 

**Advanced Placement Computer Science**

[**Shenendehowa HS**](http://www.shenet.org/shen-high-school/)[**mr Hanley**](http://hanley.co.nr)

**Unit 4: Object Oriented Programming**

**Lesson: ClassesFILLED\_OUT**

***Last Updated:*** *10/24/2017*

Lesson: Parameter Passing Mechanisms

*Last Updated: 100/11001/1100*

**Advanced Placement Computer Science**

**Unit 4: Object Oriented Programming**

Lesson: OO Programming

*Last Updated: 11/1/2013*

Modern programs are often organized into separate **classes** each with its own responsibilities.

This began in the 1980’s with the desire to recreate more **reusable components**

Classes act as software blueprints, allowing programmers to use some classes to create multiple objects, which contain their own **identity** and **data**

While there is only one class, one can create multiple **objects or instances**

|  |  |
| --- | --- |
| Class | Object |
| One definition, contains logic that all objects share. Sometimes called behaviors. | **Can have multiple instances, each has its own unique data. Share all behaviors via the class** |
| Some classes are built on top of existing classes, this is called inheritance | **Executing programs use objects primarily, not classes. Objects send each other messages.** |
|  | **State of an object = values of its instance variables** |

How does a typical class manage information? There are 4 things inside classes

public class SomeClass {

 //Variable Section, typically private so only this class can modify

//These are called instance variables
 **private int age;**

**private String name;**
 //Constructors, special methods that **instanciate** objects of this type

 //NOTE: **constructors NEVER have a return type**

 public SomeClass() { **//o arg constructor**

**age = 0;**

**name = “”;**

}

 public SomeClass(String nm) {**//1 arg constructor**

**age = 0;**

**name = “”;**

}

 //Accessors – must be public

 **//Accessors give read only access to the private variables inside of an object
 public int getAge() {**

 **return age;**

**}**

 //Mutators – must be public

 **//Mutators change the state of the object in some way
 public void setAge(int a) {
 if (a > 0) {
 age = a;**

 **}
 }**

}
public class UserOfSomeClass {
 public static void main(String[ ] args) {
 SomeClass s1 = new SomeClass(); //instanciate an object called s1
 s1.setName(“Arthur”);
 s1.setAge(16);
 System.out.println(s1.getAge());
 //Use the overloaded constructor to create an object for s2 named “Sara”

 //Set her age to 17
 //Print out her information
 **SomeClass s2 = new SomeClass(“Sara”);**

 **s2.setAge(17);**

 **System.out.println(s2.getName() + “ “ + s2.getAge())**
}

Access specifiers

**private**

**Only available to members of the class (ie methods defined INSIDE this class)

public
Available to both class members and other classes

protected
Only available to child classes (will learn this later in the year)**What is a constructor?
**A method with the exact same name as class and NO return type**

**Creates new instances of a class by setting up the state variables**

What is a member variable?
**A member variable is a global variable declared inside of a class**

**It stays around as long as the object does**

**It holds the state of the object**

What are overloaded methods?
**Overloaded methods have the same name but different parameters**

**They are common for constructors, where there are different options specified when starting up new objects of a particular class**

For example, the following structure may apply to a java program;

FILE: Client.java
Contains: main method, which will use 3 Contacts called c1, c2 and c3

main() starts the program, creates three objects and displays their information

FILE: Contact.java

Contains: no main method, has variables which pertain to a cell phone contact

**Had students do this example in class**

**public class Contact { //Represents a cell phone contact (Name and Phone Number)**

1.
2.
3. private String name, phoneNum;
4.
5.
6. public Contact(){
7. name = “”;

 phoneNum = “”;

}

1. //Accessors
2. public String getName(){
3. return name;
4. }
5. public String getPhoneNum(){

 return phoneNum;

1. }
2. //Mutators
3. public void setName(String s){
4. name = s;

 }

 public void setPhone(String ph){

 phoneNum = ph;

 }

}

 }

public class Client {

 public static void main(String[] args) {

 Contact c1 = new Contact();
 c1.setName(“Liam Morris”);
 c1.setNumber(“123-555-1212”);

 }

}