ICS4U Module 4: Note & Exercise 2a

Objects can ONLY be
compared using .equals(),
never using ==

The Object Class

- The Object class is the *superclass* of all other classes
- Classes such as Circle, and String, are *subclasses* of Object
- Subclasses *inherit*, or receive the methods of its superclass
- The object class includes methods for comparing objects (equals()) and representing an object as a string (toString())
- A subclass typically contains its own version of the equals() and toString() methods o This is done by redefining the superclass method (called *overriding*)

Updating our Circle class:

```
/**
 * Determines if the object is equal to another Circle
 object. * pre: c is a Circle object
 * post: true has been returned if the objects have
 * the same radii. False has been returned otherwise
 */

public Boolean equals(Object c) {
      Circle testObj = (Circle)c;

      if (testObj.getRadius() == radius) {
            return(true);
      } else {
            return(false);
      }
}
```

The equals() method requires
an Object parameter.
As a result, it must be cast
as the appropriate type

```
/**
 * Returns a String that represents the Circle object.
 * pre: none
 * post: A String representing the Circle object has been
 returned. */
```

```
public String toString() {
      String circleString;

      circleString = "Circle has radius: " + radius;
      return(circleString);

 }
```

The concatenation operator + also invokes the toString() method of an object.

ICS4U Module 4: Note & Exercise 2a

Programming Exercises:

a) Modify the Circle class to override the equals() and toString() methods, as shown in the previous section. Modify existing client code to test the new methods. (Submit this code)

```
public class Question1 {

     public static void main(String[] args) {
           // TODO Auto-generated method stub
           //Creates object of class Circle
           Circle a = new Circle ();
           Circle b = new Circle ();

           //Initializes the radius of both objects
           a.radius = 10;
           b.radius= 10;

           //Compares the fields of both objects
           boolean compareRadius;
           compareRadius = a.equals(b);
           System.out.println(compareRadius);

           //Outputs the radius of both objects
           System.out.println(a.toString());
           System.out.println(b.toString());

     }

}
```

```
class Circle {
     double radius;
     /*
      * Compares if this object's field equals to the field of
object passed in as a parameter
      * pre: takes circle object as a parameter
      * post: returns true if object fields are equal and false
if object fields are not equal
      */
     public boolean equals(Circle x) {
           if (this.radius == x.radius)
                 return true;
           else
                 return false;
     }

     /*
      * Returns a String that represents radius of the circle.
      * pre: none
      * post: A String representing the Circle object has been
      */
     public String toString() {
           String circleString;
           circleString = "Circle has radius " + radius;
           return circleString;
     }
}
```

b) Modify the Rectangle class to override the equals() and toString() methods. Two rectangles are equal when they both have the same length and width. Modify the existing client code to test the new method. (DO NOT submit this code yet)

```
public class Question2 {



   public static void main(String[] args) {

         // TODO Auto-generated method stub
```

```
        Rectangle a = new Rectangle ();

        Rectangle b = a;

        a.length = 10;

        a.width = 5;

        System.out.println(a.equals(b));

        System.out.println(a.toString());

        System.out.println(b.toString());


   }


}


/*

 * Class for rectangle that contains

 * the field length and width

 */

class Rectangle {

   double length;

   double width;


   public boolean equals(Rectangle x) {

        if (this.length == x.length && this.width == x.width)
```

```
                return true;

          else

                return false;

    }



    public String toString () {

          String rectangleDimension = "Rectangle length: " + length +
    "\n" + "Rectangle width: " + width;

          return rectangleDimension;

    }

}
```

c) Modify the Coin class to override the toString() method so that it indicates whether the coin is face up or face down. For example, "The coin is face up." Modify existing client code to test the new method. (Submit this code)

```
public class Question3 {



    public static void main(String[] args) {

          // TODO Auto-generated method stub

          Coin a =new Coin();

          System.out.println(a.toString());




    }
```

```
}

class Coin {

   int flipcoin;

   String face;

   /*

    * This method randomly generates a coin flip

    * pre: none

    * post: returns the coin face generated by flipcoin (heads/tail)

    */

   public int flipCoin() {

         flipcoin = (int)Math.round(Math.random());

         return flipcoin;

   }


   /*

    * This methods returns the flipcoin

    * pre: none

    * post: returns face generated from flipCoin method

    */

   public String toString () {

         String face;
```

```
            if (flipcoin == 0)

                  face = "Tails up";

            else

                  face = "Heads up";



            return face;

      }

   }
```

Add your code, including the client code, to the Google Doc: “ICS4U – Activity Submission Form”.