Domain 4: Object Oriented Programming Videos

Class

→ base definition of an object
- when defining a class, you assign the property values to that class
- each property must have a data type
- class name is first capital NOT camel

- Constructor
  - method w/ same name as class
  - defines how object is being created
  - parameter must have diff name
  - multiple constructors are possible
- ex.

```
static class Tennis Racket {
    String   name;
    int      tension;
    boolean  strung;
    String   mains;
    String   crosses;

    Tennis Racket ( String name Val, Boolean Strung) {
        name = name Val;
        Strung : Strung;
    }

    Tennis Racket (String name Val, boolean Strung, String Mains, String crosses, int tens) {
        ...
    {
}
```

- in this example there are 2 constructors, if false is passed for strung then blank is assigned to rest.

Class (continued)

- more than 1 class per file is permitted
- initial class name must match file name
- if class is made public, it can be accessed from other files
- per java file only 1 class may be public nonstatic

this

- can be used to assign default values
- must have existing class
- ex.

-
  ```
  static class Racket {
      String name;
      String cross;
      String main;
      int tension;

      Racket (String nameV, String cV, String mV, Int tenV) {
          name = nameV;
          cross = cV;
          main = mV;
          tension = tenV;
      }

      Racket () {
          this ("Technifibre Tfight 300 w/ Leather Grip", "Wasabi X", "Wasabi", 49);
      }
  }
  ```

- this example lets someone setup their own racket
- if no values are passed, then my racket values are assigned

Classes (continued)

Inheritance

- this allows a class to copy blueprint (properties, methods, ...) from other classes, then add more properties
- ex.

```
public static Weight {
    int grams;
    String metal;
    String location;
    Weight(int g, String m, String l){
        grams=g;
        metal=m;
        location=l;
    }
}

static class Modded Racket extends Racket {
    Weight [] weights = new Weight [][];
    String [] oMods = new String [][];
    Modded Racket (String nV, String cV, String mV, int tV, Weight[] wV,
                                    String [] oMV ){
        name = nV;
        cross = cV;
        main = mV;
        tension = tV;
        weights = wV;
        oMods = oMV;
    }
}
```

- Key thing to notice is that name, cross, main, tension are all assigned values yet aren't defined in Modded Racket, they are defined in Racket

Class (contived)

Overriding

- change to a default method used in a parent class
- must use @ override
- ex.

```
class Animal {
   void sound () {
      System.out.println ("Animal Sound");
   }
}

class Dog extends Animal {
 @Override
 void sound() {
    System.out.println("Bark");
 }
}
```

Class Data Members

private → only withen class

public → accessible from anywhere

protected → this class and other package members

Data Members

instance → variable that is changed per class instance, also a new memory space per instance

static → one memory space per call, or constant data members

- are generally set but can be changed

final → never be changed

- must be static final

Encapsulation → hiding information aka making smth private

Methods

- preform actions
  → often in the form of calculations

Scope

- put (private/public/protected) before static before return type then name, then args, then block

Class (continued)

instantiation → creating a new instance of a class
initialization → creating blueprint of a class

null → can be used for init data members