1. In the Light Navigation lab, you used photo-resistor and capacitor to obtain information about light level of the room. Explain in your own words how the combination these components to accomplish this task.

   Using the built-in RCTIME function, the JStamp charges up the capacitor. It then measures the time for the capacitor to discharge in the RC circuit (where R = the photoresistor). When the voltage across the capacitor drops below the threshold and the pin registers false, the time measurement stops. Depending on the relative length, the relative brightness can be ascertained.

2. Briefly describe what kind of code (or program) generates an ArrayOutOfBoundsException?
   
   Trying to access data from an index (location) that is more than (array length -1)

3. If A and B are classes and A “has a” B, what does that mean?

   Class A has an instance variable of type is B. E.g. Freshman-DormRoom:
   Freshman class had instance variable of type DormRoom.
Recall the Maze Navigation lab. Assume that there can be a scenario where the MazeBot can move, but cannot reach the end of the maze. How would you modify your Maze Navigation code? Provide your answer in terms of pseudo code.

In order to figure out that mazebot has come back to start we need to mark the starting position (which assume 0,0). Then its position needs update as it is moving the maze. We also need to keep track of it direction.

Note that your solution does not be as exact from the lab, for example not necessary for m.getController().moveForward(). Just an outline or psuedocode is required.

Assumptions:
1. The maze is two dimensional space
2. The mazebot can determine its own NEWS and initially the default position chosen is North.
3. moveForward() is called, the move will cause position to change depending on the direction.
4. Turns are in place.
   a. For left turns it ok to just change the direction. Because once it turns left, it should still sense a wall on the right but nothing ahead and continue forward.
   b. For right turns, it important to move forward (otherwise it won't sense a wall on the right and it'll keep turning). Since we turned right and moved we must update directions and co-ordinates.

```java
done = true;
x = 0; y = 0;
d = ' '; //direction is initially blank
while(done) {
    if(checkRightWall && checkWallAhead()){
        moveForward();
        if(d == 'e'){
            x = x + 1;
        }
        else if(d == 'w'){
            x = x - 1;
        }
        else if(d == 'n'){
            y = y + 1;
        }
        else if(d == 's'){
            y = y - 1;
        }
        else
            d = 'n';
        y = y+1;
    }
    else if(rightWall() && checkWallAhead())){
        turnLeft();
        if(d == 'e')
            d = 'n';
        else if(d == 'w')
            d = 's';
        else if(d == 'n')
            d = 'w';
        else if(d == 's')
            d = 'e';
        else
            d = 'n';
    }
}
else{
    turnRight();
    moveForward();
    if(d == 'n'){
        d = 'e';
        x = x + 1;
    }
    else if(d == 's'){
        d = 'w';
        x = x - 1;
    }
    else if(d == 'e'){
        d = 's';
        y = y - 1;
    }
    else if(d == 'w'){
        d = 'n';
        y = y + 1;
    }
    else{
        d = 'n';
        y = y + 1;
    }
    //end of if-else-if-else
    if(x == 0 && y == 0){
        done = false;
    }
}
//end while
stop(); //stop moving
```
5. An object is an instance of a … **class**

6. Explain each of the following keyword:
   a. **static** - variable/ method declared static belongs to the class, not to individual objects
   b. **this** - the object that is executing the current code (value of this is the object’s heap address)

7. Given the following Foo class:

   ```java
   public class Foo{
       private int x;

       public Foo(int x){
           this.x = x;
       }

       public int getX() { return x;}
   }
   ```

   Indicate the value of the variable `result` below:
   ```java
   Foo a = new Foo(5);
   Foo b = new Foo(5);
   boolean result = a == b;
   ```
   Answer: **false**

   ```java
   Foo c;
   a = c;
   b = c;
   boolean result = a == b;
   ```
   Answer: **true**

8. What is difference between return, break and continue statement?
   The “return” statement is used to select the value that the method gives back to whatever calls it. In addition, “return” exits the entire method. In a loop, just a return statement will cause a compiler error unless there is return statement also at end of the loop.
   In a loop, the “continue” keyword finishes the current iteration of the loop and goes back to the top to re-evaluate the conditional and (if warranted) move on to the next iteration. The “break” keyword exits out the inner loop entirely.
9. Given the following array declaration:
   ```java
   int [] a = new int [5000];
   ```

   What is the type of each of the following expressions?
   a. a.length - int
   b. a[4999] - int
   c. a - int []

10. What does mysteryMethod do? What does the "temp" array store? In what cases might it fail, and how could we fix this?

    ```java
    public int mysteryMethod(int[] numbers) {
        int[] temp = new int[numbers.length];
        for (int i = 0; i < numbers.length; i++) {
            temp[numbers[i]]++;
        }
        int n = 0;
        int c = temp[0];
        for (int i = 0; i < temp.length; i++) {
            if (temp[i] > c) {
                n = i;
                c = temp[i];
            }
        }
        return n;
    }
    ```

    temp stores a count of each item in the array. If 2 shows up 3 times, then temp[2] = 3. The second loop finds the maximum frequency and returns the value. So if 7 appeared most frequently in the input array, it returns 7. This method can fail due to indexOutOfBounds for line that does the statement temp[numbers[i]]++. This can be fixed with an if-statement check.
11. Consider this Toolkit class:

```java
public class Toolkit{

    public void doit1(int [] data){
        int i = 0;
        while(i < data.length){
            data[i] = data[i] * 2;
            i = i + 2;
        }
    }

    public void doit2(int [] data, int m){
        for(int i = 0; i < data.length; i++){
            if(data[i] > m){
                data[i] = m;
            }
        }
    }

}//end of class Toolkit
```

(a) What does the doit1 method do? Explain.

**Doubles every item in the array located at an even index (0, 2, 4..)**

(b) What does the doit2 method do? Explain.

**Replaces every item in the array greater than m with m.**
12. Consider the following code:

```java
public class WeatherRecord {
    private double inchesOfRain;
    private double hiTemp;

    public WeatherRecord(double inchesOfRain, double hiTemp) {
        this.inchesOfRain = inchesOfRain;
        this.hiTemp = hiTemp;
    }

    public double getRainfall() { return inchesOfRain; }
    public double getHiTemp() { return hiTemp; }
}
```

Sample Dr Java Interactions:

```java
// Analyze the weather for a hypothetical 3-day month
WeatherRecord day1 = new WeatherRecord(0, 60); // 0 inches of rain, hi temp = 60
WeatherRecord day2 = new WeatherRecord(3, 65);
WeatherRecord day3 = new WeatherRecord(1, 68); // for simplicity, this month has 3 days
WeatherRecord[] record = new WeatherRecord[]{day1, day2, day3};
WeatherReporter reporter = new WeatherReporter();
WeatherRecord monthRecord = reporter.computeStats(record);
System.out.println(monthRecord.getRainfall()); // average rainfall for the month
Output: 1.3333333333333333
System.out.println(monthRecord.getHiTemp()); // highest temperature of the month
Output: 68.0
```

A WeatherRecord (code supplied above) is flexible in that it can hold weather information for a day, a month, even a year or a millennium.

A WeatherReporter has a single method called computeStats which takes an array of WeatherRecords as input, analyzes it, and returns a WeatherRecord which contains the results of the analysis.

Complete the computeStats method. You can assume that input array is valid (i.e. not null or length is greater than 0) and that it is full (contains no nulls).
public class WeatherReporter{
    //Method computeStats
    public WeatherRecord computeStats(WeatherRecord[] record){
        double monthRain = 0;
        for (int i = 0; i < records.length; i++){
            monthRain += records[i].getRainfall();
        }
        double avgMonthRain = monthRain/records.length;
        double monthHiTemp = records[0].getHiTemp();
        for (int i = 1; i < records.length; i++){
            if (records[i].getHiTemp() > monthHiTemp){
                monthHiTemp = records[i].getHiTemp();
            }
        }
        return new WeatherRecord(avgMonthRain, monthHiTemp);
    }
}

13. Consider the following code:

public class Super{
    public Super(){
        System.out.println("Super");
    }
}

public class Sub extends Super{
    public Sub(){
        System.out.println("Sub");
    }
}

a. What is the output when the statement below is executed?

Sub s = new Sub();
    Super
        Sub

b. True or False: The statement below is a valid statement.
Object o = new Sub();

    True. The super class type variable can hold a subtype variable.
14. Consider BetterBot exercise from Lab 10. Assume that you just completed writing `turnRight()` and `turnAround()` methods in the BetterBot class. Now write method called `returnToStart` which makes the BetterBot come to starting position i.e. (0, 0) from any position (x, y). As usual BetterBot inherit Bot behaviors such as `move()`, `turnLeft()`, `getDirection()`

```java
public void returnToStart()
{
    if(getDirection() == 's')
        turnAround();
    else if(getDirection() == 'w')
        turnRight();
    else if(getDirection() == 'e')
        turnLeft();

    while(move());
    turnLeft();
    while(move());
    turnAround();
}
```