Inheritance exercises 5 – Coin Class and equals() method

The Coin class was mentioned in the lecture material when we spoke briefly about the Object class in Java.

Here is the code that was presented:

```
//Used to illustrate the use of the overridden equals method
public class Coin {
    private double value;
    private String name;

    public Coin(double v, String n) {
        value = v;
        name = n;
    }

    //When we override, we must maintain the same signature as Object superclass
    //The argument is of type Object BUT it CAN BE ANY SUBCLASS ALSO!!!!
    @Override
    public boolean equals(Object otherObject) {
        //Even though we assume that the argument is of type Coin, we must cast
        //the object so that we can use it
        Coin other;
        //other = (Coin) otherObject; //could do it like this
        //The code below is actually safer to cast than above - why?
        //Because in theory, the argument could be any object reference
        //(String,BankAccount, etc.)
        if (otherObject instanceof Coin) {
            other = (Coin) otherObject;
        }
        else {
            return false; // the two objects cannot be equal
        }

        if ((this.value == other.value) && //Note == for primitives
                (this.name.equals(other.name))) {
            return true;
        }
        else {
            return false;
        }
    }


    public String toString() {
        return "[Name: " + name + ", Value: " + value + "]";
    }
}
```

Part 1

Create a tester class which will create some (three, say) `Coin` objects, e.g.

```
Coin c1 = new Coin(10, "Ten Cent");
```

Some of the coins should be identical (in terms of having the same values in their fields) and some should differ.

Use `if-else` statements and the `equals()` method to test some coins for equality and inequality, i.e. should print out whether the coins are equal or not in each test case.

Part 2

Comment out the `equals()` method.

Now, re-run the tester.

Observe what is happening. Can you speculate on why? Find out (google it, perhaps) what the equals() method from the `Object` class actually does and make sure that your observed output makes sense now.

Part 3

Create an `ArrayList` of `Coin` objects and add a number of objects into it, e.g. `coins.add(new Coin(10, "Ten Cent");`

Use the `Arraylist's contains()` method to search for a `Coin` (try to search for one that is contained in the list and then one that isn't)

See if you can figure out how to remove a searched for `Coin` if you find a match. Print out the contents of the list to verify.

Part 4

Comment out the `equals()` method in `Coin`. Now, use IntelliJ's code generation to provide its version of `equals()`. Note: it will also provide a `hashCode()` method – they tend to come as a pair but don't worry too much about that right not.

Examine the differences between the two versions of the `equals()` method. Try to understand what each is doing – we'll chat about it in class.