Circle the bullet in front of each statement that accurately describes your instructor's course policies:

- You are expected to read the lab assignment before lab begins.
- If you miss more than three lectures, your grade will be dropped by half a letter grade.
- Lab attendance is required.
- You must submit every lab assignment in order to pass this course.
- Homework is typically due before taking a weekly quiz.
- Having a copy of all or part of another student's source code, just to look at for a little help, is considered cheating.
- All students must submit a copy of their lecture notes before the end of the day.
- Your instructor is such a nice guy that if you turn in an assignment 45 minutes late, he wouldn't consider it late.
- Your instructor will give you a free lunch if you invite him to lunch.
- The final exam for this course will be comprehensive.

Do not circle the bullet in front of any statement that incorrectly describes your instructor's course policies.

Complete the method below that accepts three numbers and returns the smallest value of the three.

```
public static double minimum(double val1, double val2, double val3) {



}
```

What will be displayed when the following code is executed?

```
try {
  String word;
  int size = word.length();
  Double.parseDouble("quiz 4");
  System.out.println("1");
} catch (NullPointerException e) {
  System.out.println("2");
} catch (NumberFormatException e) {
  System.out.println("3");
} catch (RuntimeException e) {
  System.out.println("4");
} finally {
  System.out.println("5");
}
System.out.println("6");
```

Write a class method, called splitString, that accepts a String containing a number of words and return an ArrayList of Strings with each word stored as a separate element in the ArrayList.

Event driven programming is implemented with sources and listeners. Give one example of an event source (name of a class). What must be done in order for a class to listen to that particular event source? E.g., what must be done to the class to make it able to listen and what must be done to the event source to tell it that it has a listener?

Suppose that IntA and IntB are interfaces. Suppose further that ClassA implements both interfaces, ClassB implements IntB but **not** IntA, and ClassC does not implement any interfaces. Identify which of the following lines of code are illegal.

```
public static void main(String[] args) {
  IntA intAref = null;
  IntB intBref = null;
  ClassA clsAref = null;
  ClassB clsBref = null;
  ClassC clsCref = null;
  intAref = new IntA();
  intBref = new IntB();
  clsAref = new ClassA();
  clsBref = new ClassB();
  clsCref = new ClassC();
  clsAref.toString();
  intAref = clsAref;
  intAref.toString();
  intAref = clsBref;
  intAref = clsCref;
  intBref = clsAref;
  intBref = clsBref;
  intBref = clsCref;
  clsAref = clsBref;
  clsBref = clsCref;
}
```

Consider the code provided on the additional page. Indicate what will be displayed as a result of the following statements:

```
PngImage img = new PngImage();
img.load("quiz.png");
System.out.println(img.getTransparency());
Image img2 = new JpgImage();
img2.getHeight();
img2.getScaledImage(20, 20);
```

```
public interface Transparency {
  public int getTransparency();
} // end of Transparency

public abstract class Image {

  protected int height;
  protected int width;

  public Image() {
    System.out.println("Image: Constructor");
  }

  public abstract void save(String filename);

  public abstract void load(String filename);

  public int getHeight() {
    System.out.println("Image: getHeight");
    return height;
  }

  public int getWidth() {
    System.out.println("Image: getWidth");
    return width;
  }

  public abstract Image getScaledImage(int w, int h);
} // end of Image

public class JpgImage extends Image {

  private int transparency;

  public JpgImage() {
    super();
    System.out.println("JpgImage: Constructor");
    transparency = 100;
  }

  public Image getScaledImage(int w, int h) {
    System.out.println("JpgImage: getScaledImage");
    return new JpgImage();
  }

  public void load(String filename) {
    System.out.println("JpgImage: load");
  }

  public void save(String filename) {
    System.out.println("JpgImage: save");
  }
} // end of JpgImage

public class PngImage extends Image implements Transparency {

  private int transparency;
```

```
  public PngImage() {
    System.out.println("PngImage: Constructor");
    transparency = 100;
  }

  public Image resize(int width, int height) {
    System.out.println("PngImage: resized");
    return new PngImage();
  }

  public Image getScaledImage(int width, int height) {
    System.out.println("PngImage: getScaledImage");
    return resize(width, height);
  }

  public void load(String filename) {
    System.out.println("PngImage: load");
  }

  public void save(String filename) {
    System.out.println("PngImage: save");
  }

  public int getTransparency() {
    System.out.println("PngImage: getTransparency");
    return transparency;
  }
} // end of PngImage
```

Implement the addImage() method below.

```
public class PhotoAlbum {

  protected ArrayList<Image> photos;

  public PhotoAlbum() {
    photos = new ArrayList<Image>();
  }

  // ...

  /**
    * Adds a photo to the photo album.  Accepts a string containing
    * the filename for the image.  The method checks the extension of
    * the filename.  The extension should be one of the following:
    *  .jpg — in this case a JpgImage object is added to the list of photos
    *  .png — in this case a PngImage object is added to the list of photos
    *
    * Any other extension (or no extension) should result in no image being
    * added to the list of photos.
    * @param imageFile The filename of the image to be added
    * @return True if an image was successfully added; False if no image
    *         was added.
    */
  public boolean addPhoto(String imageFile) {
```