**DEFINE THE FOLLOWING KEY OBJECT ORIENTED PROGRAMMING (OOP) PRINCIPLES WITH EXAMPLE:**

1. ENCAPSULATION

In Java a program is made up of data (variables) and methods (functions). A method is a group of statements group together to perform a specific task. The data represents the information of the program as well as holding the program state. The statements manipulate the data.

Encapsulation is an OOP principle or concept that bundles or wrap the code (data and methods) together as a single unit called class. A key aspect of encapsulation is protecting these data and methods from external misuse or abuse, commonly referred to as **data hiding**. The terms Encapsulation and data hiding are commonly used interchangeably.

In Java Encapsulation is achieved by

* Declaring the variables of a class as private. This private modifier means that only the class that contains the variables can access them. In other words external classes cannot access the variables declared private (encapsulated) in a class.
* Providing getters and setters to view and modify the variable values

Example

public class Box{

private int width;

private int height;

public int getWidth(){

return width;

}

public int getHeight(){

return height;

}

public int setWidth(int newWidth){

width = newWidth;

}

public int setHeight(int newHeight){

height = newHeight;

}

}

//The private members of the Box class can be accessed from the class below

public class TestClass{

public static void main(String args[]){

Box box = new Box();

box.setHeight(5);

box.setWidth(5);

System.out.println("Height: " + box.getHeight() + "Width: " + box.getWidth() );

}

}

1. INHERITANCE

In real world people from the same family have features or characteristics that are common amongst them. However as you take a closer look it could be observe that each member of the family would still have characteristics that is distinct from the rest of the family members. Also in a company all workers are employees. Employees could be distinct from each other by their various roles or jobs they perform. An employee can be a manager, programmer, accountant etc.

The concept of inheritance is to have a base class or parent class which contains the properties (variables) and functions that are common in a relationship and let their sub-classes also called derived classes inherit these common features. From the practical example of a company above ‘Employee’ will be the base class and ‘Manager’ or ‘Programmer’ or ‘Accountant’ will be the derived class.

Inheritance in Java allows one class to acquire all the properties and behaviours of the parent class. The idea is that you can create a new class that are built upon existing classes. When you inherit from an existing class you can reuse all variables and functions from the parent class. The child class (derived class) can add new variables and methods that makes it different from the parent class.

In Java the extent keyword is used to make a new class inherit from another class.

Example

// Parent class Employee

public class Employee{

float salary;

public float getSalary(){

return salary;

}

public setSalary(float newSalary){

salary = newSalary;

}

}

//Sub-class/derrived class Accountant

public class Accountant extends Employee{

float bonus;

public float getBonus(){

return bonus;

}

public setBonus(float newBonus){

bonus = newBonus;

}

}

/\*\* Demo: From this class the accountant object is used to call functions setSalary and getSalary which was defined in the parent class Employee. \*/

public class TestClass{

public static void main(String args[]){

Accountant accountant= new Accountant();

accountant.setSalary = 1000.99;

accountant.setBonus = 50.99;

System.out.println("Salary: " + accountant.getSalary() + "Bonus: " + accountant.getBonus() );

}

}

1. POLYMORPHISM

Polymorphism is the ability of an object to take on many forms. It is made up of the two Greek words ‘poly’ which means many and ‘morphs’ which means many forms. In Java polymorphism is divided into two parts: method overloading and method overriding.

An example of polymorphism is referring the instance of subclass with reference variable of a super-class. When the reference variable (of the super-class) is used to call a method it is the method of the subclass that will be called during runtime.

EXAMPLE

public class Animal{

void sound(){

System.out.println('sound of animal');

}

}

public class Cat extends Animal{

void sound(){

System.out.println('meoow');

}

}

//TEST CLASS

public class TestClass{

public static void main(String args[]){

/\*\*upcasting\*/

Animal animal = new Cat();

animal.sound();

}

Note:

Even though animal is a reference of super class Animal it is the sound method in the sub class Cat that will be called at runtime.

1. ABSTRACTION

In OOP abstraction is the method of hiding away implementation details from the user, only required functionality would be made available to them. The user will only know about the functionality of an object instead of knowing how it does it.

Example on my smart phone when I make a call the processes that goes on to make the connection are abstracted away from me.

In Java Abstraction is achieved using Abstract classes and Interfaces.

An abstract class may contain abstract methods. Any class inheriting the abstract class must either override the abstract class or provide implementation.

**EXAMPLE**

/\*\* Abstract class Employee with one abstract method calculateTax

public abstract class Employee{

private

public abstract float calculateTax();

}

/\*\* subclass Accountant extends abstract class Employee and must provide

implementation for abstract method calculateTax

\*/

public class Accountant extends Employee{

public float calculateTax(){

//tax calculation goes here

}

}