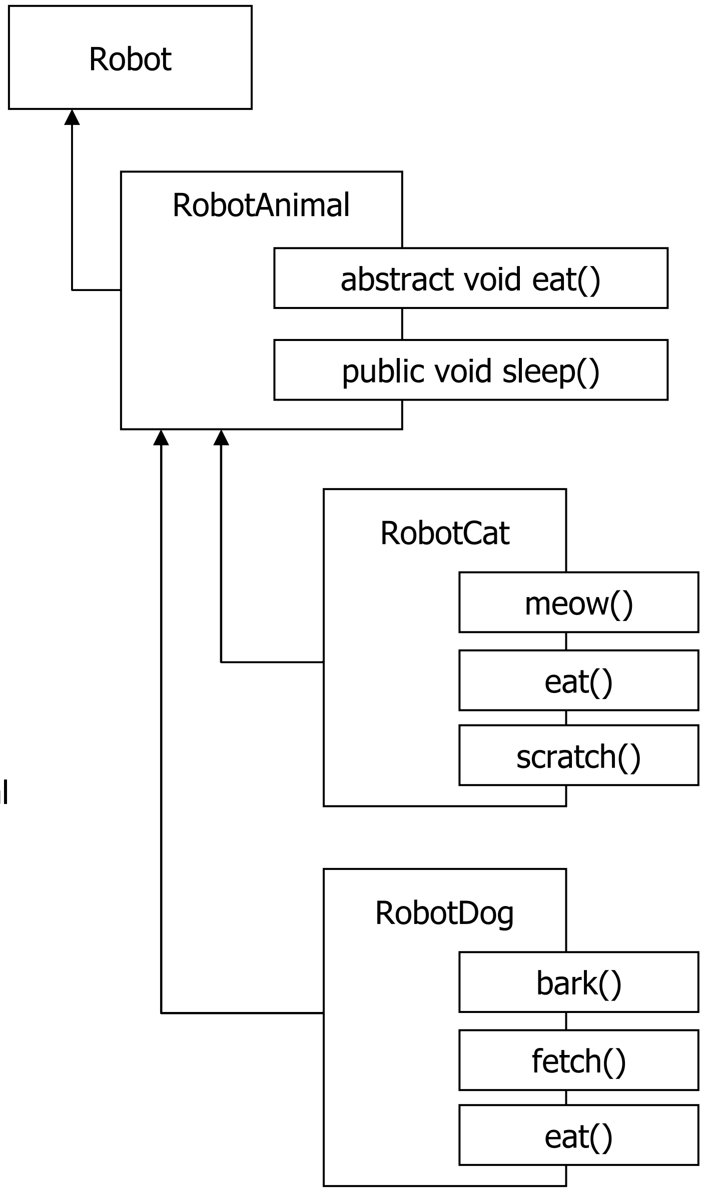
Name: ______________________________________ Date: ___________________ Period: ________

Karel Chapters 4-6 Review

Given the diagram on the right, answer the following questions:

1. What type of class is `RobotAnimal`?



2. Name 8 methods that a `RobotCat` knows.

3. How is polymorphism illustrated?

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

a.
```
RobotCat bob = new RobotAnimal(…);
bob.sleep();
```

b.
```
RobotDog tim = new RobotDog(…);
tim.eat();
```

c.
```
RobotAnimal pam = new RobotCat(…);
pam.turnLeft();
```

d.
```
Robot jon = new RobotAnimal(…);
jon.sleep();
```

e.
```
RobotCat abe = new RobotCat(…);
abe.fetch();
```

5. A class that extends `UrRobot` wants to override the `turnLeft()` method to mean: move forward twice, turn left, and move.

6. Using the concept of `Choreographer`s, write the necessary methods so that all three helper robots move, turn left, and put beepers down at the same time as any `SuperHeroRobot` created.

```
public class SuperHeroRobot extends UrRobot
{
     private UrRobot batMan = new UrRobot(…);
     private UrRobot ironMan = new UrRobot(…);
     private UrRobot spiderMan = new UrRobot(…);

     public SuperHeroRobot(…){…}   //Constructor omitted

     …    //Missing Methods


     public static void main(String [] args)
     {
          SuperHeroRobot superMan = new SuperHeroRobot(…);
          superMan.move();
          superMan.turnLeft();
          superMan.saveTheWorld();
     }
}
```

7. What is wrong with the following main method:

```
public class SuperDuperRobot extends Robot
{
     //constructor omitted

     public static void main(String [] args)
     {
          SuperDuperRobot roger = new (1, 1, North, 0);

          if(frontIsClear())
          {
               move();
               turnLeft();
          }
     }
}
```

8. Write a new predicate that tests whether a robot has more than four beepers.

9. Write a new method called `putBeepersBasedOnDirection()`, where a robot places 1 beeper on its current corner if it is facing north, places 2 beepers on its current corner if it is facing south, moves three times to the right if it is facing east, and turns around if it is facing west. Assume the world has no wall.

10. What is wrong with the following block of code? Assume all methods are defined.

```
// Returns true if the right is clear, false otherwise
public void rightIsClear()
{
     turnRight();
     if( ! frontIsClear() )
     {
          return true();
     }
     turnLeft();
     return false();
}
```

11. Simplify the following, if possible:

```
if (frontIsClear())
{
     move();
}
else
{
     turnLeft();
     move();
}
```

12. Simplify the following, if possible:

```
if (nextToABeeper())
{
     move();
}
else
{
     move();
     turnRight();
}
```

13. Re-write the following block of code using a positive test:

```
if (!nextToARobot())
     return false;
else
     return true;
```

14. Complete the following class, adding any additional methods you might need to help you.

```
public class JumpHarvestAndTest extends Robot
{
//precondition: robot is facing east and standing next to the base of an
//arbitrarily high hurdle
//postcondition: robot is facing east and is standing on the other side
//of the hurdle at the base

public void jump()
{





}

//precondition: there are an arbitrary number of piles of beepers in a
line terminated with a wall
//segment - each pile has an arbitrary number of beepers x, such that
0≤x<infinity
//postcondition: robot has picked up all beepers from the position it
started at up to and including
//the intersection next to the wall

public void harvestAllToWall()
{





}
```

```
//precondition: robot has an arbitrary number of beepers in bag
//postcondition: returns true if rear is clear and the robot has exactly
2 beepers in its bag;
//false otherwise
//note: you may NOT use && or || for this exercise

public boolean rearIsClearAndRobotHas2BeepersInBag()
{









}
```

15. Write a method that takes two parameters, the first tells a robot how many steps to take and the other parameter tells a robot how many beepers to put on each corner it walks on.

16. Write a method that takes one parameter and has a robot put down a number of beepers equal to that parameter, then moves the same number of times, then it puts one less beeper down and moves one less time repeating the process until it has no beepers, in which case it turns itself off.

17. When 7 is passed into this method as a parameter, how many beepers have been put down? (Hint: Make a table keeping track of i and j and the number of beepers)

```
public void someCrazyMethod(int n)
{
     for(int i = n; i >= 0; i--)
     {
          for(int j = 0; j < i; j++)
          {
               putBeeper();
          }
     }
}
```