Encapsulation

Java encapsulation is a form of capsulizing all of the components into one seamless mechanism. Within encapsulation, data hiding is used in a form of variables from a class are discussed from any former classes, and are only accessed through methods of the current class. Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, abstraction and polymorphism, this is a mechanism of wrapping variables and code acting on methods together as a single unit. You can create a fully encapsulated class in java by making all the data members of the class private, also use setter and getter methods to set and get the data involved in the process. Encapsulation can be described as a protective barrier that would prevent code and data being accessed outside the class. Access to the data and code is very controlled for an interface. The main objective for encapsulation is the meaning to modify implemented coding without faulting the code, encapsulation gives out flexibility to the coding.

Encapsulation is essentially the capsulizing process of data, methods and classes. For example, a pill is filled with small chemical bonds, however it acts as one seamless pill, which is what the end-user sees. This is much like an Object orientated program, it is built up of classes, methods, functions, variables and more, however they all seamlessly interact and engage with each other through the use of packages and classes. However, it does it so seamlessly that it acts as a single unit, like a single thing doing everything, however it isn’t and it’s made up of several intricate things which are hidden away from the end-user. This is what is known as data hiding within OOP. It is what makes encapsulation what it is, capsulizing the numerous classes and objects into one seamless product.

Inheritance

Inheritance is the function within an OOP language in which one class can inherit the methods, functions and variables from its parent class. It works in the similar fashion of that of a child and a parent in which the child inherits certain things from their parents such as 2 legs, 2 arms, eyes, mouth and ears, however the child can redefine these things such as the fact they have eyes but can change the eye colour. This is similar within an OOP language as you can define parent and child classes in which the child class inherits the functions from its parent but you can redefine certain things such as the variables within the child class. For example, an OOP program such as a game will have enemies, instead of re-typing the code for each type of enemy you have you can use inheritance which allows a programmer to re-use code with ease. You would initially create an enemy class (EntityEnemy.java) which would hold methods such as how to move around when no one is near, what to do when it sees a player and how it appears on the screen or the fact that I can hold a texture file. It would have variables such as its health and damage stats. However, this is your main enemy class and you could have 5 different types of enemies that all have health, all die and all do something when they die, these are things which would be stored within the main enemy class. Within their own class however they would hold or override variables such as stronger enemies have greater health or explode upon death or do something slightly different than other enemies, but they would all still hold the same data of that they have health, can move around and can get hurt and give out damage.

Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object. Java object that can pass more than one test of IS -A is considered to be polymorphic. Java objects are polymorphic since any object will pass the test for their own type and for the class object. This is important to know as the only possible way to access an object is through a reference variable. A reference variable can be of only one type, the type of the reference variable would determine a method that could invoke the object, a reference variable can be declared as a class or interface type.

Abstraction

Abstraction deals with ideas rather than events, once considered a sent email within what happens next as soon as it is sent, the email server called the protocol is hidden from the user. General email is sent you would need content and mention details such as the email address of the receiver and then click send, for object orientated programming such as the implementation details abstraction is the process for hiding it. Functionality would be provided for the user for example he user would have information on what the object does, instead of how the object do it.