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 non-loopback address of the preferred type (v4 or v6).

Please compare the two comments: "Returns the first IP address of the preferred type in the network interface" and "Get the ip of the computer on linux through Java." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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: Converts the passed object to a byte array.

Please compare the two comments: "Writes the value of o into a new object of output stream" and "Serialize object with outputstream." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

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

1	2	3	4	5

Comment #1	○	○	◉	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

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: Halves the dimensions of im

Please compare the two comments: "construct the image of halved dimensions, render it through bilinear interpolation, and display the result." and "Creating a scaled version of an image." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	○	◉	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

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: Clones the passed node by serializing then deserializing it.

Please compare the two comments: "Serialize the output object stream and deserialize the input object stream" and "Is more reliable to read and write (String) objects, which bypasses the encoding/decoding gamble." *

I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

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 deletes a directory tree in the file system.

Please compare the two comments: "Delete all files within the directory through recursion" and "Delete a folder with files using Java." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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: Loads properties through the input stream.

Please compare the two comments: "Load scratch_file_is into a new Properties object" and "So the first thing you have to is read the properties file in." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

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 individual time components from cal

Please compare the two comments: "Retrieve the date and time" and "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.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

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 get file contents as a char array. Handles errors by ignoring them.

Please compare the two comments: "Return the file input stream as a character array" and "Close an established network connection." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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 file into a StringBuilder, fixing newlines removed by readLine.

Please compare the two comments: "Reads the input line-by-line and appends the separator between them" and "Here is a function to read the file." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

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 time components stored in cal.

Please compare the two comments: "Returns the date and time up to millisecond precision" and "Get year, month, day, hours, minutes, seconds and milliseconds of the current moment in Java." *

I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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 properties from the file.

Please compare the two comments: "Load scratch_file_is into a new Properties object" and "try to load a Properties object first." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

1 2 3 4 5

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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 size provided in options, possibly scaled down to fit the provided ideal dimensions.

Please compare the two comments: "Returns the scaling factor of the sample compared to the ideal dimensions" and "Calculate sample size." *

- I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the

marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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: Prints a message if the external storage is mounted read only.

Please compare the two comments: "If the external storage is read-only,

then report it" and "Get path to secondary external directory for Camera files." *

- (◉) I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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 the given activity has hardware acceleration enabled in the window or in the manifest.

Please compare the two comments: "Checks whether hardware acceleration is activated on the current window, or in the manifest", "Detect Hardware Acceleration at Runtime." *

- I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	○	◉	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

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: Concatenates newTexts with spaces, with all but the first string surrounded by square brackets

Please compare the two comments: "Combine all the texts using [ ] for the second text onwards" and "Is often best to use StringBuilder to concatenate strings." *

- ◉ I had read this comment.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	◉	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	◉	○	○	○	○	Comment #2

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: Initializes convertView, or a newly created view if convertView is null.

Please compare the two comments: "Construct the currView object from convertView and set its text view and image view" and "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.

Which comment is more accurate in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment is more adequate (i.e., not missing information) in

describing the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Which comment is more concise (i.e., straight to the point) in describing the marked code: *

	1	2	3	4	5	
Comment #1	○	◉	○	○	○	Comment #2

Which comment helps better understand the marked code: *

	1	2	3	4	5	
Comment #1	○	○	○	◉	○	Comment #2

Post Study

Do you think it is helpful to have a tool that generates comments automatically for the source code? *

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

The first comment is written by a human and the second comment is

generated automatically. Will you use a tool that can generate comments automatically? *

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