1. Define the following terms using YOUR own words/diagrams. I’m not looking for a book definition (in fact, you’ll get no credit if it resembles a book definition).

   a. reference

   b. object

   c. class

2. What’s the difference between class and object?

3. Give an example of a NullPointerException (you may not use the one we used in class).

4. Using class definition to the right, what is the final location (including direction) of karel given the code below?

   ```java
   public class Monster extends Robot {
   public Monster(int st, int ave, Direction dir, int num) {
   super (st, ave, dir, num);
   }
   public void hipHop(int m, int n) {
   for (int i = 0; i < m; i++)
   turnLeft();
   while(n > 0)
   {
   move();
   n--;
   }
   }
   }
   ```

5. What would a `move()` method look like for a class (with superclass `UrRobot`) that wanted to override the `move()`. It would like to redefine the meaning of `move` to mean: drop a beeper at the current location and then go forward 2 steps.

6. Using the following inheritance diagram, write code that would demonstrate polymorphism. You must give 2 different examples using 2 different types of references. You should document your code stating which statements are polymorphic and why.

![Inheritance Diagram]

7. Using the diagram above, which of these are legal (i.e., will compile)?

```java
SportRobot tina = new SoccerRobot();
tina.play();

SportRobot tina = new SoccerRobot();
tina.move();

SportRobot tina = new SoccerRobot();
tina.kickGoal();

SportRobot tina = new SportRobot();
tina.play();
```

```
8. In front of Karel is a straight path of beepers on every corner. Karel’s job is to travel to the end of the path of beepers, and then return along the same path, picking up the beepers as she returns to the corner on which she started. You are to write a single recursive method called followPathAndPickUpOnReturn that accomplishes the task. You may not use any variables in your solutions, and you may assume the following:

- Karel is facing the correct direction
- Karel is on the first beeper of the path
- There are no walls in Karel’s path

//precondition: Karel is on the first beeper on the path, facing along the path, and there are no walls in her path.
//postcondition: All beepers have been picked up, and Karel is back on the starting corner facing any direction.