***Dt : 23/11/2022***

***1.Thread Creation:***

***=>The process of creating thread using start() method is known as Thread***

***creation process or New Thread creation.***

***2.Ready-to-run:***

***=>The state of thread which is ready to execute by Thread-scheduler is known***

***as Ready-to-run.***

***3.Running:***

***=>The state in which the thread under execution is known a "Running State"***

***Note:***

***=>Thread Scheduler will shedule the threads from Ready-to-run state to running***

***state based on algorithms.***

***(a)Thread-Completion:***

***=>The state in which the thread executed successfully and generated result***

***is known as Thread-Completion state.***

***(b)Thread-Blocked-state:***

***=>The state in which the thread is temporarly blocked from execution is***

***known as Blocked state.***

***Note:***

***=>when we use wait() or sleep() methods then the thread in under blocked state.***

***=>wait() method will block the thread execution until it receives msg in the***

***form of notify() or notifyAll()***

***=>sleep() method will block the thread execution in sime timer***

***Thread-Dead-Lock :***

***=>The permanent blockage of thread is known as Thread-Dead-lock.***

***Note:***

***=>If any event raised under blocked state is permanent then the thread is under***

***deadlock.***

***==========================================================================***

***faq:***

***define LiveLock?***

***=>The temporary blockage of thread is known as LiveLock.***

***(Blocked state of thread is known as LiveLock)***

***faq:***

***define Daemon Thread?***

***=>The thread which executes contineously is known as Daemon Thread***

***(Server Service threads are daemon threads)***

***=========================================================================***

***Application of Threads:***

***(i)Threads are used in Server Application development***

***(ii)Threads are used in Server Development***

***(iii)Threads are used in Gaming Applications***

***=====================================================================***

***\*imp***

***define "java.lang.Object" class?***

***=>"java.lang.Object" class is the ParentClass or SuperClass of all the classes***

***declared in the application.***

***=>The following are some important methods of "Object" class:***

***1.hashCode()***

***2.toString()***

***3.clone()***

***4.equals()***

***5.wait()***

***6.notify()***

***7.notifyAll()***

***8.getClass()***

***9.finalize()***

***1.hashCode():***

***=>The unique numeric number which is generated while object creation process is***

***known as hashCode.***

***=>we use hashCode() method to display the hashCode of an object.***

***syntax:***

***int hc = obj.hashCode();***

***=>we display the hashCode to check the object is created or not.***

***2.toString():***

***=>toString() method is used to display the content from the object.***

***syntax:***

***String data = obj.toString();***

***=>toString() method is auto-executable method and which is executed automatically***

***when we display object\_reference***

***\*imp***

***3.clone():***

***=>The process of creating the duplicate copy of an object is known as cloning***

***process.***

***=>we use clone() method to perform Object-Cloning process.***

***syntax:***

***Object o = obj.clone();***

***Types of Cloning processes:***

***=>Cloning process is categorized into two types:***

***(a)Shallow Cloning process***

***(b)Deep Clonning process***

***(a)Shallow Cloning process:***

***=>In Shallow Cloning process only OuterObjects are cloned and referred objects***

***are not cloned.***

***(b)Deep Clonning process:***

***=>In Deep Cloning process both OuterObjects and reffered Objects are cloned.***

***---------------------------------------------------------------------***

***=>The following steps are used in Cloning process:***

***step-1 : The user-defined class must be implemented from "java.lang.Cloneable"***

***interface***

***step-2 : The user-defined class must be declared with one user-defined Object***

***return type method***

***step-3 : This user-defined Object return type method will call pre-defined clone()***

***method to perform cloning process***

***step-4 : we call user-defined object return type method to start the cloning***

***process***

***=======================================================================***

***Ex-program : Demonstrating "Shallow Cloning Process".***

***EmpContact.java***

***package test;***

***public class EmpContact extends Object{***

***public String mailId;***

***public long phoneNo;***

***@Override***

***public String toString() {***

***return "MailId:"+mailId+"\nPhoneNo:"+phoneNo;***

***}***

***}***

***Employee.java***

***package test;***

***public class Employee extends Object implements Cloneable{***

***public String empId,name,desg;***

***public EmpContact ec = new EmpContact();***

***@Override***

***public String toString() {***

***return "EmpId:"+empId+"\nEmpName:"+name+"\nEmpDesg:"+desg;***

***}***

***public Object startCloning() {***

***Object o = null;***

***try {***

***o = super.clone();***

***}catch(Exception e) {e.printStackTrace();}***

***return o;***

***}***

***}***

***DemoObject1.java(MainClass)***

***package maccess;***

***import test.\*;***

***import java.util.\*;***

***public class DemoObject1 {***

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

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

***//Original Object***

***Employee ob1 = new Employee();***

***System.out.println("Enter the empId:");***

***ob1.empId = s.nextLine();***

***System.out.println("Enter the empName:");***

***ob1.name=s.nextLine();***

***System.out.println("Enter the empDesg:");***

***ob1.desg=s.nextLine();***

***System.out.println("Enter the MailId:");***

***ob1.ec.mailId=s.nextLine();***

***System.out.println("Enter the PhoneNo:");***

***ob1.ec.phoneNo = s.nextLong();***

***System.out.println("\*\*\*\*\*\*\*\*Original Object\*\*\*\*\*\*\*\*\*\*");***

***System.out.println("=====Display data from Objects====");***

***System.out.println(ob1);***

***System.out.println(ob1.ec);***

***System.out.println("====hashCodes===");***

***System.out.println("hashCode of Employee Object : "+ob1.hashCode());***

***System.out.println("hashCode of EmpContact Object : "+ob1.ec.hashCode());***

***//Cloned Object or Duplicate Object***

***Employee ob2 = (Employee)ob1.startCloning();***

***System.out.println("\*\*\*\*\*\*\*\*Cloned Object\*\*\*\*\*\*\*\*\*\*");***

***System.out.println("=====Display data from Objects====");***

***System.out.println(ob2);***

***System.out.println(ob2.ec);***

***System.out.println("====hashCodes===");***

***System.out.println("hashCode of Employee Object : "+ob2.hashCode());***

***System.out.println("hashCode of EmpContact Object : "+ob2.ec.hashCode());***

***s.close();***

***}***

***}***

***o/p:***

***Enter the empId:***

***A121***

***Enter the empName:***

***Raj***

***Enter the empDesg:***

***SE***

***Enter the MailId:***

***raj@gmail.com***

***Enter the PhoneNo:***

***9898981234***

***\*\*\*\*\*\*\*\*Original Object\*\*\*\*\*\*\*\*\*\****

***=====Display data from Objects====***

***EmpId:A121***

***EmpName:Raj***

***EmpDesg:SE***

***MailId:raj@gmail.com***

***PhoneNo:9898981234***

***====hashCodes===***

***hashCode of Employee Object : 2074407503***

***hashCode of EmpContact Object : 999966131***

***\*\*\*\*\*\*\*\*Cloned Object\*\*\*\*\*\*\*\*\*\****

***=====Display data from Objects====***

***EmpId:A121***

***EmpName:Raj***

***EmpDesg:SE***

***MailId:raj@gmail.com***

***PhoneNo:9898981234***

***====hashCodes===***

***hashCode of Employee Object : 1989780873***

***hashCode of EmpContact Object : 999966131***

***==========================================================***