**AP Computer Science - Introduction to Classes**

At this point, we have covered the major topics of computer programming in Java. The next big step is to use the foundation that we've built to create customized data structures called classes. This process is also called Object Oriented Programming. What we are going to do is build programs like we've already done, but make them more modular and easy to use, especially for anyone that wasn't a part of writing the code.

What is a class?

You've already been working with classes already in AP Computer Science. A class is a usable, programmable object that someone created to make writing code easier for all of us. Take the String class for example. Someone wrote the code for a String class and embedded in it a bunch of methods and data fields that we can use to make text program way easier than if we had to work with arrays of characters.

Why use classes?

* We can create multiple instances of a class, saving the declaration of multiple sets of variables.
* They are easier to read and code from.
* They are easily exported via an import command.

The Structure of a Class

A class file, at first glance will look like other java programs that we've written, but you will see it is missing a main method. In other words, the class files that we build will not run by themselves. Think about the String class. It is a resource that we can use in our programs, but it doesn't run by itself. It is merely a tool that we can use to more easily handle/manipulate text.

Think of a class as a blueprint for what the object will look like. We will define data fields to store the properties that make up that object. We can also create methods to retrieve or change the data fields of a class.

For this example, we will be creating a Monster object:

**Step # 1 - Data Fields**

* The data fields are the individual properties that hold the data that is relevant to the object created.
* Any data label that is defined using the static keyword is shared between all instances of the class.
* A ***static*** data field means the value stored will be shared between all objects of this type.
* A ***private*** data field means that it cannot be accessed directly by other objects (from outside the class).

**Example # 1** - Here the data fields are not assigned any values.

**public** **class** Monster {

 private String name;

 **int** health, maxHealth, attack, defense, gold;

 **boolean** dead, isBoss;

 **static** **int** *numMonsters*;

**Example # 2** - Alternately, we can assign default values for the data fields.

**public** **class** Monster {

 String name = "Vampire";

 **int** health=25;**int** maxHealth = 25; **int** attack=10; **int** defense=5, **int** gold=0;

 **boolean** dead= **false**; **boolean** isBoss= **true**;

**Step # 2 - Constructors** - Constructors are used when an object is created or ***instantiated***. You can have multiple constructors for a given object, based on what information is provided when the object is created.

**Example # 1** - No-Arg Constructor - This constructor receives no data when the object is created. Thus the object created will receive the default values for its data labels.

 **public** Monster() {

 name = "Orc";

 health = 20;

 maxHealth = 20;

 attack = 3;

 defense = 2;

 dead = **false**;

 isBoss = **false**;

 gold = 1;

 }

To create a Monster object using this constructor:

Monster monster1 = new Monster(); The name of the Monster is ***monster1***.

**Example # 2** - Constructor with received values - This constructor receives 3 pieces of data: A String and two integers, similar to how we can have a method receive values:

**public** Monster(String n, **int** a, **int** d) {

 name = n;

 attack = a;

 defense = d;

 //Now we assign values to fill in the rest of the data fields.

 maxHealth = 10;

 health = 10;

 gold = 2;

 dead = **false**;

 isBoss=**false**;

 }

To create a Monster object using this constructor:

Monster monster2 = new Monster("Goblin", 12, 3);

The name of the Monster is ***monster1***. It will have a name of "Goblin", an attack of 12 and a defense of 3. The rest of the data fields are assigned the values shown above.

\*Understand that in separate classes, called ***runners***, is where we will be creating instances of the monster class.

**Step # 3 - Methods** - There are several methods that are created when we build custom classes:

Getter (or Accessor) Methods - These methods will return the value stored in the class's data fields.

Setter (or Mutator) Methods - These methods will change the value stored in the data fields.

**Method keywords**

private – These methods are only usable by the class itself.

public – these methods are usable by classes outside the class itself.