Polymorphism, *super,* constructors, *equals, instanceof*

Two important concepts:

1. Strongly Typed *at compile time*: The list of methods you can call depend on the type(class) of the reference.

2. Polymorphic *at runtime*: What code will be executed depends on the actual class of the object at runtime. Extreme case: no object -> "NullPointerException'

#1 Using **instanceof**:

```
  Object x = new Point(1,2) // defined below
  Object y = new Integer (5);
  Object z = new LabeledPoint("Hi",2,3);
//LabeledPoint extends Point to include a label.

  (x instanceof Point) -> true
  (y instanceof Point) -> false
  (z instanceof Point) -> true
  (x instanceof LabeledPoint) -> ?
```

#2 Which of the following are valid?

```
  Object o = new LabeledPoint("Hi",2,3);
  Point p = (Point)o;
  LabeledPoint lp = (LabeledPoint)o;

  o.toString();
  p.toString();
  lp.toString();

  o.getX();
  p.getX();
  lp.getX();
```

#3

```
public class Point {
      double x,y;
      double getX() {return x;}
      double getY() {return y;}
      String toString() {return "("+x+","+y+")";}
```

//Write a constructor that takes 2 doubles to initialize the point to the given x,y position:

#4 Point class continued:

// Write an equals instance method that takes an object reference. If the given object is of type 'Point' check that both points have the same x,y values.

}

#5 Write a public **class** method 'createPoints' that takes an integer parameter 'N' (the number of points to create) and returns an array of Points. Initialize each Point with a random (x,y) position.

#6 Write a public class method 'count' that takes an array of points and returns the number of points which have x>y

```
  public class PointMaker {
```

}

#7 Write a class method main(String[] args) that creates 1000 random points and then prints the number of points that have x>y. Also print the first point.

```
public class PointTester {
```

}

Don't forget Turings Craft!

For example - (Turings Craft #20748)
Given an existing class, BankAccount, containing:

- a constructor accepting a String corresponding to the name of the account holder.
- a method, getBalance, that returns a double corresponding to the account balance.
- a method withdraw that accepts a double, and deducts the amount from the account balance.

Write a class definition for a subclass, CheckingAccount, that contains:
A boolean instance variable, *overdraft*.
A constructor that accepts a String and a boolean. The String paramteter is used in the invocation of the superclass (BankAccount) constructor, while the boolean is used to initialize the overdraft instance variable.
A method, hasOverdraft, that returns a boolean. hasOverdraft returns true if the account supports overdraft.
A method, clearCheck, that accepts a double and returns a boolean. clearCheck will determine if the amount (of the check) can be cashed--this will be the case if the amount is less than the balance in the account, or if the account allows overdraft. If the check can be cashed, clearCheck returns true, and also calls the withdraw method to update the account balance; otherwise, clearCheck returns false.

Iterators