UNIVERSITY OF WISCONSIN–LA CROSSE
Department of Computer Science
CS 120: Test Out Practice Exam

Name: ____

- Do not turn the page until instructed to do so. This exam contains 13 pages including the cover page and separate worksheets.
- You cannot use any notes, peers, or electronic devices. If you do so you will immediately receive 0 points on this exam.
- You have exactly **120 minutes** to complete the exam.
- Be sure to write your responses **clearly and legibly**. Points will not be awarded for responses that are unreadable by the professor.
- Show all of your work where appropriate.
- Pay attention to **spacing**, **capitalization**, and **formatting**, particularly when writing code.
- Additional sheets of paper may be requested if you need more room than is provided. Any additional sheets of paper used must be turned in with the exam and clearly labeled with your name and date. If answers are found on your additional paper, please be sure to clearly note that in the problem itself in this booklet
- This exam makes heavy use of the provided classes at the end of the exam. You will refer to them for many questions.

Problem	**Points**
1	______ / 10 pts.
2	______ / 10 pts.
3	______ / 10 pts.
4	______ / 3 pts.
5	______ / 5 pts.
6	______ / 8 pts.
7	______ / 13 pts.
8	______ / 8 pts.
Total	______ / 67 pts.

1. (*10 points*) Match the vocabulary/values at the bottom of the page with the statements below. There should be only one match for each statement; as a result, you will have vocabulary left over.

(n)	The technique that allows multiple Java classes to share common pieces of code.
(e)	A sequence of characters that should be interpreted as textual data, not programming instructions.
(q)	Checks the contents of the objects for equality.
(d)	The physical components of a computer system.
(p)	When a data type is automatically converted to a wider type during evaluation.
(o)	The mechanism by which a program waits for user input (e.g., mouse click, button press).
(u)	Region of code where a variable can be used.
(m)	The ability to change the way a child class implements a method from the parent class.
(r)	Information or options passed to a method.
??	Generally describes the ability of methods to perform in different ways depending on the context.

(a) overloading
(b) parameters
(c) class
(d) hardware
(e) string literal
(f) selection sort
(g) casting
(h) main method
(i) `==`
(j) object
(k) control flow
(l) condition
(m) overriding
(n) inheritance
(o) listeners
(p) coercion
(q) `.equals()`
(r) arguments
(s) data structure
(t) insertion sort
(u) scope
(v) high-level programming language
(w) CPU
(x) `static`
(y) `super`

2. (*10 points*) Below are several Java variable declarations with the datatype of the declaration omitted. For each line, write the datatype that is necessary to make the declaration and assignment valid on the left, and the value that will be stored in the variable on the right; the first has been filled in for you as an example. Note that not all lines have a spot for filling in what the variable will contain. You can assume that all variables have not been previously declared, but that all the below lines are part of the same program (i.e., previously declared variables, like the example provided, can be referenced on subsequent lines). You can show work on scrap paper if desired (be sure to let me know where to find that work).

int	`value = 42; //`	42
String	`var = "value"; //`	"value" (1 point)
boolean	`x = 18 > 12 && 16 / 4 != 4.0; //`	false (1.5 points)
int	`y = 12/1/12; //`	1 (1 point)
Color	`c = Color.RED;`	
String	`q = "Magic number:  " + value; //`	"Magic number: 42" (1 point)
int	`k = "some text".substring(1, 5).indexOf('e'); //`	2 (1.5 points)
Scanner	`s = new Scanner(System.in);`	
char	`f = "more text".charAt(3); //`	'e' (1 point)
double	`w = (int) Math.pow(2, 1) * 0.25; //`	0.5 (1.5 points)
Drone or Robot	`d = new Drone("Quad", 0.5, 4);`	

3. (*10 points*) Below are several lines of Java code, some of which have incorrect syntax (i.e., they would be underlined in red in Eclipse). **Cross out** the statements that will fail to compile because of incorrect syntax.

```java
public class CompileCrossOut {
    public static void main(String[] args) {

        Drone d = new Drone("Krazy", 0.25, 3);

        Humanlike h = new Humanlike("Nao", 1.75, true);

        Robot r = new Robot("Generic");  invalid

        Robot temp;

        String batt = h.getBatteryLifeInHoursAndMinutes(5);  invalid

        temp = h;

        h = d;   invalid

        int mod = d.getModel();  invalid

        System.out.println(d);

        System.out.println(d.toString);  invalid

    }
}
```

Consider the following two pieces of code. Declare whether or not they will successfully execute (i.e., with no exceptions thrown). If no exception will be thrown, then write the output of the code. Otherwise, if an exception will be thrown, identify what exception it is and modify the code with the smallest fix necessary to remove the exception.

4. (*3 points*)

```
public class Arrays {
    public static void main(String[] args) {

        char[] arr = {'a', 'b', 'c', '!', '?'};

        for(int i = 0; i < arr.length; i--) {
            System.out.println(arr[i]);
        }

    }
}
```

Solution:

`ArrayIndexOutOfBoundsException` thrown when `i` decrements; fix is to have `i` increment

5. (*5 points*)

```
public class ObjectTrace {
    public static void main(String[] args) {

        Drone d1 = new Drone("Flyer", 1.5, 4);
        Drone d2 = new Drone("Aero", 0.75, 2);
        Robot temp, r1 = new Robot("Robo", 1.0);

        temp = r1;
        r1 = d1;
        d1 = d2;

        System.out.println(temp);
        System.out.println(r1);
        System.out.println(d1);
        System.out.println(d2);

    }
}
```

Solution:

```
Hi, my name is Robo.  I have 1 hour of battery life.
Hi, my name is Flyer.  I have 1 hour 30.0 minutes of battery life.  I am equipped
with 4 propellers.
Hi, my name is Aero.  I have 0 hour 45.0 minutes of battery life.  I am equipped
with 2 propellers.
Hi, my name is Aero.  I have 0 hour 45.0 minutes of battery life.  I am equipped
with 2 propellers.
```

6. (*8 points*) Write a `public`, `static`, non-void method called `split` that takes a `String` and `char` as input, and returns a 1D `String` array of the `String` argument split up according to the `char` argument. Note that the `char` itself should **not** appear in the array of returned values. **No credit will be given to solutions that use the `split` method provided by Java.** Below is a Javadoc comment with examples.

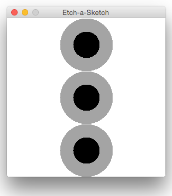
```
/**
 * Splits the provided string into an array, separating each piece of the
 *   string according to the char value. Two examples provided below:
 *
 * Input: "A very short sentence", ' '
 * Output: {"A", "very", "short", "sentence"}
 *
 * Input: "One sentence! Two sentences! Three", '!'
 * Output: {"One sentence", " Two sentences", " Three"}
 *
 * @param String str: the String to split
 * @param char ch: the char to use to split the String
 * @return An array of String values
 *
 */

 public static String[] split(String str, char ch) {
      String temp = str;
      int count = 0;
      while (temp.indexOf(ch) >= 0) {
           count++;
           temp = temp.substring(temp.indexOf(ch)+1);
      }

      String[] toReturn = new String[count];
      for (int i = 0; i < toReturn.length; ++i) {
           toReturn[i] = str.substring(0, str.indexOf(ch));
           str = str.substring(str.indexOf(ch)+1);
      }

      return toReturn;
 }
```

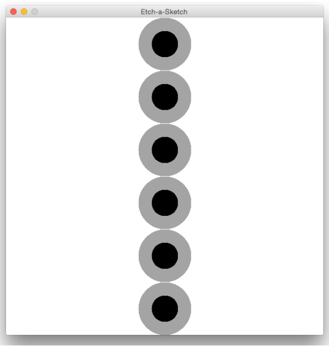
7. (*13 points*) Write a class called `Rescue` that is a subclass of `Robot` and describes a rescue robot used for emergencies. The `Rescue` class will have an attribute for emergency type, which is a `String` value (e.g., `"nuclear disaster", "first response"`). In addition to setting up the additional attribute, fill in the methods below.

```
public class Rescue extends Robot {

    private String type;

   /**
    * The constructor for Recuse should take a model, battery life,
    *   and rescue type as described above. (3 points)
    *
    * @param String model: the model of the robot
    * @param double batteryLife: the battery life of the robot
    * @param String type: the rescue type of the robot
    */

     public Rescue(String model, double batteryLife, String type) {
         super(model, batteryLife);
         this.type = type;
     }

   /**
    * The toString method for Rescue should override the parent's
    *   toString method. This toString should print everything the parent's
    *   toString method prints. It also prints an additional sentence
    *   afterwards noting the type. Example below:
    *
    * "Hi, my name is <model>. I have <time> hours of battery life. I am a
    *    robot intended to help with <type>."
    *
    * Full points will only be awarded for implementations that use the
    *   parent's toString method. (3 points)
    *
    * @return The String representation of this object
    */

    public String toString() {
        return super.toString() + " I am a robot intended to help with " + this.type + ".";
    }
```

```
/**
 * The method absorbBatteryLife takes in a 1D array of Robot objects and
 *   adds the battery life of each robot in the array to this robot.
 *   (no need to modify the robots in the array) The method should return
 *   a double value indicating how much battery life was added. (7 points)
 *
 * @param Robot[]: an array of Robot/Drone/Humanlike objects
 * @return The amount of battery life added
 */

 public double absorbBatteryLife(Robot[] arr) {
     double toAdd = 0;

     for (int i = 0; i < arr.length; ++i) {
         toAdd += arr[i].batteryLife();
     }

     this.batteryLife += toAdd;

     return toAdd;
  }
```

8. (*8 points*) Write a method `displayCircles` that produces some number of concentric circles, according to the height of the `Window` parameter `win`. Details are given in the method comment. Supporting GUI classes can be found on the attached **Class Worksheet**. The examples below show images for a window height of 300 and 600, respectively:





```
/**
 * This method draws a column of concentric circles with the following
 * properties:
 * - Each concentric circle is comprised of an outer circle with a diameter of
 *   100 pixels and an inner circle with a diameter of 50 pixels.
 * - The color of the outer circle is always a blend of 20 red, 191 green,
 *   and 68 blue. The color of the inner circle is always black.
 * - The x location of both the outer and the inner circle should be such that
 *   the center of the circle is at half the width of the window.
 * - The y location of the first outer circle will be at 0 pixels. Subsequent
 *   outer circles will be 100 pixels below the previous outer circle.
 * - All inner circles will be centered on top of an outer circle.
 *
 * @param win The window in which to draw the circles. It is guaranteed to
 *            have a white background, a width >= 100, and a height such that
 *            height % 100 == 0.
 */
private static void displayCircles(Window win) {
```

Solution:

```
    int middle = window.getWidth()/2;

    for(int i = 0; i < window.getHeight(); i+=100) {
        Oval oMagenta = new Oval(middle - 50, i, 100, 100);
        oMagenta.setBackground(Color.MAGENTA);
        window.add(oMagenta);
        Oval oBlack = new Oval(middle-25, i+25, 50, 50);
        oBlack.setBackground(Color.BLACK);
        window.add(oBlack);
    }
```

```
}
```

Class Worksheet – GUIs

`Window` Class

```
Window()
void setTitle( String s )
void setLocation( int x, int y )
void setSize( int w, int h )
void setBackground( Color color )
int getX()
int getY()
int getWidth()
int getHeight()
Color getBackground()
```

`Rectangle` Class

```
Rectangle( int x, int y, int w, int h )
void setLocation( int x, int y )
void setSize( int w, int h )
void setBackground( Color color )
int getX()
int getY()
int getWidth()
int getHeight()
Color getBackground()
```

`Oval` Class

```
Oval( int x, int y, int w, int h )
void setLocation( int x, int y )
void setSize( int w, int h )
void setBackground( Color color )
int getX()
int getY()
int getWidth()
int getHeight()
Color getBackground()
```

`Triangle` Class

```
// dir = 1 is up, dir = 0 is down
Triangle( int x, int y, int w, int h, int dir)
void setLocation( int x, int y )
void setSize( int w, int h )
void setBackground( Color color )
int getX()
int getY()
int getWidth()
int getHeight()
Color getBackground()
```

`Line` Class

```
Line( int x1, int y1, int x2, int y2 )
Line( int x1, int y1, int x2, int y2, int t )
void setLocation( int x, int y )
void setSize( int w, int h )
void setBackground( Color color )
int getX()
int getY()
int getWidth()
int getHeight()
Color getBackground()
```

`Color` Class

```
Color( int r, int g, int b )
int getRed()
int getBlue()
int getGreen()
```

`Robot` Class

```java
public class Robot {

    protected String model;
    protected double batteryLife;

    public Robot(String m, double bl) {
        model = m;
        batteryLife = bl;
    }

    public String getModel() {
        return model;
    }

    public double batteryLife() {
        return batteryLife;
    }

    public String toString() {
        String toReturn = "Hi, my name is " + model + ". I have "
                          + getBatteryLifeInHoursAndMinutes(batteryLife)
                          + " hours of battery life.";

        return toReturn;
    }

    private String getBatteryLifeInHoursAndMinutes(double value) {
        String toReturn = "";

        if(value > 0) {
            toReturn += (int) value + " hours ";
            value -= (int) value;
        }

        if(value > 0) {
            toReturn += (60 * value) + " minutes";
        }

        return toReturn;
    }

}
```

`Drone` Class

```java
public class Drone extends Robot {

    protected int numPropellers;

    public Drone(String m, double bl, int pr) {
        super(m, bl);
        numPropellers = pr;
    }

    public int getPropellers() {
        return numPropellers;
    }

    public String toString() {
        return super.toString() + " I am equipped with " + numPropellers +
    " propellers.";
    }

}
```

`Humanlike` Class

```java
public class Humanlike extends Robot {

    protected boolean hasLegs;

    public Humanlike(String m, double bl, boolean hl) {
        super(m, bl);
        hasLegs = hl;
    }

    public boolean hasLegs() {
        return hasLegs;
    }

}
```