**Java Full Stack Training - onMobile Client**

**Day 1 - Core Java - 04-09-2019**

**C :**

**#include<stdio.h>**

**void mno() {**

**xyz();**

**}**

**void abc() {**

**mno();**

**}**

**void main() {**

**printf("Welcome to C");**

**abc();**

**}**

**OOPs :**

**object : Any real world entity.**

**Ex:**

**properties(State)----> have ---> fields/variables**

**Person**

**behaviour ------>do/does --- function/methods**

**Bank**

**Animal**

**Car**

**etc**

**class : Blue print of object or template of object or collection of object which have same properties and behaviour or use-defined data type which help to describe the objects.**

**syntax**

**class ClassName {**

**properties**

**behaviour**

**}**

**class Person {**

**String name;**

**int age;**

**void sleeping() {**

**}**

**void teaching() {}**

**}**

**Person p1 = new Person();**

**C + OOPs = C++ + PIEA**

**Java .net**

**Java : Java is a pure object and platform independent programming language.**

**Java 1.0, 1.2, 1.4, 1.5,1.6,1.7 ,**

**1.8, 1.9, 1.10, 1.11, Java 12**

**Data Types : Data types is a type of data which tells what type of data it can hold.**

**2 types**

**1. Primitive Data types : It is use to store only value**

**2. Non primitive data types ore reference data type: it is use to store value as well as reference of another data type.**

**Primitive data types**

**8 types**

**1. byte 1 byte**

**2. short 2 byte**

**3. int 4 byte**

**4. long 8**

**5. float 4**

**6. double 8**

**7. char 2**

**8. boolean 1 bit**

**class Demo {**

**public static void main(String args[]) {**

**int a=10;**

**System.out.println("Welcome to Java ");**

**System.out.println(a);**

**System.out.println("Value of a "+a);**

**System.out.printf("Value of a = %d\n",a);**

**}**

**}**

**Non Primitive data types**

**1. array**

**2. class (pre-defined or user-defined)**

**3. interface (pre-defined or user-defined)**

**4. enum (pre-defined or user-defined)**

**types casting : converting one data type to another data types is known as type casting.**

**2 types**

**1. Implicit type casting**

**2. explicit type casting**

**----------------------> Implicit ---------------------------------->**

**byte short int long**

**<---------------------Explicit ---------------------<-----------------**

**class Demo {**

**public static void main(String args[]) {**

**byte a = 10; // range of byte value -128 to 127**

**short b = a;**

**System.out.println(a);**

**System.out.println(b);**

**short c = 129;**

**byte d = (byte)c;**

**System.out.println(c);**

**System.out.println(d);**

**}**

**}**

**class Demo {**

**public static void main(String args[]) {**

**int a=10;**

**float b =a;**

**System.out.println(a);**

**System.out.println(b);**

**float c = (float)10.10;**

**int d = (int)c;**

**System.out.println(c);**

**System.out.println(d);**

**}**

**}**

**Operator**

**if statements**

**switch statement**

**looping**

**while loop**

**do while loop**

**for loop**

**for each loop or enhanced loop**

**array : it is a user-defined data types which is use to store same type of values.**

**syntax**

**dataType arrayName[];**

**int abc[];**

**int []abc2**

**int [] abc3;**

**int[] abc4**

**[]int abc5; InValid**

**int []abc2={10,20,30,40,40,50}**

**for enhanced loop**

**for(dataType variableName:arrayName){**

**}**

**class Demo {**

**public static void main(String args[]) {**

**int []abc={10,20,30,40,50};**

**System.out.println(abc[0]);**

**System.out.println(abc[4]);**

**System.out.println("Using for loop");**

**for(int i=0;i<abc.length;i++) {**

**System.out.println(abc[i]);**

**}**

**System.out.println("For each loop");**

**for(int a:abc) {**

**System.out.println(a);**

**}**

**}**

**}**

**Creating memory for array in Java**

**dataType []arrayName = new dataType[size];**

**int []abc = new int[10]; 10\*4 = 40 byte memory**

**int abc[10];**

**class Demo {**

**public static void main(String args[]) {**

**int []abc={10,20,30,40,50};**

**System.out.println(abc[0]);**

**System.out.println(abc[4]);**

**System.out.println("Using for loop");**

**for(int i=0;i<abc.length;i++) {**

**System.out.println(abc[i]);**

**}**

**System.out.println("For each loop");**

**for(int a:abc) {**

**System.out.println(a);**

**}**

**}**

**}**

**Taking value through keyboards in Java.**

**1. Using Scanner class**

**2. Using DataInputStream class**

**3. BufferedReader class**

**4. Command Line Arguments.**

**Scanner : Scanner is a pre-defined class part of util package. Which contains set of methods which help to scan the value through keyboards.**

**package : It is collection of classes and interfaces.**

**syntax to create the Scanner class object.**

**Scanner obj = new Scanner(System.in);**

**byte : obj.nextByte()**

**short : obj.nextShort();**

**int : obj.nextInt();**

**long: obj.nextLong();**

**float : obj.nextFloat();**

**double : obj.nextDouble();**

**boolean : obj.nextBoolean();**

**String str = obj.next(); //only one word**

**String str = obj.nextLine(); more than one word.**

**Create the Scanner class object in main method, create array id,name,salary variables and create the memory size for those variable using scanner class object. then take id,name,salary for more than one employee and find hra, da and pf where HRA is 10%on salary, DA is 5%salary and PF is 7%salary. Then display all employees id,name,salary(grosssSalary).**

**import java.util.Scanner;**

**class Demo {**

**public static void main(String args[]){**

**Scanner obj = new Scanner(System.in);**

**System.out.println("How many record do you wan to store?");**

**int n = obj.nextInt();**

**int id[]=new int[n];**

**String name[]=new String[n];**

**float salary[]=new float[n];**

**System.out.println("Enter the records one by one");**

**//Receive the values**

**for(int i=0;i<n;i++) {**

**System.out.println("Enter the id");**

**id[i]=obj.nextInt();**

**obj.nextLine(); //use to hold the enter key**

**System.out.println("Enter the name");**

**name[i]=obj.nextLine();**

**System.out.println("Enter the salary");**

**salary[i]=obj.nextFloat();**

**}**

**//Business logic**

**for(int i=0;i<n;i++) {**

**float hra, da,pf;**

**hra = salary[i]\*0.10f;**

**da = salary[i]\*0.05f;**

**pf = salary[i]\*0.07f;**

**salary[i]=salary[i]+hra +da -pf;**

**}**

**//Display Records;**

**for(int i=0;i<n;i++) {**

**System.out.println("id is"+id[i]+" Name is"+name[i]+" Salary is"+salary[i]);**

**}**

**}**

**}**

**OOPs**

**object and class**

**object : any real world entity**

**class : blue print of object or template of object.**

**price, wheel, color etc**

**Car :**

**start(), appliedGear(), moving(), stop()**

**className objectRefeferenceName = new className();**

**Car santro =new Car();**

**Access the class non static methods**

**objectReferencename.methodName();**

**santro.start();**

**objectRefernceName.variableName = value;**

**santro.wheel=4;**

**Types of variables or fields**

**In Java variables are divided into 3 types**

**1. Instance variable**

**a. The variable which declare outside a method including main method is known as Instance variable.**

**b. All instance variable hold default values according to their data types.**

**like**

**int family 0**

**float family 0.0**

**char space (white)**

**boolean false**

**String null**

**c. We can access all instance variable inside same class method directly but the method must be non static.**

**2. Local variable**

**a. The variable which declare inside a method including main method is known as local variable.**

**b. The scope of the variable within that block where it declare.**

**c. The local variable doesn't hold default value we have to initialize.**

**class Car {**

**int wheel;**

**String color;**

**float price;**

**void start() {**

**int temp=0;**

**System.out.println(wheel);**

**System.out.println(color);**

**System.out.println(temp);**

**}**

**void stop() {**

**String msg="Welcome";**

**System.out.println(wheel);**

**System.out.println(color);**

**System.out.println(msg);**

**}**

**}**

**class CarTest {**

**public static void main(String args[]) {**

**Car santro = new Car();**

**santro.start(); santro.stop();**

**}**

**}**

**Create two classes Employee and EmployeeTest**

**Where Employee class must be contains three instance variables id,name,salary and Scanner class objects.**

**read(), calSalary() and display() non static methods.**

**EmployeeTest class must be contains main methods then create the Employee class object and call read() which help to receive the value through keyboards. calSalary() method which help to do calculation on salary.**

**hra, da and pf (local variables).**

**hra = 10%on Salary , da = 5%on salary and pf = 7%onsalary**

**display() method to display id,name,salaray(GrosssSalary)**

**First Employee object all three methods.**

**Second Employee object only two methods (read and display)**

**import java.util.Scanner;**

**class Employee {**

**int id;**

**String name;**

**float salary;**

**Scanner obj =new Scanner(System.in);**

**void read() {**

**System.out.println("Enter the id");**

**id = obj.nextInt();**

**System.out.println("Enter the name");**

**name= obj.next();**

**System.out.println("Enter the salary");**

**salary = obj.nextFloat();**

**}**

**void calSalary() {**

**float hra, da,pf;**

**hra = salary\*0.10f;**

**da = salary \*0.05f;**

**pf = salary \*0.07f;**

**salary = salary+hra+da-pf;**

**}**

**void display() {**

**System.out.println("Id is "+id+" name is "+name+" salary is "+salary);**

**}**

**}**

**class EmployeeTest {**

**public static void main(String args[]) {**

**Employee emp1 = new Employee();**

**emp1.read(); emp1.calSalary(); emp1.display();**

**Employee emp2 = new Employee();**

**emp2.read();**

**//emp2.calSalary();**

**emp2.display();**

**}**

**}**

**Constructor : it is a type of special method which help to create the object the objects.**

**pts**

**1. Constructor must be same name as the class itself.**

**2. Constructor doesn't contains return type not event void also.**

**3. Constructor no need to call it will call automatically when we create the object of that class.**

**class Employee {**

**Employee() {**

**System.out.println("Object created....");**

**}**

**void display() {**

**System.out.println("display method");**

**}**

**}**

**class EmployeeTest {**

**public static void main(String args[]) {**

**Employee emp1 = new Employee();**

**emp1.display();**

**emp1.Employee();**

**}**

**}**

**In the life of the object if we want to perform task only one time that type of code write inside constructor ex: initialization.**

**In the life of the object if we want to perform task more than one time that type of code write inside a methods.**

**class Employee {**

**int id;**

**Employee() {**

**System.out.println("Object created....");**

**}**

**void display() {**

**System.out.println("display method");**

**}**

**}**

**class EmployeeTest {**

**public static void main(String args[]) {**

**//int a;**

**Employee emp1;**

**new Employee();**

**new Employee().display();**

**new Employee().display();**

**Employee emp2 = new Employee();**

**emp2.display();**

**emp2.display();**

**emp2.display();**

**}**

**}**

**Encapsulation :Binding or wrapping data(fields/variables) and code(functions/methods) in a single unit is known as Encapsulation.**

**Ex :**

**class**

**class Employee {**

**private String name;**

**private float salary;**

**void setValue(String name, float salary) {**

**this.name =name;**

**if(salary<0) {**

**this.salary = 8000;**

**}else {**

**this.salary = salary;**

**}**

**}**

**void display() {**

**System.out.println("Name is "+name);**

**System.out.println("Salary is "+salary);**

**}**

**}**

**class EmployeeTest {**

**public static void main(String args[]) {**

**//display();**

**//name="Raj";**

**//salary = 12000;**

**Employee emp = new Employee();**

**//emp.name="Ravi";**

**//emp.salary = -12000;**

**emp.setValue("Ravi",-12000);**

**emp.display();**

**}**

**}**

**JavaBean class : JavaBean is a type of normal class with set of rules and regulation on property and behaviours.**

**Normal class Vs JavaBeanClass**

**class Employee {**

**private int id,age;**

**private String name;**

**//helper method**

**void setValue() {}**

**}**

**public class Employee {**

**private int id;**

**private String name;**

**private int age;**

**//setter and getter**

**//set is use to set the value with term and conditions.**

**//get is use to get the value**

**public void setId(int id) {**

**this.id = id;**

**}**

**public int getId() {**

**return id;**

**}**

**}**

**Pure Encapsulation class ie JavaBean**

**Inheritance : Inheritance is use to inherits or acquire the properties and behaviour of old class to new class.**

**class OldClass { super class, base class, parent class**

**properties**

**behaviour**

**}**

**class NewClass extends OldClass{**

**properties**

**behaviour //sub class, derived class, child class.**

**}**

**class A {**

**void dis1() {**

**System.out.println("A class method");**

**}**

**}**

**class B extends A{**

**void dis2() {**

**System.out.println("B class method");**

**}**

**}**

**class EmployeeTest {**

**public static void main(String args[]) {**

**A obj1 = new A();**

**B obj2 = new B();**

**obj1.dis1();**

**obj2.dis2();**

**obj2.dis1();**

**}**

**}**

**Types of Inheritance**

**1. Single Inheritance**

**one super class and one sub class**

**class A {}**

**class B extends A {}**

**2. Multilevel Inheritance**

**one super class and n number of sub classes connected one by one**

**class A {}**

**class B extends A{}**

**class C extends B{}**

**class D extends C{}**

**3. Hierarchical Inheritance**

**one super class and n number of sub classes connected directly to super class.**

**class A{}**

**class B extends A{}**

**class C extends A{}**

**4. Multiple Inheritance**

**more than one super class and one sub class**

**class A {}**

**class B {}**

**class C extends A,B {} Java doesn't support this type of inheritance. This type of inheritance java support indirectly using interface.**

**class Employee {**

**id,name,salary**

**Scanner obj**

**read()**

**calSalary()**

**display()**

**}**

**class Manager extends Employee{**

**numberOfEmp**

**readMgr()**

**disMgr();**

**}**

**class Programmer extends Employee{**

**projectName;**

**readPrg()**

**disPrg()**

**}**

**OOPs Relationship**

**1. Manager is a Employee**

**2. Employee/Manager has a Address**

**class Manager extends Employee{**

**}**

**class Employee {**

**Address add = new Address();**

**}**

**class Address {**

**}**

**has a relationship**

**1. Association**

**2.Aggregation**

**3. Composition**

**3. static variable**