Introduction to Programming

with Java, for Beginners

Inheritance

Example: Bot and BetterBot

```java
public class Bot{
    private int x;
    private int y;
    public int getX() { .. }
    public int getY() { .. }
    public void eatDot(){ .. }
    public void move(){ .. }
    public void turnLeft(){ .. }
}

public class BetterBot extends Bot{
    public void turnRight(){
        turnLeft();
        turnLeft();
        turnLeft();
    }
}
```

Inheritance

One of key concepts of OOP

- A hierarchical relationship among classes
- Establishes a superclass/subclass relationship
- Establishes "is a" relationships
  - e.g. a BetterBot "is a" Bot

Benefits:

- Reusability of code
  - Put code in one class, use it in all the subclasses
  - Revisions only needs to be done in 1 place
- Polymorphic code (works on "many forms")
  - Write general purpose code designed for a supertype that works for all subtypes
- Polymorphic data structures
  - E.g. an array of a supertype can hold objects of any subtype

Inheritance Tree

Java has single inheritance; each node has one parent
(Except for Object which has no parent)
C++ has multiple inheritance, but it's messy
The Object Class

All classes inherit from the Object class
- The Object class is the root of the class hierarchy
- When we create a new class, "extends Object" is implied

    public class Car {
        
    }

    public class Car extends Object{
        
    }

The Object class has several methods which all object inherit, most notably:
    toString() and equals()
Type Rules

A reference variable of type t may hold a value of its own type or any subtype (but not of a supertype).

Given the following variable declaration:

    Bot b;

Which of the following assignments are valid?

```
    b = new Bot();
b = new String();
b = new BetterBot();
b = new Object();
```

How about these?

```
    BetterBot bb;
    bb = new Bot();
    bb = new String();
    bb = new BetterBot();
    bb = new Object();
```

The “instanceof” Operator

```
> Bot b = new Bot();
> b instanceof Bot true
> b instanceof Object true
> b instanceof BetterBot false
> BetterBot bb = new BetterBot();
> bb instanceof BetterBot true
> bb instanceof Object true
> bb instanceof Bot true
> Bot bbb = new BetterBot(); //a variable can store a subtype
> bbb instanceof BetterBot() true
> bbb instanceof Bot true
```

Another Example

```java
public class Dog{
    private String name;
    private int age;
    Dog(String dogName, int dogAge){
        name = dogName;
        age = dogAge;
    }
}
```

class BetterDog extends Dog{

    public BetterDog(String name, int age){
        super(name, age);
    }
}
```

Constructors and Inheritance

```
> BetterDog d = new BetterDog();
```

When an object is created, its constructor is called

- But first, a constructor from its highest ancestor (Object) is called, then the next highest (Dog), then its own (BetterDog).
- The default behavior is to that the default (no-argument constructor) is executed

A constructor can explicitly call its parent's (its superclass') constructor by making a call to super()

```java
class BetterDog extends Dog{
    public BetterDog(String name, int age){
        super(name, age);
    }
}
```