty String if two newlines appear in a row.
          */
      while ( ( line = input.readLine() ) != null ) {
        contents.append( line );
        contents.append( System.getProperty( "line.separator" ) );
      }
    }
    finally {
      input.close();
    ####
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
  }

  return contents.toString();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: writes the contents of input to contents then closes input

Please read this comment: "Here is a function to read the file." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1	2	3	4	5
				Strongly

Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q9

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);
  ####

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
```

Please write your comment that describes the functionality of the above marked code segment *

- I do not have an answer

Other: set time variables using cal.get() functions

Please read this comment: "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q10

Please read the marked code between "===="

```
if ( scratch_file_is == null ){

  result = new LightHashMap();

}else{

    // System.out.println( "read cache file " + scratch_file_name + " for " + this );

  ####
  Properties p = new Properties();

  InputStream fis = scratch_file_is;

  try{

    p.load( fis );

    fis.close();
    ####
    scratch_file_is = new FileInputStream( scratch_file_name );

    messages = new LightHashMap();

    messages.putAll( p );

    result = messages;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: load properties from scratch_file_is

Please read this comment: "try to load a Properties object first." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q11

Please read the marked code between "===="

```
private int getBitmapSampleSize(BitmapFactory.Options options, int idealWidth, int idealHeight) {
    int width = options.outWidth;
    ####
    int height = options.outHeight;
    int inSampleSize = 1;
    if (height > idealHeight || width > idealWidth) {
        if (width > height) {
            inSampleSize = Math.round((float)height / (float)idealHeight);
        } else {
            inSampleSize = Math.round((float)width / (float)idealWidth);
    ###
        }
    }
    return inSampleSize;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: if width or height exceed ideal dimensions, calculate new sample size by dividing the smalled dimension by the larger dimension

Please read this comment: "Calculate sample size." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q12

Please read the marked code between "===="

```
public void checkExternalStorage(){
  #####
  String state = Environment.getExternalStorageState();
  if(Environment.MEDIA_MOUNTED.equals(state)){
    // ok
  } else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    AccessibleToast.makeText(this, R.string.sd_mounted_ro, Toast.LENGTH_LONG).show();
  } else {
  ####
    AccessibleToast.makeText(this, R.string.sd_unmounted, Toast.LENGTH_LONG).show();
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: check the state of an external storage system

Please read this comment: "Get path to secondary external directory for Camera files." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q13

Please read the marked code between "===="

```
/**
 * Returns true if the given Activity has hardware acceleration enabled
 * in its manifest, or in its foreground window.
 *
 * TODO(husky): Remove when initialize() is refactored (see TODO there)
 * TODO(dtrainor) This is still used by other classes.  Make sure to pull some version of this
 * out before removing it.
 */
public static boolean hasHardwareAcceleration(Activity activity) {
    // Has HW acceleration been enabled manually in the current window?
    #####
    Window window = activity.getWindow();
    if (window != null) {
        if ((window.getAttributes().flags
                & WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    }

    // Has HW acceleration been enabled in the manifest?
    try {
        ActivityInfo info = activity.getPackageManager().getActivityInfo(
                activity.getComponentName(), 0);
        if ((info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {
        Log.e("Chrome", "getActivityInfo(self) should not fail");
    ####
    }

    return false;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: detect whether hardware acceleration is available for the input activity

Please read this comment: "Detect Hardware Acceleration at Runtime." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q14

Please read the marked code between "===="

```
final void append(String itemID, String source, String[] newTexts, String linkURL) {

  StringBuilder newTextCombined = new StringBuilder();

  if (source != null) {
    newTextCombined.append(source).append(" : ");
  }

  int linkStart = newTextCombined.length();

  ####
  boolean first = true;
  for (String newText : newTexts) {
    if (first) {
      newTextCombined.append(newText);
      first = false;
    } else {
      newTextCombined.append(" [");
      newTextCombined.append(newText);
  ###
      newTextCombined.append(']');
    }
  }

  int linkEnd = newTextCombined.length();

  String newText = newTextCombined.toString();
  Spannable content = new SpannableString(newText + "\n\n");
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: append the first string from newTexts array to newTextsCombined, then append all other strings enclosed in square brackets

Please read this comment: "Is often best to use StringBuilder to concatenate strings." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q15

Please read the marked code between "===="

```
public View getView(int position, View convertView, ViewGroup parent)
{
  ####
  View currView;
  if(convertView == null)
  {
    LayoutInflater li = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    currView = li.inflate(R.layout.icon, null);
  }
  else
  {
    currView = convertView;
  }

  TextView tv = (TextView) currView.findViewById(R.id.icon_text);
  tv.setText("" + position);
  ImageView iv = (ImageView) currView.findViewById(R.id.icon_image);
  ####
  iv.setImageResource(Icons.iconToResId(position));

  return currView;
 }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: adds new text to the input view or creats ne view in parent to add text to

Please read this comment: "In the code example under the referred link the author sets values for the view only at creation time, so each time the framework is reusing the view, it has the same properties." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

1	2	3	4	5

Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Post Study

All the comments that were shown were generated using an automated tool. Do you think it is helpful to have comments like that? *

- ○ Yes
- ○ No
- ○ Maybe
- ◉ Other: Depends on quality of comments

Will you use the automated tool if it is made available? *

- ○ Yes
- ○ No
- ○

Maybe

◉ Other: Not at this stage. Most comments were unhelpful. Would use if the comments were more helpful

---

AutoComment User Study

Information Letter for Quality of Source Code Comments

You are invited to participate in a research study conducted by Edmund Wong and Jinqiu Yang, under the supervision of Prof. Lin Tan of the Electrical and Computer Engineering Department at University of Waterloo, Canada. The objective of the research study is to assess the quality of source code comments.

If you decide to volunteer, you will be asked to complete a 1 hour session. 10 minutes will be spent on training, and the remainder on some tasks we ask you to perform. As a participant, you would be asked to write comments for source code segments. After that, we will provide you with comments that describe the given code segments, and you will rate the given comments' accuracy, adequacy, conciseness and usefulness based on a five-point scale. An example question sheet will be provided prior to the study.

Participants must have at least one year of programming experience in Java. Participation in this study is voluntary. You may decline to answer any questions that you do not wish to answer and you can withdraw your participation at any time by advising the researcher. There are no known or anticipated risk from participating in this study. No direct benefit is anticipated from this study.

This is an in-person study that will take approximately 1 hour at DC 3573. In appreciation, we will remunerate you $10. If you withdraw participation you will receive $5 per half hour. The amount received is taxable. It is your responsibility to report the amount received for income tax purposes.

Any information about you will be kept confidential. All of the data will be aggregated and no individual will be identifiable from the aggregated results. The data, with no personal identifiers, collected from this study will be maintained on a password-protected computer database in a restricted access area of the University. As well, the data will be electronically archived after completion of the study and maintained for two years and then erased.

Should you have any questions about the study, please contact either Edmund Wong (e32wong@uwaterloo.ca), Jinqiu Yang (j223yang@uwaterloo.ca) or Lin Tan (lintan@uwaterloo.ca). Further, if you would like to receive a copy of the results of this study, please contact either investigator.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin at 519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.

====

There will be a total of 15 questions. The steps for answering each question is of the following:

1. Read the marked source code that is highlighted and try to understand it
2. Write down a short description of what you think the code is about
   (without looking at the next question, it is okay if you don't know the answer)
3. Read the provided comment, this comment describes the marked source code that you just read
4. Rate the provided comment on its accuracy, adequacy, conciseness and usefulness.

I had read the above information and would like to participate in the user study. *

◉ I would like to participate into the user study.

- ○ I would not like to participate into the user study.

What is your education level? *

- ○ Undergraduate Student
- ◉ Graduate Student
- ○ Faculty Staff

Do you have experience in the software development industry? *

- ◉ Yes
- ○ No

Do you have experience in Android software development? *

- ◉ Yes
- ○ No

How many years of programming experience do you have? *

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ◉ 5
- ○ 6
- ○ 7
- ○ 8
- ○ 9
- ○

10+

Tutorial Question

Please read the marked code between "===="

```
private static InetAddress getFirstNonLoopbackAddress(boolean preferIpv4, boolean preferIPv6) throws SocketException {
  ####
  Enumeration en = NetworkInterface.getNetworkInterfaces();
  while (en.hasMoreElements()) {
    NetworkInterface i = (NetworkInterface) en.nextElement();
    for (Enumeration en2 = i.getInetAddresses(); en2.hasMoreElements();) {
      InetAddress addr = (InetAddress) en2.nextElement();
      if (!addr.isLoopbackAddress()) {
        if (addr instanceof Inet4Address) {
          if (preferIPv6) {
            continue;
          }
          return addr;
        }
        if (addr instanceof Inet6Address) {
          if (preferIpv4) {
            continue;
  ####
          }
          return addr;
        }
      }
    }
  }
  return null;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Returns the first IP address of the preferred type in the network interface

Please read this comment: "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q1

Please read the marked code between "===="

```
@SuppressWarnings({"unchecked"})
public static <T> T serializeDeserialize(T o) throws Exception {
  if ( o == null ) {
    return null;
  }

  ====
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  ObjectOutputStream oos = new ObjectOutputStream( baos );
  oos.writeObject( o );
  byte[] buffer = baos.toByteArray();
  ====
  baos.close();

  ByteArrayInputStream bais = new ByteArrayInputStream( buffer );
  ObjectInputStream ois = new ObjectInputStream( bais );
  return (T) ois.readObject();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Writes the value of o into a new object of output stream

Please read this comment: "Serialize object with outputstream." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
	○	○	○	○	○	Strongly

Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q2

Please read the marked code between "===="

```
final BufferedImage cached = scaledImages.get(d);
if (cached != null) return cached;

// Directly scaling to less than half size would ignore some pixels.
// Prevent that by halving the base image size as often as needed.
while(wNew*2 <= w && hNew*2 <= h) {
    w = (w+1)/2;
    h = (h+1)/2;
    ====
    BufferedImage halved = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = halved.createGraphics();
    // For halving bilinear should most correctly average 2x2 pixels.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    g.drawImage(im, 0, 0, w, h, null);
    g.dispose();
    ====
    im = halved;
}

if(wNew != w || hNew != h) {
    BufferedImage scaled = new BufferedImage(wNew, hNew,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = scaled.createGraphics();
    // Bicubic should give best quality for odd scaling factors.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    g.drawImage(im, 0, 0, wNew, hNew, null);
    g.dispose();
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: construct the image of halved dimensions, render it through bilinear interpolation, and display the result

Please read this comment: "Creating a scaled version of an image." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q3

Please read the marked code between "===="

```
/**
 * @param node
 * @return
 */
public IDocumentElementNode clone(IDocumentElementNode node) {
  IDocumentElementNode clone = null;
  try {
    // Serialize
    ####
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bout);
    out.writeObject(node);
    out.flush();
    out.close();
    byte[] bytes = bout.toByteArray();
    // Deserialize
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream in = new ObjectInputStream(bin);
    ####
    clone = (IDocumentElementNode) in.readObject();
    in.close();
    // Reconnect
    clone.reconnect(this, fModel);
  } catch (IOException e) {
    clone = null;
  } catch (ClassNotFoundException e) {
    clone = null;
  }

  return clone;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Serialize the output object stream and deserialize the input object stream

Please read this comment: "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q4

Please read the marked code between "===="

```
/**
 * Deletes a directory or file
 * <p/>
 * Taken from
 * http://forum.java.sun.com/thread.jspa?threadID=470197&messageID=2169110
 * <p/>
 * Author: jfbriere
 *
 * @param file
 */
public static void deleteRecursive(File file) {
    if (file.isDirectory()) {
    ####
        File[] fileArray = file.listFiles();

        if (fileArray != null) {
            for (File aFileArray : fileArray) {
                deleteRecursive(aFileArray);
            }
        }
    }
    file.delete();
    ####
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Delete all files within the directory through recursion

Please read this comment: "Delete a folder with files using Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the

marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q5

Please read the marked code between "===="

```
Map result;
if ( scratch_file_is == null ){
  result = new LightHashMap();
}else{
    // System.out.println( "read cache file " + scratch_file_name + " for " + this );
  ####
  Properties p = new Properties();
  InputStream fis = scratch_file_is;
  try{
    p.load( fis );
    fis.close();
  ####
    scratch_file_is = new FileInputStream( scratch_file_name );
    messages = new LightHashMap();
    messages.putAll( p );
    result = messages;
  }catch( Throwable e ){
    if ( fis != null ){
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Load scratch_file_is into a new Properties object

Please read this comment: "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q6

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  ####
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
      cal.setTime(dateValue);
    cal.set(Calendar.YEAR, year);
    cal.set(Calendar.MONTH, month);
    cal.set(Calendar.DAY_OF_MONTH, day);
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Retrieve the date and time

Please read this comment: "If you want to extract more human-readable date/time information from that Calendar object, you can do something like." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q7

Please read the marked code between "===="

```
/**
 * Returns the contents of the given file as a char array.
 * When encoding is null, then the platform default one is used
 * @throws IOException if a problem occured reading the file.
 */
public static char[] getFileCharContent(File file, String encoding) throws IOException {
  ####
  InputStream stream = null;
  try {
    stream = new FileInputStream(file);
    return getInputStreamAsCharArray(stream, (int) file.length(), encoding);
  } finally {
    if (stream != null) {
      try {
        stream.close();
      } catch (IOException e) {
  ####
        // ignore
      }
    }
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Return the file input stream as a character array

Please read this comment: "Close an established network connection." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q8

Please read the marked code between "===="

```
public static String getMetaModelSourceAsString(Class<?> clazz) {
  File sourceFile = getMetaModelSourceFileFor( clazz );
  StringBuilder contents = new StringBuilder();

  try {
    BufferedReader input = new BufferedReader( new FileReader( sourceFile ) );
    ####
    try {
      String line;
      /*
          * readLine is a bit quirky :
          * it returns the content of a line MINUS the newline.
          * it returns null only for the END of the stream.
          * it returns an empty String if two newlines appear in a row.
          */
      while ( ( line = input.readLine() ) != null ) {
        contents.append( line );
        contents.append( System.getProperty( "line.separator" ) );
      }
    }
    finally {
      input.close();
    ####
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
  }

  return contents.toString();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Reads the input line-by-line and appends the separator between them

Please read this comment: "Here is a function to read the file." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1	2	3	4	5
Strongly				

Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q9

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);
  ####

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
```

Please write your comment that describes the functionality of the above marked code segment *

- I do not have an answer

Other: Returns the date and time up to millisecond precision

Please read this comment: "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q10

Please read the marked code between "===="

```
if ( scratch_file_is == null ){

  result = new LightHashMap();

}else{

    // System.out.println( "read cache file " + scratch_file_name + " for " + this );

  ####
  Properties p = new Properties();

  InputStream fis = scratch_file_is;

  try{

    p.load( fis );

    fis.close();
    ####
    scratch_file_is = new FileInputStream( scratch_file_name );

    messages = new LightHashMap();

    messages.putAll( p );

    result = messages;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Load scratch_file_is into a new Properties object

Please read this comment: "try to load a Properties object first." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q11

Please read the marked code between "===="

```
private int getBitmapSampleSize(BitmapFactory.Options options, int idealWidth, int idealHeight) {
    int width = options.outWidth;
    ####
    int height = options.outHeight;
    int inSampleSize = 1;
    if (height > idealHeight || width > idealWidth) {
        if (width > height) {
            inSampleSize = Math.round((float)height / (float)idealHeight);
        } else {
            inSampleSize = Math.round((float)width / (float)idealWidth);
    ###
        }
    }
    return inSampleSize;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Returns the scaling factor of the sample compared to the ideal dimensions

Please read this comment: "Calculate sample size." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q12

Please read the marked code between "===="

```
public void checkExternalStorage(){
  #####
  String state = Environment.getExternalStorageState();
  if(Environment.MEDIA_MOUNTED.equals(state)){
    // ok
  } else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    AccessibleToast.makeText(this, R.string.sd_mounted_ro, Toast.LENGTH_LONG).show();
  } else {
  ######
    AccessibleToast.makeText(this, R.string.sd_unmounted, Toast.LENGTH_LONG).show();
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: If the external storage is read-only, then report it

Please read this comment: "Get path to secondary external directory for Camera files." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q13

Please read the marked code between "===="

```
/**
 * Returns true if the given Activity has hardware acceleration enabled
 * in its manifest, or in its foreground window.
 *
 * TODO(husky): Remove when initialize() is refactored (see TODO there)
 * TODO(dtrainor) This is still used by other classes.  Make sure to pull some version of this
 * out before removing it.
 */
public static boolean hasHardwareAcceleration(Activity activity) {
    // Has HW acceleration been enabled manually in the current window?
    #####
    Window window = activity.getWindow();
    if (window != null) {
        if ((window.getAttributes().flags
                & WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    }

    // Has HW acceleration been enabled in the manifest?
    try {
        ActivityInfo info = activity.getPackageManager().getActivityInfo(
                activity.getComponentName(), 0);
        if ((info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {
        Log.e("Chrome", "getActivityInfo(self) should not fail");
    ####
    }

    return false;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Checks whether hardware acceleration is activated on the current window, or in the manifest

Please read this comment: "Detect Hardware Acceleration at Runtime." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q14

Please read the marked code between "===="

```
final void append(String itemID, String source, String[] newTexts, String linkURL) {

  StringBuilder newTextCombined = new StringBuilder();

  if (source != null) {
    newTextCombined.append(source).append(" : ");
  }

  int linkStart = newTextCombined.length();

  ####
  boolean first = true;
  for (String newText : newTexts) {
    if (first) {
      newTextCombined.append(newText);
      first = false;
    } else {
      newTextCombined.append(" [");
      newTextCombined.append(newText);
  ###
      newTextCombined.append(']');
    }
  }

  int linkEnd = newTextCombined.length();

  String newText = newTextCombined.toString();
  Spannable content = new SpannableString(newText + "\n\n");
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Combine all the texts using [ ] for the second text onwards

Please read this comment: "Is often best to use StringBuilder to concatenate strings." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q15

Please read the marked code between "===="

```
public View getView(int position, View convertView, ViewGroup parent)
{
  ####
  View currView;
  if(convertView == null)
  {
    LayoutInflater li = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    currView = li.inflate(R.layout.icon, null);
  }
  else
  {
    currView = convertView;
  }

  TextView tv = (TextView) currView.findViewById(R.id.icon_text);
  tv.setText("" + position);
  ImageView iv = (ImageView) currView.findViewById(R.id.icon_image);
  ####
  iv.setImageResource(Icons.iconToResId(position));

  return currView;
 }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Construct the currView object from convertView and set its text view and image view

Please read this comment: "In the code example under the referred link the author sets values for the view only at creation time, so each time the framework is reusing the view, it has the same properties." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Post Study

All the comments that were shown were generated using an automated tool. Do you think it is helpful to have comments like that? *

- ○ Yes
- ○ No
- ◉ Maybe
- ○ Other: _____

Will you use the automated tool if it is made available? *

- ○ Yes
- ◉ No

- ○ Maybe
- ○ Other: _____

AutoComment User Study

Information Letter for Quality of Source Code Comments

You are invited to participate in a research study conducted by Edmund Wong and Jinqiu Yang, under the supervision of Prof. Lin Tan of the Electrical and Computer Engineering Department at University of Waterloo, Canada. The objective of the research study is to assess the quality of source code comments.

If you decide to volunteer, you will be asked to complete a 1 hour session. 10 minutes will be spent on training, and the remainder on some tasks we ask you to perform. As a participant, you would be asked to write comments for source code segments. After that, we will provide you with comments that describe the given code segments, and you will rate the given comments' accuracy, adequacy, conciseness and usefulness based on a five-point scale. An example question sheet will be provided prior to the study.

Participants must have at least one year of programming experience in Java. Participation in this study is voluntary. You may decline to answer any questions that you do not wish to answer and you can withdraw your participation at any time by advising the researcher. There are no known or anticipated risk from participating in this study. No direct benefit is anticipated from this study.

This is an in-person study that will take approximately 1 hour at DC 3573. In appreciation, we will remunerate you $10. If you withdraw participation you will receive $5 per half hour. The amount received is taxable. It is your responsibility to report the amount received for income tax purposes.

Any information about you will be kept confidential. All of the data will be aggregated and no individual will be identifiable from the aggregated results. The data, with no personal identifiers, collected from this study will be maintained on a password-protected computer database in a restricted access area of the University. As well, the data will be electronically archived after completion of the study and maintained for two years and then erased.

Should you have any questions about the study, please contact either Edmund Wong (e32wong@uwaterloo.ca), Jinqiu Yang (j223yang@uwaterloo.ca) or Lin Tan (lintan@uwaterloo.ca). Further, if you would like to receive a copy of the results of this study, please contact either investigator.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin at 519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.

====

There will be a total of 15 questions. The steps for answering each question is of the following:

1. Read the marked source code that is highlighted and try to understand it
2. Write down a short description of what you think the code is about
   (without looking at the next question, it is okay if you don't know the answer)
3. Read the provided comment, this comment describes the marked source code that you just read
4. Rate the provided comment on its accuracy, adequacy, conciseness and usefulness.

I had read the above information and would like to participate in the user study. *

◉ I would like to participate into the user study.

- ○ I would not like to participate into the user study.

What is your education level? *

- ○ Undergraduate Student
- ◉ Graduate Student
- ○ Faculty Staff

Do you have experience in the software development industry? *

- ◉ Yes
- ○ No

Do you have experience in Android software development? *

- ◉ Yes
- ○ No

How many years of programming experience do you have? *

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5
- ○ 6
- ○ 7
- ○ 8
- ◉ 9
- ○

10+

Tutorial Question

Please read the marked code between "===="

```
private static InetAddress getFirstNonLoopbackAddress(boolean preferIpv4, boolean preferIPv6) throws SocketException {
  ####
  Enumeration en = NetworkInterface.getNetworkInterfaces();
  while (en.hasMoreElements()) {
    NetworkInterface i = (NetworkInterface) en.nextElement();
    for (Enumeration en2 = i.getInetAddresses(); en2.hasMoreElements();) {
      InetAddress addr = (InetAddress) en2.nextElement();
      if (!addr.isLoopbackAddress()) {
        if (addr instanceof Inet4Address) {
          if (preferIPv6) {
            continue;
          }
          return addr;
        }
        if (addr instanceof Inet6Address) {
          if (preferIpv4) {
            continue;
  ####
          }
          return addr;
        }
      }
    }
  }
  return null;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Finds and returns an IPv4 non-loopback address

Please read this comment: "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q1

Please read the marked code between "===="

```
@SuppressWarnings({"unchecked"})
public static <T> T serializeDeserialize(T o) throws Exception {
  if ( o == null ) {
    return null;
  }

  ====
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  ObjectOutputStream oos = new ObjectOutputStream( baos );
  oos.writeObject( o );
  byte[] buffer = baos.toByteArray();
  ====
  baos.close();

  ByteArrayInputStream bais = new ByteArrayInputStream( buffer );
  ObjectInputStream ois = new ObjectInputStream( bais );
  return (T) ois.readObject();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Serializes the given object

Please read this comment: "Serialize object with outputstream." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
						Strongly

Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q2

Please read the marked code between "===="

```
final BufferedImage cached = scaledImages.get(d);
if (cached != null) return cached;

// Directly scaling to less than half size would ignore some pixels.
// Prevent that by halving the base image size as often as needed.
while(wNew*2 <= w && hNew*2 <= h) {
    w = (w+1)/2;
    h = (h+1)/2;
    ====
    BufferedImage halved = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = halved.createGraphics();
    // For halving bilinear should most correctly average 2x2 pixels.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    g.drawImage(im, 0, 0, w, h, null);
    g.dispose();
    ====
    im = halved;
}

if(wNew != w || hNew != h) {
    BufferedImage scaled = new BufferedImage(wNew, hNew,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = scaled.createGraphics();
    // Bicubic should give best quality for odd scaling factors.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    g.drawImage(im, 0, 0, wNew, hNew, null);
    g.dispose();
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Draws the image

Please read this comment: "Creating a scaled version of an image." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q3

Please read the marked code between "===="

```
/**
 * @param node
 * @return
 */
public IDocumentElementNode clone(IDocumentElementNode node) {
  IDocumentElementNode clone = null;
  try {
    // Serialize
    ####
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bout);
    out.writeObject(node);
    out.flush();
    out.close();
    byte[] bytes = bout.toByteArray();
    // Deserialize
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream in = new ObjectInputStream(bin);
    ####
    clone = (IDocumentElementNode) in.readObject();
    in.close();
    // Reconnect
    clone.reconnect(this, fModel);
  } catch (IOException e) {
    clone = null;
  } catch (ClassNotFoundException e) {
    clone = null;
  }

  return clone;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Serializes an object and creates an input stream on it

Please read this comment: "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q4

Please read the marked code between "===="

```
/**
 * Deletes a directory or file
 * <p/>
 * Taken from
 * http://forum.java.sun.com/thread.jspa?threadID=470197&messageID=2169110
 * <p/>
 * Author: jfbriere
 *
 * @param file
 */
public static void deleteRecursive(File file) {
    if (file.isDirectory()) {
    ####
        File[] fileArray = file.listFiles();

        if (fileArray != null) {
            for (File aFileArray : fileArray) {
                deleteRecursive(aFileArray);
            }
        }
    }
    file.delete();
    ####
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Deletes a file or directory

Please read this comment: "Delete a folder with files using Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the

marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q5

Please read the marked code between "===="

```
Map result;
if ( scratch_file_is == null ){
  result = new LightHashMap();
}else{
    // System.out.println( "read cache file " + scratch_file_name + " for " + this );
  ####
  Properties p = new Properties();
  InputStream fis = scratch_file_is;
  try{
    p.load( fis );
    fis.close();
  ####
    scratch_file_is = new FileInputStream( scratch_file_name );
    messages = new LightHashMap();
    messages.putAll( p );
    result = messages;
  }catch( Throwable e ){
    if ( fis != null ){
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Reads in properties from a file

Please read this comment: "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q6

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  ####
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
      cal.setTime(dateValue);
    cal.set(Calendar.YEAR, year);
    cal.set(Calendar.MONTH, month);
    cal.set(Calendar.DAY_OF_MONTH, day);
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Gets date and time

Please read this comment: "If you want to extract more human-readable date/time information from that Calendar object, you can do something like." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q7

Please read the marked code between "===="

```
/**
 * Returns the contents of the given file as a char array.
 * When encoding is null, then the platform default one is used
 * @throws IOException if a problem occured reading the file.
 */
public static char[] getFileCharContent(File file, String encoding) throws IOException {
  ####
  InputStream stream = null;
  try {
    stream = new FileInputStream(file);
    return getInputStreamAsCharArray(stream, (int) file.length(), encoding);
  } finally {
    if (stream != null) {
      try {
        stream.close();
      } catch (IOException e) {
  ####
        // ignore
      }
    }
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Reads a file in chars

Please read this comment: "Close an established network connection." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q8

Please read the marked code between "===="

```
public static String getMetaModelSourceAsString(Class<?> clazz) {
  File sourceFile = getMetaModelSourceFileFor( clazz );
  StringBuilder contents = new StringBuilder();

  try {
    BufferedReader input = new BufferedReader( new FileReader( sourceFile ) );
    ####
    try {
      String line;
      /*
          * readLine is a bit quirky :
          * it returns the content of a line MINUS the newline.
          * it returns null only for the END of the stream.
          * it returns an empty String if two newlines appear in a row.
          */
      while ( ( line = input.readLine() ) != null ) {
        contents.append( line );
        contents.append( System.getProperty( "line.separator" ) );
      }
    }
    finally {
      input.close();
    ####
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
  }

  return contents.toString();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Reads a file and inserts a separator after every line.

Please read this comment: "Here is a function to read the file." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1	2	3	4	5
Strongly				

Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q9

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);
  ####

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉

Other: Gets date and time (in ms)

Please read this comment: "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

- (◉) I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q10

Please read the marked code between "===="

```
if ( scratch_file_is == null ){

  result = new LightHashMap();

}else{

    // System.out.println( "read cache file " + scratch_file_name + " for " + this );

  ####
  Properties p = new Properties();

  InputStream fis = scratch_file_is;

  try{

    p.load( fis );

    fis.close();
    ####
    scratch_file_is = new FileInputStream( scratch_file_name );

    messages = new LightHashMap();

    messages.putAll( p );

    result = messages;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Loads in properties from the input

Please read this comment: "try to load a Properties object first." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q11

Please read the marked code between "===="

```
private int getBitmapSampleSize(BitmapFactory.Options options, int idealWidth, int idealHeight) {
    int width = options.outWidth;
    ####
    int height = options.outHeight;
    int inSampleSize = 1;
    if (height > idealHeight || width > idealWidth) {
        if (width > height) {
            inSampleSize = Math.round((float)height / (float)idealHeight);
        } else {
            inSampleSize = Math.round((float)width / (float)idealWidth);
    ###
        }
    }
    return inSampleSize;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Caculates sample size

Please read this comment: "Calculate sample size." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q12

Please read the marked code between "===="

```
public void checkExternalStorage(){
  #####
  String state = Environment.getExternalStorageState();
  if(Environment.MEDIA_MOUNTED.equals(state)){
    // ok
  } else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    AccessibleToast.makeText(this, R.string.sd_mounted_ro, Toast.LENGTH_LONG).show();
  } else {
  ######
    AccessibleToast.makeText(this, R.string.sd_unmounted, Toast.LENGTH_LONG).show();
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Checks external storage state

Please read this comment: "Get path to secondary external directory for Camera files." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q13

Please read the marked code between "===="

```
/**
 * Returns true if the given Activity has hardware acceleration enabled
 * in its manifest, or in its foreground window.
 *
 * TODO(husky): Remove when initialize() is refactored (see TODO there)
 * TODO(dtrainor) This is still used by other classes.  Make sure to pull some version of this
 * out before removing it.
 */
public static boolean hasHardwareAcceleration(Activity activity) {
    // Has HW acceleration been enabled manually in the current window?
    #####
    Window window = activity.getWindow();
    if (window != null) {
        if ((window.getAttributes().flags
                & WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    }

    // Has HW acceleration been enabled in the manifest?
    try {
        ActivityInfo info = activity.getPackageManager().getActivityInfo(
                activity.getComponentName(), 0);
        if ((info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {
        Log.e("Chrome", "getActivityInfo(self) should not fail");
    ####
    }

    return false;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Checks if hardware acceleration takes effect

Please read this comment: "Detect Hardware Acceleration at Runtime." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q14

Please read the marked code between "===="

```
final void append(String itemID, String source, String[] newTexts, String linkURL) {

  StringBuilder newTextCombined = new StringBuilder();

  if (source != null) {
    newTextCombined.append(source).append(" : ");
  }

  int linkStart = newTextCombined.length();

  ####
  boolean first = true;
  for (String newText : newTexts) {
    if (first) {
      newTextCombined.append(newText);
      first = false;
    } else {
      newTextCombined.append(" [");
      newTextCombined.append(newText);
  ###
      newTextCombined.append(']');
    }
  }

  int linkEnd = newTextCombined.length();

  String newText = newTextCombined.toString();
  Spannable content = new SpannableString(newText + "\n\n");
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Reformats texts

Please read this comment: "Is often best to use StringBuilder to concatenate strings." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q15

Please read the marked code between "===="

```
public View getView(int position, View convertView, ViewGroup parent)
{
  ####
  View currView;
  if(convertView == null)
  {
    LayoutInflater li = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    currView = li.inflate(R.layout.icon, null);
  }
  else
  {
    currView = convertView;
  }

  TextView tv = (TextView) currView.findViewById(R.id.icon_text);
  tv.setText("" + position);
  ImageView iv = (ImageView) currView.findViewById(R.id.icon_image);
  ####
  iv.setImageResource(Icons.iconToResId(position));

  return currView;
 }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Sets text and image views

Please read this comment: "In the code example under the referred link the author sets values for the view only at creation time, so each time the framework is reusing the view, it has the same properties." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

1	2	3	4	5

Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Post Study

All the comments that were shown were generated using an automated tool. Do you think it is helpful to have comments like that? *

- ○ Yes
- ○ No
- ◉ Maybe
- ○ Other: _____

Will you use the automated tool if it is made available? *

- ○ Yes
- ○ No
- ◉

Maybe

Other: _____

AutoComment User Study

Information Letter for Quality of Source Code Comments

You are invited to participate in a research study conducted by Edmund Wong and Jinqiu Yang, under the supervision of Prof. Lin Tan of the Electrical and Computer Engineering Department at University of Waterloo, Canada. The objective of the research study is to assess the quality of source code comments.

If you decide to volunteer, you will be asked to complete a 1 hour session. 10 minutes will be spent on training, and the remainder on some tasks we ask you to perform. As a participant, you would be asked to write comments for source code segments. After that, we will provide you with comments that describe the given code segments, and you will rate the given comments' accuracy, adequacy, conciseness and usefulness based on a five-point scale. An example question sheet will be provided prior to the study.

Participants must have at least one year of programming experience in Java. Participation in this study is voluntary. You may decline to answer any questions that you do not wish to answer and you can withdraw your participation at any time by advising the researcher. There are no known or anticipated risk from participating in this study. No direct benefit is anticipated from this study.

This is an in-person study that will take approximately 1 hour at DC 3573. In appreciation, we will remunerate you $10. If you withdraw participation you will receive $5 per half hour. The amount received is taxable. It is your responsibility to report the amount received for income tax purposes.

Any information about you will be kept confidential. All of the data will be aggregated and no individual will be identifiable from the aggregated results. The data, with no personal identifiers, collected from this study will be maintained on a password-protected computer database in a restricted access area of the University. As well, the data will be electronically archived after completion of the study and maintained for two years and then erased.

Should you have any questions about the study, please contact either Edmund Wong (e32wong@uwaterloo.ca), Jinqiu Yang (j223yang@uwaterloo.ca) or Lin Tan (lintan@uwaterloo.ca). Further, if you would like to receive a copy of the results of this study, please contact either investigator.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin at 519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.

====

There will be a total of 15 questions. The steps for answering each question is of the following:

1. Read the marked source code that is highlighted and try to understand it
2. Write down a short description of what you think the code is about
   (without looking at the next question, it is okay if you don't know the answer)
3. Read the provided comment, this comment describes the marked source code that you just read
4. Rate the provided comment on its accuracy, adequacy, conciseness and usefulness.

I had read the above information and would like to participate in the user study. *

◉ I would like to participate into the user study.

- ○ I would not like to participate into the user study.

What is your education level? *

- ○ Undergraduate Student
- ◉ Graduate Student
- ○ Faculty Staff

Do you have experience in the software development industry? *

- ◉ Yes
- ○ No

Do you have experience in Android software development? *

- ○ Yes
- ◉ No

How many years of programming experience do you have? *

- ○ 1
- ○ 2
- ○ 3
- ◉ 4
- ○ 5
- ○ 6
- ○ 7
- ○ 8
- ○ 9
- ○

10+

Tutorial Question

Please read the marked code between "===="

```
private static InetAddress getFirstNonLoopbackAddress(boolean preferIpv4, boolean preferIPv6) throws SocketException {
  ####
  Enumeration en = NetworkInterface.getNetworkInterfaces();
  while (en.hasMoreElements()) {
    NetworkInterface i = (NetworkInterface) en.nextElement();
    for (Enumeration en2 = i.getInetAddresses(); en2.hasMoreElements();) {
      InetAddress addr = (InetAddress) en2.nextElement();
      if (!addr.isLoopbackAddress()) {
        if (addr instanceof Inet4Address) {
          if (preferIPv6) {
            continue;
          }
          return addr;
        }
        if (addr instanceof Inet6Address) {
          if (preferIpv4) {
            continue;
  ####
          }
          return addr;
        }
      }
    }
  }
  return null;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: get the first non-loopback Ip with the prefer of IPV4 or IPV6

Please read this comment: "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q1

Please read the marked code between "===="

```
@SuppressWarnings({"unchecked"})
public static <T> T serializeDeserialize(T o) throws Exception {
  if ( o == null ) {
    return null;
  }

  ====
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  ObjectOutputStream oos = new ObjectOutputStream( baos );
  oos.writeObject( o );
  byte[] buffer = baos.toByteArray();
  ====
  baos.close();

  ByteArrayInputStream bais = new ByteArrayInputStream( buffer );
  ObjectInputStream ois = new ObjectInputStream( bais );
  return (T) ois.readObject();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Create a byte array buffer of serialized Object o

Please read this comment: "Serialize object with outputstream." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
						Strongly

Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q2

Please read the marked code between "===="

```
final BufferedImage cached = scaledImages.get(d);
if (cached != null) return cached;

// Directly scaling to less than half size would ignore some pixels.
// Prevent that by halving the base image size as often as needed.
while(wNew*2 <= w && hNew*2 <= h) {
    w = (w+1)/2;
    h = (h+1)/2;
    ====
    BufferedImage halved = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = halved.createGraphics();
    // For halving bilinear should most correctly average 2x2 pixels.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    g.drawImage(im, 0, 0, w, h, null);
    g.dispose();
    ====
    im = halved;
}

if(wNew != w || hNew != h) {
    BufferedImage scaled = new BufferedImage(wNew, hNew,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = scaled.createGraphics();
    // Bicubic should give best quality for odd scaling factors.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    g.drawImage(im, 0, 0, wNew, hNew, null);
    g.dispose();
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: havled and draw img im

Please read this comment: "Creating a scaled version of an image." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q3

Please read the marked code between "===="

```
/**
 * @param node
 * @return
 */
public IDocumentElementNode clone(IDocumentElementNode node) {
  IDocumentElementNode clone = null;
  try {
    // Serialize
    ####
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bout);
    out.writeObject(node);
    out.flush();
    out.close();
    byte[] bytes = bout.toByteArray();
    // Deserialize
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream in = new ObjectInputStream(bin);
    ####
    clone = (IDocumentElementNode) in.readObject();
    in.close();
    // Reconnect
    clone.reconnect(this, fModel);
  } catch (IOException e) {
    clone = null;
  } catch (ClassNotFoundException e) {
    clone = null;
  }

  return clone;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Serialize an Object with OutputStream and also providing deserialized Stream

Please read this comment: "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q4

Please read the marked code between "===="

```
/**
 * Deletes a directory or file
 * <p/>
 * Taken from
 * http://forum.java.sun.com/thread.jspa?threadID=470197&messageID=2169110
 * <p/>
 * Author: jfbriere
 *
 * @param file
 */
public static void deleteRecursive(File file) {
    if (file.isDirectory()) {
    ####
        File[] fileArray = file.listFiles();

        if (fileArray != null) {
            for (File aFileArray : fileArray) {
                deleteRecursive(aFileArray);
            }
        }
    }
    file.delete();
    ####
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: recursively delete a file or directory

Please read this comment: "Delete a folder with files using Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the

marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q5

Please read the marked code between "===="

```
Map result;
if ( scratch_file_is == null ){
  result = new LightHashMap();
}else{
    // System.out.println( "read cache file " + scratch_file_name + " for " + this );
  ####
  Properties p = new Properties();
  InputStream fis = scratch_file_is;
  try{
    p.load( fis );
    fis.close();
  ####
    scratch_file_is = new FileInputStream( scratch_file_name );
    messages = new LightHashMap();
    messages.putAll( p );
    result = messages;
  }catch( Throwable e ){
    if ( fis != null ){
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: loading properties of a scrach file

Please read this comment: "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q6

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  ####
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
      cal.setTime(dateValue);
    cal.set(Calendar.YEAR, year);
    cal.set(Calendar.MONTH, month);
    cal.set(Calendar.DAY_OF_MONTH, day);
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: get finer granularity information of a Calendar

Please read this comment: "If you want to extract more human-readable date/time information from that Calendar object, you can do something like." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q7

Please read the marked code between "===="

```
/**
 * Returns the contents of the given file as a char array.
 * When encoding is null, then the platform default one is used
 * @throws IOException if a problem occured reading the file.
 */
public static char[] getFileCharContent(File file, String encoding) throws IOException {
  ####
  InputStream stream = null;
  try {
    stream = new FileInputStream(file);
    return getInputStreamAsCharArray(stream, (int) file.length(), encoding);
  } finally {
    if (stream != null) {
      try {
        stream.close();
      } catch (IOException e) {
  ####
        // ignore
      }
    }
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: open a file as a charArray

Please read this comment: "Close an established network connection." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q8

Please read the marked code between "===="

```
public static String getMetaModelSourceAsString(Class<?> clazz) {
  File sourceFile = getMetaModelSourceFileFor( clazz );
  StringBuilder contents = new StringBuilder();

  try {
    BufferedReader input = new BufferedReader( new FileReader( sourceFile ) );
    ####
    try {
      String line;
      /*
          * readLine is a bit quirky :
          * it returns the content of a line MINUS the newline.
          * it returns null only for the END of the stream.
          * it returns an empty String if two newlines appear in a row.
          */
      while ( ( line = input.readLine() ) != null ) {
        contents.append( line );
        contents.append( System.getProperty( "line.separator" ) );
      }
    }
    finally {
      input.close();
    ####
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
  }

  return contents.toString();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: append the readed line to contents with System line separator

Please read this comment: "Here is a function to read the file." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1	2	3	4	5
				Strongly

Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q9

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);
  ####

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
```

Please write your comment that describes the functionality of the above marked code segment *

- I do not have an answer

Other: get details information of a Canlendar

Please read this comment: "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q10

Please read the marked code between "===="

```
if ( scratch_file_is == null ){

  result = new LightHashMap();

}else{

    // System.out.println( "read cache file " + scratch_file_name + " for " + this );

  ####
  Properties p = new Properties();

  InputStream fis = scratch_file_is;

  try{

    p.load( fis );

    fis.close();
    ####
    scratch_file_is = new FileInputStream( scratch_file_name );

    messages = new LightHashMap();

    messages.putAll( p );

    result = messages;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: loading proerties of scratch file

Please read this comment: "try to load a Properties object first." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q11

Please read the marked code between "===="

```
private int getBitmapSampleSize(BitmapFactory.Options options, int idealWidth, int idealHeight) {
    int width = options.outWidth;
    ####
    int height = options.outHeight;
    int inSampleSize = 1;
    if (height > idealHeight || width > idealWidth) {
        if (width > height) {
            inSampleSize = Math.round((float)height / (float)idealHeight);
        } else {
            inSampleSize = Math.round((float)width / (float)idealWidth);
    ###
        }
    }
    return inSampleSize;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: get inSampleSize based on outWidth and outHeight

Please read this comment: "Calculate sample size." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q12

Please read the marked code between "===="

```
public void checkExternalStorage(){
  #####
  String state = Environment.getExternalStorageState();
  if(Environment.MEDIA_MOUNTED.equals(state)){
    // ok
  } else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    AccessibleToast.makeText(this, R.string.sd_mounted_ro, Toast.LENGTH_LONG).show();
  } else {
  ######
    AccessibleToast.makeText(this, R.string.sd_unmounted, Toast.LENGTH_LONG).show();
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: issue a message if the state of external storage is not MEDIA_MOUNTED

Please read this comment: "Get path to secondary external directory for Camera files." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q13

Please read the marked code between "===="

```
/**
 * Returns true if the given Activity has hardware acceleration enabled
 * in its manifest, or in its foreground window.
 *
 * TODO(husky): Remove when initialize() is refactored (see TODO there)
 * TODO(dtrainor) This is still used by other classes.  Make sure to pull some version of this
 * out before removing it.
 */
public static boolean hasHardwareAcceleration(Activity activity) {
    // Has HW acceleration been enabled manually in the current window?
    #####
    Window window = activity.getWindow();
    if (window != null) {
        if ((window.getAttributes().flags
                & WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    }

    // Has HW acceleration been enabled in the manifest?
    try {
        ActivityInfo info = activity.getPackageManager().getActivityInfo(
                activity.getComponentName(), 0);
        if ((info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {
        Log.e("Chrome", "getActivityInfo(self) should not fail");
    ####
    }

    return false;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: check whether an activity has hardware acceleration

Please read this comment: "Detect Hardware Acceleration at Runtime." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q14

Please read the marked code between "===="

```
final void append(String itemID, String source, String[] newTexts, String linkURL) {

  StringBuilder newTextCombined = new StringBuilder();

  if (source != null) {
    newTextCombined.append(source).append(" : ");
  }

  int linkStart = newTextCombined.length();

  ####
  boolean first = true;
  for (String newText : newTexts) {
    if (first) {
      newTextCombined.append(newText);
      first = false;
    } else {
      newTextCombined.append(" [");
      newTextCombined.append(newText);
  ###
      newTextCombined.append(']');
    }
  }

  int linkEnd = newTextCombined.length();

  String newText = newTextCombined.toString();
  Spannable content = new SpannableString(newText + "\n\n");
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: wrap newText with [] except first one

Please read this comment: "Is often best to use StringBuilder to concatenate strings." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

Q15

Please read the marked code between "===="

```
public View getView(int position, View convertView, ViewGroup parent)
{
  ####
  View currView;
  if(convertView == null)
  {
    LayoutInflater li = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    currView = li.inflate(R.layout.icon, null);
  }
  else
  {
    currView = convertView;
  }

  TextView tv = (TextView) currView.findViewById(R.id.icon_text);
  tv.setText("" + position);
  ImageView iv = (ImageView) currView.findViewById(R.id.icon_image);
  ####
  iv.setImageResource(Icons.iconToResId(position));

  return currView;
 }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: get the TextView and ImageView of the currView

Please read this comment: "In the code example under the referred link the author sets values for the view only at creation time, so each time the framework is reusing the view, it has the same properties." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

1	2	3	4	5

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Post Study

All the comments that were shown were generated using an automated tool. Do you think it is helpful to have comments like that? *

- ○ Yes
- ○ No
- ◉ Maybe
- ○ Other: _____

Will you use the automated tool if it is made available? *

- ○ Yes
- ○ No
- ◉

Maybe

Other: _____

AutoComment User Study

Information Letter for Quality of Source Code Comments

You are invited to participate in a research study conducted by Edmund Wong and Jinqiu Yang, under the supervision of Prof. Lin Tan of the Electrical and Computer Engineering Department at University of Waterloo, Canada. The objective of the research study is to assess the quality of source code comments.

If you decide to volunteer, you will be asked to complete a 1 hour session. 10 minutes will be spent on training, and the remainder on some tasks we ask you to perform. As a participant, you would be asked to write comments for source code segments. After that, we will provide you with comments that describe the given code segments, and you will rate the given comments' accuracy, adequacy, conciseness and usefulness based on a five-point scale. An example question sheet will be provided prior to the study.

Participants must have at least one year of programming experience in Java. Participation in this study is voluntary. You may decline to answer any questions that you do not wish to answer and you can withdraw your participation at any time by advising the researcher. There are no known or anticipated risk from participating in this study. No direct benefit is anticipated from this study.

This is an in-person study that will take approximately 1 hour at DC 3573. In appreciation, we will remunerate you $10. If you withdraw participation you will receive $5 per half hour. The amount received is taxable. It is your responsibility to report the amount received for income tax purposes.

Any information about you will be kept confidential. All of the data will be aggregated and no individual will be identifiable from the aggregated results. The data, with no personal identifiers, collected from this study will be maintained on a password-protected computer database in a restricted access area of the University. As well, the data will be electronically archived after completion of the study and maintained for two years and then erased.

Should you have any questions about the study, please contact either Edmund Wong (e32wong@uwaterloo.ca), Jinqiu Yang (j223yang@uwaterloo.ca) or Lin Tan (lintan@uwaterloo.ca). Further, if you would like to receive a copy of the results of this study, please contact either investigator.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin at 519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.

====

There will be a total of 15 questions. The steps for answering each question is of the following:

1. Read the marked source code that is highlighted and try to understand it
2. Write down a short description of what you think the code is about
   (without looking at the next question, it is okay if you don't know the answer)
3. Read the provided comment, this comment describes the marked source code that you just read
4. Rate the provided comment on its accuracy, adequacy, conciseness and usefulness.

I had read the above information and would like to participate in the user study. *

◉ I would like to participate into the user study.

- ○ I would not like to participate into the user study.

What is your education level? *

- ◉ Undergraduate Student
- ○ Graduate Student
- ○ Faculty Staff

Do you have experience in the software development industry? *

- ◉ Yes
- ○ No

Do you have experience in Android software development? *

- ◉ Yes
- ○ No

How many years of programming experience do you have? *

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5
- ○ 6
- ◉ 7
- ○ 8
- ○ 9
- ○

10+

Tutorial Question

Please read the marked code between "===="

```
private static InetAddress getFirstNonLoopbackAddress(boolean preferIpv4, boolean preferIPv6) throws SocketException {
  ####
  Enumeration en = NetworkInterface.getNetworkInterfaces();
  while (en.hasMoreElements()) {
    NetworkInterface i = (NetworkInterface) en.nextElement();
    for (Enumeration en2 = i.getInetAddresses(); en2.hasMoreElements();) {
      InetAddress addr = (InetAddress) en2.nextElement();
      if (!addr.isLoopbackAddress()) {
        if (addr instanceof Inet4Address) {
          if (preferIPv6) {
            continue;
          }
          return addr;
        }
        if (addr instanceof Inet6Address) {
          if (preferIpv4) {
            continue;
  ####
          }
          return addr;
        }
      }
    }
  }
  return null;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: return prefered vesrion of the ip address form of the local system

Please read this comment: "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q1

Please read the marked code between "===="

```
@SuppressWarnings({"unchecked"})
public static <T> T serializeDeserialize(T o) throws Exception {
  if ( o == null ) {
    return null;
  }

  ====
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  ObjectOutputStream oos = new ObjectOutputStream( baos );
  oos.writeObject( o );
  byte[] buffer = baos.toByteArray();
  ====
  baos.close();

  ByteArrayInputStream bais = new ByteArrayInputStream( buffer );
  ObjectInputStream ois = new ObjectInputStream( bais );
  return (T) ois.readObject();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- (●) I do not have an answer
- ( ) Other: _____

Please read this comment: "Serialize object with outputstream." *

- (●) I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	( )	(●)	( )	( )	( )	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
						Strongly

Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q2

Please read the marked code between "===="

```
final BufferedImage cached = scaledImages.get(d);
if (cached != null) return cached;

// Directly scaling to less than half size would ignore some pixels.
// Prevent that by halving the base image size as often as needed.
while(wNew*2 <= w && hNew*2 <= h) {
    w = (w+1)/2;
    h = (h+1)/2;
    ====
    BufferedImage halved = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = halved.createGraphics();
    // For halving bilinear should most correctly average 2x2 pixels.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    g.drawImage(im, 0, 0, w, h, null);
    g.dispose();
    ====
    im = halved;
}

if(wNew != w || hNew != h) {
    BufferedImage scaled = new BufferedImage(wNew, hNew,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = scaled.createGraphics();
    // Bicubic should give best quality for odd scaling factors.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    g.drawImage(im, 0, 0, wNew, hNew, null);
    g.dispose();
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: compress image by half

Please read this comment: "Creating a scaled version of an image." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q3

Please read the marked code between "===="

```
/**
 * @param node
 * @return
 */
public IDocumentElementNode clone(IDocumentElementNode node) {
  IDocumentElementNode clone = null;
  try {
    // Serialize
    ####
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bout);
    out.writeObject(node);
    out.flush();
    out.close();
    byte[] bytes = bout.toByteArray();
    // Deserialize
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream in = new ObjectInputStream(bin);
    ####
    clone = (IDocumentElementNode) in.readObject();
    in.close();
    // Reconnect
    clone.reconnect(this, fModel);
  } catch (IOException e) {
    clone = null;
  } catch (ClassNotFoundException e) {
    clone = null;
  }

  return clone;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ◉ I do not have an answer
- ○ Other: _____

Please read this comment: "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q4

Please read the marked code between "===="

```
/**
 * Deletes a directory or file
 * <p/>
 * Taken from
 * http://forum.java.sun.com/thread.jspa?threadID=470197&messageID=2169110
 * <p/>
 * Author: jfbriere
 *
 * @param file
 */
public static void deleteRecursive(File file) {
    if (file.isDirectory()) {
    ####
        File[] fileArray = file.listFiles();

        if (fileArray != null) {
            for (File aFileArray : fileArray) {
                deleteRecursive(aFileArray);
            }
        }
    }
    file.delete();
    ####
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: rm -r

Please read this comment: "Delete a folder with files using Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the

marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q5

Please read the marked code between "===="

```
Map result;
if ( scratch_file_is == null ){
  result = new LightHashMap();
}else{
    // System.out.println( "read cache file " + scratch_file_name + " for " + this );
  ####
  Properties p = new Properties();
  InputStream fis = scratch_file_is;
  try{
    p.load( fis );
    fis.close();
  ####
    scratch_file_is = new FileInputStream( scratch_file_name );
    messages = new LightHashMap();
    messages.putAll( p );
    result = messages;
  }catch( Throwable e ){
    if ( fis != null ){
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: read properity from a file

Please read this comment: "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q6

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  ####
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
      cal.setTime(dateValue);
    cal.set(Calendar.YEAR, year);
    cal.set(Calendar.MONTH, month);
    cal.set(Calendar.DAY_OF_MONTH, day);
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: get current time and date

Please read this comment: "If you want to extract more human-readable date/time information from that Calendar object, you can do something like." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q7

Please read the marked code between "===="

```
/**
 * Returns the contents of the given file as a char array.
 * When encoding is null, then the platform default one is used
 * @throws IOException if a problem occured reading the file.
 */
public static char[] getFileCharContent(File file, String encoding) throws IOException {
  ####
  InputStream stream = null;
  try {
    stream = new FileInputStream(file);
    return getInputStreamAsCharArray(stream, (int) file.length(), encoding);
  } finally {
    if (stream != null) {
      try {
        stream.close();
      } catch (IOException e) {
  ####
        // ignore
      }
    }
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- (◉) I do not have an answer
- ( ) Other: _____

Please read this comment: "Close an established network connection." *

- (◉) I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	( )	( )	( )	(◉)	( )	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q8

Please read the marked code between "===="

```
public static String getMetaModelSourceAsString(Class<?> clazz) {
  File sourceFile = getMetaModelSourceFileFor( clazz );
  StringBuilder contents = new StringBuilder();

  try {
    BufferedReader input = new BufferedReader( new FileReader( sourceFile ) );
    ####
    try {
      String line;
      /*
          * readLine is a bit quirky :
          * it returns the content of a line MINUS the newline.
          * it returns null only for the END of the stream.
          * it returns an empty String if two newlines appear in a row.
          */
      while ( ( line = input.readLine() ) != null ) {
        contents.append( line );
        contents.append( System.getProperty( "line.separator" ) );
      }
    }
    finally {
      input.close();
    ####
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
  }

  return contents.toString();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: read content from a file

Please read this comment: "Here is a function to read the file." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1	2	3	4	5
Strongly				

Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q9

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);
  ####

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
```

Please write your comment that describes the functionality of the above marked code segment *

- I do not have an answer

Other: get time and date up to millisec

Please read this comment: "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q10

Please read the marked code between "===="

```
if ( scratch_file_is == null ){

  result = new LightHashMap();

}else{

    // System.out.println( "read cache file " + scratch_file_name + " for " + this );

  ####
  Properties p = new Properties();

  InputStream fis = scratch_file_is;

  try{

    p.load( fis );

    fis.close();
    ####
    scratch_file_is = new FileInputStream( scratch_file_name );

    messages = new LightHashMap();

    messages.putAll( p );

    result = messages;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: try load the properity from a file

Please read this comment: "try to load a Properties object first." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q11

Please read the marked code between "===="

```
private int getBitmapSampleSize(BitmapFactory.Options options, int idealWidth, int idealHeight) {
    int width = options.outWidth;
    ####
    int height = options.outHeight;
    int inSampleSize = 1;
    if (height > idealHeight || width > idealWidth) {
        if (width > height) {
            inSampleSize = Math.round((float)height / (float)idealHeight);
        } else {
            inSampleSize = Math.round((float)width / (float)idealWidth);
    ###
        }
    }
    return inSampleSize;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: get ratio to ideal sample size

Please read this comment: "Calculate sample size." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q12

Please read the marked code between "===="

```
public void checkExternalStorage(){
  #####
  String state = Environment.getExternalStorageState();
  if(Environment.MEDIA_MOUNTED.equals(state)){
    // ok
  } else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    AccessibleToast.makeText(this, R.string.sd_mounted_ro, Toast.LENGTH_LONG).show();
  } else {
  ####
    AccessibleToast.makeText(this, R.string.sd_unmounted, Toast.LENGTH_LONG).show();
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- (●) I do not have an answer
- ( ) Other: .....

Please read this comment: "Get path to secondary external directory for Camera files." *

- (●) I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	( )	( )	(●)	( )	( )	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q13

Please read the marked code between "===="

```
/**
 * Returns true if the given Activity has hardware acceleration enabled
 * in its manifest, or in its foreground window.
 *
 * TODO(husky): Remove when initialize() is refactored (see TODO there)
 * TODO(dtrainor) This is still used by other classes.  Make sure to pull some version of this
 * out before removing it.
 */
public static boolean hasHardwareAcceleration(Activity activity) {
    // Has HW acceleration been enabled manually in the current window?
    #####
    Window window = activity.getWindow();
    if (window != null) {
        if ((window.getAttributes().flags
                & WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    }

    // Has HW acceleration been enabled in the manifest?
    try {
        ActivityInfo info = activity.getPackageManager().getActivityInfo(
                activity.getComponentName(), 0);
        if ((info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {
        Log.e("Chrome", "getActivityInfo(self) should not fail");
    ####
    }

    return false;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- (•) I do not have an answer
- ( ) Other: _____

Please read this comment: "Detect Hardware Acceleration at Runtime." *

- (•) I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	(•)	( )	( )	( )	( )	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q14

Please read the marked code between "===="

```
final void append(String itemID, String source, String[] newTexts, String linkURL) {

  StringBuilder newTextCombined = new StringBuilder();

  if (source != null) {
    newTextCombined.append(source).append(" : ");
  }

  int linkStart = newTextCombined.length();

  ####
  boolean first = true;
  for (String newText : newTexts) {
    if (first) {
      newTextCombined.append(newText);
      first = false;
    } else {
      newTextCombined.append(" [");
      newTextCombined.append(newText);
  ###
      newTextCombined.append(']');
    }
  }

  int linkEnd = newTextCombined.length();

  String newText = newTextCombined.toString();
  Spannable content = new SpannableString(newText + "\n\n");
```

Please write your comment that describes the functionality of the above marked code segment *

- (●) I do not have an answer
- ( ) Other: _____

Please read this comment: "Is often best to use StringBuilder to concatenate strings." *

- (●) I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	( )	( )	( )	(●)	( )	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q15

Please read the marked code between "===="

```
public View getView(int position, View convertView, ViewGroup parent)
{
  ####
  View currView;
  if(convertView == null)
  {
    LayoutInflater li = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    currView = li.inflate(R.layout.icon, null);
  }
  else
  {
    currView = convertView;
  }

  TextView tv = (TextView) currView.findViewById(R.id.icon_text);
  tv.setText("" + position);
  ImageView iv = (ImageView) currView.findViewById(R.id.icon_image);
  ####
  iv.setImageResource(Icons.iconToResId(position));

  return currView;
 }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- (●) I do not have an answer
- ( ) Other: _____

Please read this comment: "In the code example under the referred link the author sets values for the view only at creation time, so each time the framework is reusing the view, it has the same properties." *

- (●) I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	( )	(●)	( )	( )	( )	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

1	2	3	4	5

Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Post Study

All the comments that were shown were generated using an automated tool. Do you think it is helpful to have comments like that? *

- ◉ Yes
- ○ No
- ○ Maybe
- ○ Other: _____

Will you use the automated tool if it is made available? *

- ○ Yes
- ○ No
- ○

- Maybe
- Other: _____

AutoComment User Study

Information Letter for Quality of Source Code Comments

You are invited to participate in a research study conducted by Edmund Wong and Jinqiu Yang, under the supervision of Prof. Lin Tan of the Electrical and Computer Engineering Department at University of Waterloo, Canada. The objective of the research study is to assess the quality of source code comments.

If you decide to volunteer, you will be asked to complete a 1 hour session. 10 minutes will be spent on training, and the remainder on some tasks we ask you to perform. As a participant, you would be asked to write comments for source code segments. After that, we will provide you with comments that describe the given code segments, and you will rate the given comments' accuracy, adequacy, conciseness and usefulness based on a five-point scale. An example question sheet will be provided prior to the study.

Participants must have at least one year of programming experience in Java. Participation in this study is voluntary. You may decline to answer any questions that you do not wish to answer and you can withdraw your participation at any time by advising the researcher. There are no known or anticipated risk from participating in this study. No direct benefit is anticipated from this study.

This is an in-person study that will take approximately 1 hour at DC 3573. In appreciation, we will remunerate you $10. If you withdraw participation you will receive $5 per half hour. The amount received is taxable. It is your responsibility to report the amount received for income tax purposes.

Any information about you will be kept confidential. All of the data will be aggregated and no individual will be identifiable from the aggregated results. The data, with no personal identifiers, collected from this study will be maintained on a password-protected computer database in a restricted access area of the University. As well, the data will be electronically archived after completion of the study and maintained for two years and then erased.

Should you have any questions about the study, please contact either Edmund Wong (e32wong@uwaterloo.ca), Jinqiu Yang (j223yang@uwaterloo.ca) or Lin Tan (lintan@uwaterloo.ca). Further, if you would like to receive a copy of the results of this study, please contact either investigator.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin at 519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.

====

There will be a total of 15 questions. The steps for answering each question is of the following:

1. Read the marked source code that is highlighted and try to understand it
2. Write down a short description of what you think the code is about
   (without looking at the next question, it is okay if you don't know the answer)
3. Read the provided comment, this comment describes the marked source code that you just read
4. Rate the provided comment on its accuracy, adequacy, conciseness and usefulness.

I had read the above information and would like to participate in the user study. *

◉ I would like to participate into the user study.

- ○ I would not like to participate into the user study.

What is your education level? *

- ○ Undergraduate Student
- ◉ Graduate Student
- ○ Faculty Staff

Do you have experience in the software development industry? *

- ◉ Yes
- ○ No

Do you have experience in Android software development? *

- ◉ Yes
- ○ No

How many years of programming experience do you have? *

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ◉ 5
- ○ 6
- ○ 7
- ○ 8
- ○ 9
- ○

10+

Tutorial Question

Please read the marked code between "===="

```
private static InetAddress getFirstNonLoopbackAddress(boolean preferIpv4, boolean preferIPv6) throws SocketException {
  ####
  Enumeration en = NetworkInterface.getNetworkInterfaces();
  while (en.hasMoreElements()) {
    NetworkInterface i = (NetworkInterface) en.nextElement();
    for (Enumeration en2 = i.getInetAddresses(); en2.hasMoreElements();) {
      InetAddress addr = (InetAddress) en2.nextElement();
      if (!addr.isLoopbackAddress()) {
        if (addr instanceof Inet4Address) {
          if (preferIPv6) {
            continue;
          }
          return addr;
        }
        if (addr instanceof Inet6Address) {
          if (preferIpv4) {
            continue;
  ###
          }
          return addr;
        }
      }
    }
  }
  return null;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Return a non-loopback InetAddress given that the given preference of IP version matches

Please read this comment: "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q1

Please read the marked code between "===="

```
@SuppressWarnings({"unchecked"})
public static <T> T serializeDeserialize(T o) throws Exception {
  if ( o == null ) {
    return null;
  }

  ====
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  ObjectOutputStream oos = new ObjectOutputStream( baos );
  oos.writeObject( o );
  byte[] buffer = baos.toByteArray();
  ====
  baos.close();

  ByteArrayInputStream bais = new ByteArrayInputStream( buffer );
  ObjectInputStream ois = new ObjectInputStream( bais );
  return (T) ois.readObject();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Write the given object to output stream

Please read this comment: "Serialize object with outputstream." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
						Strongly

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q2

Please read the marked code between "===="

```
final BufferedImage cached = scaledImages.get(d);
if (cached != null) return cached;

// Directly scaling to less than half size would ignore some pixels.
// Prevent that by halving the base image size as often as needed.
while(wNew*2 <= w && hNew*2 <= h) {
    w = (w+1)/2;
    h = (h+1)/2;
    ====
    BufferedImage halved = new BufferedImage(w, h,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = halved.createGraphics();
    // For halving bilinear should most correctly average 2x2 pixels.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    g.drawImage(im, 0, 0, w, h, null);
    g.dispose();
    ====
    im = halved;
}

if(wNew != w || hNew != h) {
    BufferedImage scaled = new BufferedImage(wNew, hNew,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = scaled.createGraphics();
    // Bicubic should give best quality for odd scaling factors.
    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BICUBIC);
    g.drawImage(im, 0, 0, wNew, hNew, null);
    g.dispose();
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Create and render halved image

Please read this comment: "Creating a scaled version of an image." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q3

Please read the marked code between "===="

```
/**
 * @param node
 * @return
 */
public IDocumentElementNode clone(IDocumentElementNode node) {
  IDocumentElementNode clone = null;
  try {
    // Serialize
    ####
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bout);
    out.writeObject(node);
    out.flush();
    out.close();
    byte[] bytes = bout.toByteArray();
    // Deserialize
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream in = new ObjectInputStream(bin);
    ####
    clone = (IDocumentElementNode) in.readObject();
    in.close();
    // Reconnect
    clone.reconnect(this, fModel);
  } catch (IOException e) {
    clone = null;
  } catch (ClassNotFoundException e) {
    clone = null;
  }

  return clone;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Write and read the given object

Please read this comment: "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

Q4

Please read the marked code between "===="

```
/**
 * Deletes a directory or file
 * <p/>
 * Taken from
 * http://forum.java.sun.com/thread.jspa?threadID=470197&messageID=2169110
 * <p/>
 * Author: jfbriere
 *
 * @param file
 */
public static void deleteRecursive(File file) {
    if (file.isDirectory()) {
    ####
        File[] fileArray = file.listFiles();

        if (fileArray != null) {
            for (File aFileArray : fileArray) {
                deleteRecursive(aFileArray);
            }
        }
    }
    file.delete();
    ####
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: If the given file is a directory, recursively delete all files inside

Please read this comment: "Delete a folder with files using Java." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the

marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q5

Please read the marked code between "===="

```
Map result;
if ( scratch_file_is == null ){
  result = new LightHashMap();
}else{
    // System.out.println( "read cache file " + scratch_file_name + " for " + this );
  ####
  Properties p = new Properties();
  InputStream fis = scratch_file_is;
  try{
    p.load( fis );
    fis.close();
  ####
    scratch_file_is = new FileInputStream( scratch_file_name );
    messages = new LightHashMap();
    messages.putAll( p );
    result = messages;
  }catch( Throwable e ){
    if ( fis != null ){
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Load properties

Please read this comment: "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1 2 3 4 5

Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q6

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  ####
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
      cal.setTime(dateValue);
    cal.set(Calendar.YEAR, year);
    cal.set(Calendar.MONTH, month);
    cal.set(Calendar.DAY_OF_MONTH, day);
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Get the date information (year,month, etc) from value

Please read this comment: "If you want to extract more human-readable date/time information from that Calendar object, you can do something like." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q7

Please read the marked code between "===="

```
/**
 * Returns the contents of the given file as a char array.
 * When encoding is null, then the platform default one is used
 * @throws IOException if a problem occured reading the file.
 */
public static char[] getFileCharContent(File file, String encoding) throws IOException {
  ####
  InputStream stream = null;
  try {
    stream = new FileInputStream(file);
    return getInputStreamAsCharArray(stream, (int) file.length(), encoding);
  } finally {
    if (stream != null) {
      try {
        stream.close();
      } catch (IOException e) {
  ####
        // ignore
      }
    }
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Read file as char array

Please read this comment: "Close an established network connection." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Q8

Please read the marked code between "===="

```
public static String getMetaModelSourceAsString(Class<?> clazz) {
  File sourceFile = getMetaModelSourceFileFor( clazz );
  StringBuilder contents = new StringBuilder();

  try {
    BufferedReader input = new BufferedReader( new FileReader( sourceFile ) );
    ####
    try {
      String line;
      /*
          * readLine is a bit quirky :
          * it returns the content of a line MINUS the newline.
          * it returns null only for the END of the stream.
          * it returns an empty String if two newlines appear in a row.
          */
      while ( ( line = input.readLine() ) != null ) {
        contents.append( line );
        contents.append( System.getProperty( "line.separator" ) );
      }
    }
    finally {
      input.close();
    ####
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
  }

  return contents.toString();
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Create a string from reading the given file line by line

Please read this comment: "Here is a function to read the file." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

1	2	3	4	5
				Strongly

Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q9

Please read the marked code between "===="

```
protected void doSetValue(Object value) {
  Date date = (Date) value;

  Date dateValue;
  Date timeValue;

  Calendar cal = (Calendar) calendar.get();
  if (date == null)
    cal.clear();
  else
    cal.setTime(date);

  ####
  int year = cal.get(Calendar.YEAR);
  int month = cal.get(Calendar.MONTH);
  int day = cal.get(Calendar.DAY_OF_MONTH);
  int hour = cal.get(Calendar.HOUR_OF_DAY);
  int minute = cal.get(Calendar.MINUTE);
  int second = cal.get(Calendar.SECOND);
  int millis = cal.get(Calendar.MILLISECOND);
  ####

  if (date == null) {
    dateValue = null;
  } else {
    dateValue = (Date) dateObservable.getValue();
    if (dateValue == null)
      cal.clear();
    else
```

Please write your comment that describes the functionality of the above marked code segment *

- I do not have an answer

Other: Get the date information (year,month, etc) from value

Please read this comment: "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	◉	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q10

Please read the marked code between "===="

```
if ( scratch_file_is == null ){

  result = new LightHashMap();

}else{

    // System.out.println( "read cache file " + scratch_file_name + " for " + this );

  ####
  Properties p = new Properties();

  InputStream fis = scratch_file_is;

  try{

    p.load( fis );

    fis.close();
    ####
    scratch_file_is = new FileInputStream( scratch_file_name );

    messages = new LightHashMap();

    messages.putAll( p );

    result = messages;
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Load properties

Please read this comment: "try to load a Properties object first." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q11

Please read the marked code between "===="

```
private int getBitmapSampleSize(BitmapFactory.Options options, int idealWidth, int idealHeight) {
    int width = options.outWidth;
    ####
    int height = options.outHeight;
    int inSampleSize = 1;
    if (height > idealHeight || width > idealWidth) {
        if (width > height) {
            inSampleSize = Math.round((float)height / (float)idealHeight);
        } else {
            inSampleSize = Math.round((float)width / (float)idealWidth);
    ###
        }
    }
    return inSampleSize;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Get the ratio at which the image will be decreased towards ideal size

Please read this comment: "Calculate sample size." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q12

Please read the marked code between "===="

```
public void checkExternalStorage(){
  #####
  String state = Environment.getExternalStorageState();
  if(Environment.MEDIA_MOUNTED.equals(state)){
    // ok
  } else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    AccessibleToast.makeText(this, R.string.sd_mounted_ro, Toast.LENGTH_LONG).show();
  } else {
  ######
    AccessibleToast.makeText(this, R.string.sd_unmounted, Toast.LENGTH_LONG).show();
  }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Check if the external storage is in the right state

Please read this comment: "Get path to secondary external directory for Camera files." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q13

Please read the marked code between "===="

```
/**
 * Returns true if the given Activity has hardware acceleration enabled
 * in its manifest, or in its foreground window.
 *
 * TODO(husky): Remove when initialize() is refactored (see TODO there)
 * TODO(dtrainor) This is still used by other classes.  Make sure to pull some version of this
 * out before removing it.
 */
public static boolean hasHardwareAcceleration(Activity activity) {
    // Has HW acceleration been enabled manually in the current window?
    #####
    Window window = activity.getWindow();
    if (window != null) {
        if ((window.getAttributes().flags
                & WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    }

    // Has HW acceleration been enabled in the manifest?
    try {
        ActivityInfo info = activity.getPackageManager().getActivityInfo(
                activity.getComponentName(), 0);
        if ((info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {
        Log.e("Chrome", "getActivityInfo(self) should not fail");
    ####
    }

    return false;
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Check if hardware acceleration is enabled

Please read this comment: "Detect Hardware Acceleration at Runtime." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

Q14

Please read the marked code between "===="

```
final void append(String itemID, String source, String[] newTexts, String linkURL) {

  StringBuilder newTextCombined = new StringBuilder();

  if (source != null) {
    newTextCombined.append(source).append(" : ");
  }

  int linkStart = newTextCombined.length();

  ####
  boolean first = true;
  for (String newText : newTexts) {
    if (first) {
      newTextCombined.append(newText);
      first = false;
    } else {
      newTextCombined.append(" [");
      newTextCombined.append(newText);
  ###
      newTextCombined.append(']');
    }
  }

  int linkEnd = newTextCombined.length();

  String newText = newTextCombined.toString();
  Spannable content = new SpannableString(newText + "\n\n");
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Concatenate newTexts

Please read this comment: "Is often best to use StringBuilder to concatenate strings." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	◉	○	○	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	◉	○	○	○	Strongly Disagree

Q15

Please read the marked code between "===="

```
public View getView(int position, View convertView, ViewGroup parent)
{
  ####
  View currView;
  if(convertView == null)
  {
    LayoutInflater li = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    currView = li.inflate(R.layout.icon, null);
  }
  else
  {
    currView = convertView;
  }

  TextView tv = (TextView) currView.findViewById(R.id.icon_text);
  tv.setText("" + position);
  ImageView iv = (ImageView) currView.findViewById(R.id.icon_image);
  ####
  iv.setImageResource(Icons.iconToResId(position));

  return currView;
 }
}
```

Please write your comment that describes the functionality of the above marked code segment *

- ○ I do not have an answer
- ◉ Other: Update the text of icon with the given position

Please read this comment: "In the code example under the referred link the author sets values for the view only at creation time, so each time the framework is reusing the view, it has the same properties." *

- ◉ I had read this comment.

This comment is accurate in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is adequate (i.e., not missing information) in describing the marked code: *

1	2	3	4	5

Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment is concise in describing the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	◉	○	○	Strongly Disagree

This comment helps me understand the marked code: *

	1	2	3	4	5	
Strongly Agree	○	○	○	◉	○	Strongly Disagree

Post Study

All the comments that were shown were generated using an automated tool. Do you think it is helpful to have comments like that? *

- ○ Yes
- ○ No
- ◉ Maybe
- ○ Other: _____

Will you use the automated tool if it is made available? *

- ◉ Yes
- ○ No
- ○

Maybe

Other: _____
