AP Computer Science Name: _____
Karel Chapter 4 Review

POLYMORPHISM

1. Declare a `BeeperLayer` object with a reference of `carl`. To what object does the reference point?

2. What is wrong with the following code? What kind of error is it?

```
public static void main(String[] args)
{
     SuperRobot ben;
     ben.move();
}
```

3. What does it mean to *initialize* a variable (reference)? Give an example.

4. Given the following code, change the assignment of the reference below five times to a robot and increase the street and avenue by one for each change in assignment.

```
public static void main(String[] args)
{
     UrRobot buddy = new UrRobot(1, 1, North, infinity);




}
```

For Questions #5-6, use the given information:

```
StairSweeper extends UrRobot
MileWalker extends UrRobot
BeeperLayer extends UrRobot

TwoRowLayer extends BeeperLayer
FourRowLayer extends BeeperLayer
SuperMileWalker extends MileWalker
```

5. Draw a diagram that illustrates the relationships between all the classes above with `UrRobot` at the top.

6. Which of the following are acceptable? Explain your answer in one statement.

a. `UrRobot alex = new StairSweeper(1, 1, North, 0);`

b. `MileWalker bud = new BeeperLayer(1, 1, North, 0);`

c. `BeeperLayer cathy = new UrRobot(1, 1, North, 0);`

d. `UrRobot david = new SuperMileWalker(1, 1, North, 0);`

e. `MileWalker eric = new SuperMileWalker(1, 1, North, 0);`

For Questions #7-8, use the given information:

```
StairSweeper extends UrRobot
MileWalker extends UrRobot
BeeperLayer extends UrRobot

TwoRowLayer extends BeeperLayer
FourRowLayer extends BeeperLayer
SuperMileWalker extends MileWalker
```

7. Which of the types of robots from above can be passed into the following method as a parameter?

```
public void doSomethingSpecial(UrRobot someRobot)
{
     someRobot.move();
     someRobot.turnLeft();
     someRobot.turnOff();
}
```

8. Which of the types of robots from above can be passed into the following method as a parameter?

```
public void doSomethingSpecial(BeeperLayer someRobot)
{
     someRobot.move();
     someRobot.turnLeft();
     someRobot.turnOff();
}
```

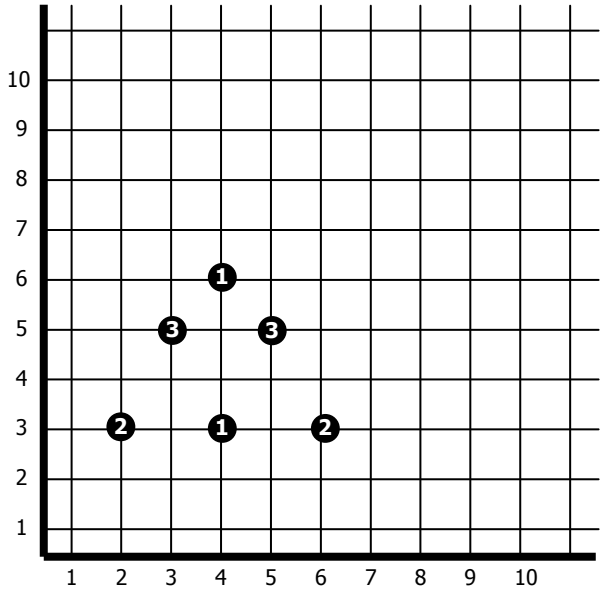
9. What does it mean to override a method? When do you override a method? When is "`super`" used?

10. What does the term polymorphism mean in java?

11. Describe the final outcome of the following code.

```
public static void main(String[] args)
{
  World.setVisible(true);

  UrRobot billy =
     new UrRobot(3, 7, North, 0);
  billy.move();
  billy.move();
  UrRobot bobby =
     new UrRobot( 4, 1, South, 0);
  bobby = billy;
  billy.turnLeft();
  billy.move();
  billy.move();
  bobby.move();
}
```



11. What is an `abstract` class? What must be done if you extend an abstract class in order for that class to be concrete?

12. Assume the class `BeeperPicker` is an abstract class and `TwoPicker`, `ThreePicker`, and `FivePicker` are all concrete classes that extend `BeeperPicker`. Which of the following instantiations are correct?

a. `BeeperPicker jorge = new BeeperPicker(1, 2, East, 0);`

b. `BeeperPicker george = new TwoPicker(1, 3, West, 5);`

c. `FiverPicker helen = new ThreePicker(2, 4, North, 3);`

d. `ThreePicker iggy = new BeeperPicker(3, 5, South, 9);`

13. Given the following codes for `TwoPicker`, `ThreePicker`, and `FivePicker`, factor out the common attributes that tie these classes together and write the abstract class `BeeperPicker`.

```
public class TwoPicker extends UrRobot
{
     // Constructor omitted

     public void get()
     {
          move();
          move();
     }

     public void pickUp()
     {
          pickBeeper();
          pickBeeper();
     }

     public void turnAround()
     {
          turnLeft();
          turnLeft();

     }
}

public class ThreePicker extends UrRobot
{
     // Constructor omitted

     public void get()
     {
          move();
          move();
          move();
     }

     public void pickUp()
     {
          pickBeeper();
          pickBeeper();
          pickBeeper();
     }

     public void turnAround()
     {
          turnLeft();
          turnLeft();
     }
}
```

```
public class FivePicker extends UrRobot
{
     // Constructor omitted

     public void get()
     {
          move();
          move();
          move();
          move();
          move();
     }

     public void pickUp()
     {
          pickBeeper();
          pickBeeper();
          pickBeeper();
          pickBeeper();
          pickBeeper();
     }

     public void turnAround()
     {
          turnLeft();
          turnLeft();

     }

     public void putFiveBeepers()
     {
          putBeeper();
          putBeeper();
          putBeeper();
          putBeeper();
          putBeeper();
     }
}
```

14. What is an interface?

15. How do classes use a specific interface?

16. How are interfaces and abstract classes similar and how are they different?

17. Write an interface called `Moveable` with the methods `changeLocationOfBeepers()` and `teleport()`.

18. A class that extends `UrRobot` wants to override the `turnOff()` method to mean: move, put down a beeper, move, then turn off. Override this method.