CS101
Quiz No. 4
Nov. 30 & Dec. 1, 2015
Student Name: Kadir Can Çelik
(Extended on Dec. 6, 2015)

CS101
Quiz No. 4
Nov. 30 & Dec. 1, 2015
Student Name: Kadir Can Çelik

1. Create a class called Circle with the instance variables coordinates x and y, radius and color. For this class implement the following public methods: **1**. a constructor( ) with four parameters one for each instance variable, **2**. another constructor with a parameter of type Circle, **3**. default constructor, **4**. equals( ), **5**. clone( ), **6**. compareTo( ), **7**. toString( ). Please write the methods in the order they are listed above.

Circle.java:

```
public class Circle {

   // Instance variables
   private double x;
   private double y;
   private double radius;
   private String color;

   // A constructor with four parameters, one for each instance variable
   // Note that formal parameter names override instance variable names
   // That's why we are using "this" keyword
   public Circle (double x, double y, double radius, String color) {
      this.x = x;
      this.y = y;
      this.radius = radius;
      this.color = color;
   }

   // Another constructor with a Circle parameter
   // Here we can omit the keyword "this", it is a matter of choice
   public Circle (Circle other) {
      this.x = other.x;
      this.y = other.y;
      this.radius = other.radius;
      this.color = other.color;
   }

   // A default constructor
   // Although you can use any values to initialise instance variables
   // it's probably better to use reasonable values
```

```
public Circle () {
   this.x = 0;
   this.y = 0;
   this.radius = 0;
   this.color = "";
}

// equals method which checks if two circles are identical
public boolean equals (Circle other) {
   boolean result;
   result = (this.x == other.x && this.y == other.y &&
      this.radius == other.radius && this.color.equals (other.color));
   return result;
}

// clone method
public Circle clone () {
    return (new Circle (this));
}

// compareTo method
// As it was not defined in the question clearly, it is possible to
// have different type of comparisons
// Here I found it logical enough to compare the areas
public int compareTo (Circle other) {
   int result;
   double area1;
   double area2;
   area1 = Math.PI * Math.pow (this.radius, 2);
   area2 = Math.PI * Math.pow (other.radius, 2);
   if ( area1 < area2 ) {
      result = -1;
   } else if ( area1 > area2 ) {
      result = 1;
   } else {
      result = 0;
   }
   return result;
}

// toString method
public String toString () {
   String result;
   result = "x: " + this.x;
   result = result + " y: " + this.y;
   result = result + " radius: " + this.radius;
   result = result + " color: " + this.color;
   return result;
```

```
    }
}
```

CircleTester.java:

```
public class CircleTester {

    public static void main (String[] Args) {

        // Creating some Circle objects
        Circle myCircle = new Circle ();
        Circle yourCircle = new Circle (4, 5, 3, "blue");
        Circle hisCircle = new Circle (yourCircle);

        // Checking the equals method
        System.out.println (myCircle.equals (yourCircle));
        System.out.println (yourCircle.equals (hisCircle));

        // Checking the compareTo method
        System.out.println (myCircle.compareTo (yourCircle));

        // Checking the toString method
        // Note that we didn't have to write toString explicitly
        System.out.println (myCircle);
        System.out.println (yourCircle);
        System.out.println (hisCircle);

        // Checking the clone method
        myCircle = hisCircle.clone();

        System.out.println (myCircle);
        System.out.println (yourCircle);
        System.out.println (hisCircle);
    }
}
```

2. Write a class called RentalCar with the instance variables longitude, latitude, renterName, dailyRate (rental cost for one day). For this class implement the following public methods: **1**. constructor( ) with four parameters one for each instance variable, **2**. another constructor with a parameter of type RentalCar, **3**. default constructor, **4**. equals( ), **5**. clone( ), **6**. compareTo( ) - using dailyRate-, **7**. toString( ). Please write the methods in the order they are listed above.

RentalCar.java

```
public class RentalCar {

    // Instance variables
    private double longtitude;
    private double latitude;
    private double dailyRate;
    private String renterName;
```

```
// A constructor with four parameters, one for each instance variable
// Note that formal parameter names override instance variable names
// That's why we are using "this" keyword
public RentalCar (double longtitude, double latitude, double dailyRate, String renterName) {
   this.longtitude = longtitude;
   this.latitude = latitude;
   this.dailyRate = dailyRate;
   this.renterName = renterName;
}

// Another constructor with a RentalCar parameter
// Here we can omit the keyword "this", it is a matter of choice
public RentalCar (RentalCar other) {
   this.longtitude = other.longtitude;
   this.latitude = other.latitude;
   this.dailyRate = other.dailyRate;
   this.renterName = other.renterName;
}

// A default constructor
// Although you can use any values to initialise instance variables
// it's probably better to use reasonable values
public RentalCar () {
   this.longtitude = 0;
   this.latitude = 0;
   this.dailyRate = 0;
   this.renterName = "";
}

// equals method which checks if two RentalCars are identical
public boolean equals (RentalCar other) {
   boolean result;
   result = (this.longtitude == other.longtitude && this.latitude == other.latitude &&
         this.dailyRate == other.dailyRate && this.renterName.equals (other.renterName));
   return result;
}

// clone method
public RentalCar clone () {
   return (new RentalCar (this));
}

// compareTo method
public int compareTo (RentalCar other) {
   int result;
   if ( this.dailyRate < other.dailyRate ) {
      result = -1;
```

```
        } else if ( this.dailyRate > other.dailyRate ) {
            result = 1;
        } else {
            result = 0;
        }
        return result;
    }

    // toString method
    public String toString () {
        String result;
        result = "longtitude: " + this.longtitude;
        result = result + " latitude: " + this.latitude;
        result = result + " dailyRate: " + this.dailyRate;
        result = result + " renterName: " + this.renterName;
        return result;
    }
}
```

RentalCarTester.java:

```
public class RentalCarTester {

    public static void main (String[] Args) {

        // Creating some RentalCar objects
        RentalCar myRentalCar = new RentalCar ();
        RentalCar yourRentalCar = new RentalCar (4, 5, 3, "Ahmet");
        RentalCar hisRentalCar = new RentalCar (yourRentalCar);

        // Checking the equals method
        System.out.println (myRentalCar.equals (yourRentalCar));
        System.out.println (yourRentalCar.equals (hisRentalCar));

        // Checking the compareTo method
        System.out.println (myRentalCar.compareTo (yourRentalCar));

        // Checking the toString method
        // Note that we didn't have to write toString explicitly
        System.out.println (myRentalCar);
        System.out.println (yourRentalCar);
        System.out.println (hisRentalCar);

        // Checking the clone method
        myRentalCar = hisRentalCar.clone();

        System.out.println (myRentalCar);
        System.out.println (yourRentalCar);
        System.out.println (hisRentalCar);
```

```
    }
}
```

3. Create a class called Circle with the instance variables coordinates x and y, radius and color. For this class implement the following public methods: **1**. a constructor( ) with four parameters one for each instance variable, **2**. another constructor with a parameter of type Circle, **3**. multiply(double): multiplies the area of the circle with the amount indicated in the parameter; **4**. clone( ), **5**. distance( ): from center to center, **6**. toString( ); **7**. overlaps( ) for example the circles given below overlap. For the implementation of overlap( ) if there is another method that you can use you have to use the other method. Please write the methods in the order they are listed above.

Circle.java:

```
public class Circle {

    // Instance variables
    private double x;
    private double y;
    private double radius;
    private String color;

    // A constructor with four parameters, one for each instance variable
    // Note that formal parameter names override instance variable names
    // That's why we are using "this" keyword
    public Circle (double x, double y, double radius, String color) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = color;
    }

    // Another constructor with a Circle parameter
    // Here we can omit the keyword "this", it is a matter of choice
    public Circle (Circle other) {
        this.x = other.x;
        this.y = other.y;
        this.radius = other.radius;
        this.color = other.color;
    }

    // multiply method
    public void multiply (double times) {
        radius = radius * Math.sqrt (times);
    }

    // clone method
```

```
    public Circle clone () {
        return (new Circle (this));
    }

    // distance method
    public double distance (Circle other) {
        double result;
        result = Math.sqrt (Math.pow (x - other.x, 2) + Math.pow (y - other.y, 2));
        return result;
    }

    // toString method
    public String toString () {
       String result;
       result = "x: " + this.x;
       result = result + " y: " + this.y;
       result = result + " radius: " + this.radius;
       result = result + " color: " + this.color;
       return result;
    }

    // overlaps method
    // If the sum of the two radii is bigger than the distance between two centes
    // In that case the circles overlap
    public boolean overlaps (Circle other) {
       boolean result;
       double distanceBetweenCenters;
       double sumOfRadii;
       distanceBetweenCenters = distance (other);
       sumOfRadii = radius  + other.radius;
       if ( sumOfRadii > distanceBetweenCenters ) {
          result = true;
       } else {
          result = false;
       }
       return result;
    }
}
```

CircleTester.java:

```
public class CircleTester {
    public static void main (String[] Args) {
       Circle circle1;
       Circle circle2;
       Circle circle3;

       // Testing constructors
       circle1 = new Circle (0, 0, 5, "blue");
```

```
        circle2 = new Circle (circle1);
        circle3 = new Circle (5, 12, 6, "black");

        // Testing toString
        System.out.println (circle1);
        System.out.println (circle2);
        System.out.println (circle3);

        // Testing distance and overlaps
        System.out.println (circle1.distance (circle3));
        System.out.println (circle1.overlaps (circle3));

         // Testing multiply
         circle3.multiply (4);
         System.out.println (circle1);
         System.out.println (circle2);
         System.out.println (circle3);
         System.out.println (circle1.distance (circle3));
         System.out.println (circle1.overlaps (circle3));
    }
}
```

4. Write a class called Line with the ins tance variables for the coordinates of the beginning and ending points, and color (yes a colorful line!). **1**. constructor( ) with all necessary parameters one for each instance variable, **2**. another constructor with a parameter of type Line, **3**. default constructor, **4**. equals( ), **5**. clone( ), **6**. compareTo(): uses their lengths, **7**. distance( ): computes distance from midpoint to midpoint. Please write the methods in the order they are listed above.

Line.java:

```
public class Line {
    // Instance variables
    private double startX;
    private double startY;
    private double endX;
    private double endY;
    private String color;

    // A constructor with four parameters
    public Line (double startX, double startY, double endX, double endY, String color) {
        this.startX = startX;
        this.startY = startY;
        this.endX = endX;
        this.endY = endY;
        this.color = color;
    }

    // A copy constructor
    public Line (Line other) {
        this.startX = other.startX;
```

```
        this.startY = other.startY;
        this.endX = other.endX;
        this.endY = other.endY;
        this.color = other.color;
    }

    // A default constructor
    public Line () {
        this.startX = 0;
        this.startY = 0;
        this.endX = 0;
        this.endY = 0;
        this.color = “”;
    }

    // equals method
    public boolean equals (Line other) {
        boolean result;
        result = this.startX == other.startX;
        result = result && (this.startY == other.startY);
        result = result && (this.endX == other.endX);
        result = result && (this.endY == other.endY);
        result = result && (this.color.equals (other.color));
        return result;
    }

    // clone method
    public Line clone () {
        return (new Line (this));
    }

    // compareTo method
    public int compareTo (Line other) {
        int result;
        double distance1;
        double distance2;
        distance1 = Math.sqrt (Math.pow (startX - endX, 2) + Math.pow (startY - endY, 2));
        distance2 = Math.sqrt (Math.pow (other.startX - other.endX, 2) + Math.pow (other.startY -
other.endY, 2));
        if ( distance1 < distance2 ) {
            result = -1;
        } else if ( distance1 > distance2 ) {
            result = 1;
        } else {
            result = 0;
        }
         return result;
    }
```

```
   // distance method
   // We can obtain the midpoints by summing the coordinates and then dividing by 2
   public double distance (Line other) {
      double result;
      double midPoint1X;
      double midPoint1Y;
      double midPoint2X;
      double midPoint2Y;
      midPoint1X =  (startX + endX) / 2;
      midPoint1Y = (startY + endY) / 2;
      midPoint2X = (other.startX + other.endX) / 2;
      midPoint2Y =  (other.startY + other.endY) / 2;
      result = Math.sqrt (Math.pow (midPoint1X - midPoint2X, 2) + Math.pow (midPoint1Y -
midPoint2Y, 2));
       return result;
   }
}
```

LineTester.java:

```
public class LineTester {
   public static void main (String[] Args) {
      Line line1;
      Line line2;
      Line line3;

      // Testing constructors
      line1 = new Line (0, 0, 3, 4, "blue");
       line2 = new Line (line1);
       line3 = new Line ();

       // Testing equals method
       System.out.println (line1.equals (line2));
       System.out.println (line2.equals (line3));
       System.out.println (line3.equals (line1));

       // Testing compareTo method
       System.out.println (line1.compareTo (line2));
       System.out.println (line2.compareTo (line3));

       // Testing distance method
       System.out.println (line1.distance (line2));
       System.out.println (line2.distance (line3));
       System.out.println (line3.distance (line1));
   }
}
```