**Encapsulation**

Hiding implementation from external users. Keeps variables of a class hidden, only accessible through methods defined in said class. Known as data hiding. This can be implemented using what is known as Access Modifiers (Public, Private, Protected)

Example:

class Customers

{

private String name;

public String getName();

{

return name;

}

public void setName(String name);

{

this.name=name;

}

}

**Abstraction**

This is the process of hiding complicated implenmentaion details and a baisic functional interface is given. Unnecessary data/code will be removed. An easy way to think of this is that it gives you what it does and NOT how it does it.

Example: This is done by creating an abstract class.

**Inheritance**

This is a way of linking classes together. A subclass (also known as a child class) will extend to a superclass (also known as a parent class). A class can only extend to one other class but a class can be extended to many other classes thus creating a hierarchal structure. (Just like a child can have one mother, but a mother can have many children). From the example below, an easy to to understand, is that an Animal is the parent class, and can get information from both the classes of Dog and Cat (child classes), but this cannot be done the other way around.

Example:

class Animal

{

….

}

class Dog

{

…

}

class Cat

{

…

}

**Polymorphism**

The ability for an object/variable to take on multiple forms. Often used when a super class calls for information from a child class.

Example: if we have a parent class called Transport. This is inherited by 3 child classes; Car, Plane, Boat. For the parent class, a method of move is called upon, and for each of the child classes they may have different methods to implement how to get this method of move. So if the method move was called for a plane, then fly will be returned. For a car, drive will be returned. Finally, for boat, sail will be returned.