***Dt : 24/11/2022***

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

***EmpContact.java***

***package test;***

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

***public String mailId;***

***public long phoneNo;***

***@Override***

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

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

***}***

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

***Object o = null;***

***try {***

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

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

***return o;***

***}***

***}***

***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() {***

***Employee e = null;***

***try {***

***e = (Employee)super.clone();***

***e.ec = (EmpContact)e.ec.startCloning();***

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

***return e;***

***}***

***}***

***DemoObject2.java(MainClass)***

***package maccess;***

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

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

***public class DemoObject2 {***

***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:***

***A222***

***Enter the empName:***

***Ram***

***Enter the empDesg:***

***TE***

***Enter the MailId:***

***ram@gmail.com***

***Enter the PhoneNo:***

***7878787812***

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

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

***EmpId:A222***

***EmpName:Ram***

***EmpDesg:TE***

***MailId:ram@gmail.com***

***PhoneNo:7878787812***

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

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

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

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

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

***EmpId:A222***

***EmpName:Ram***

***EmpDesg:TE***

***MailId:ram@gmail.com***

***PhoneNo:7878787812***

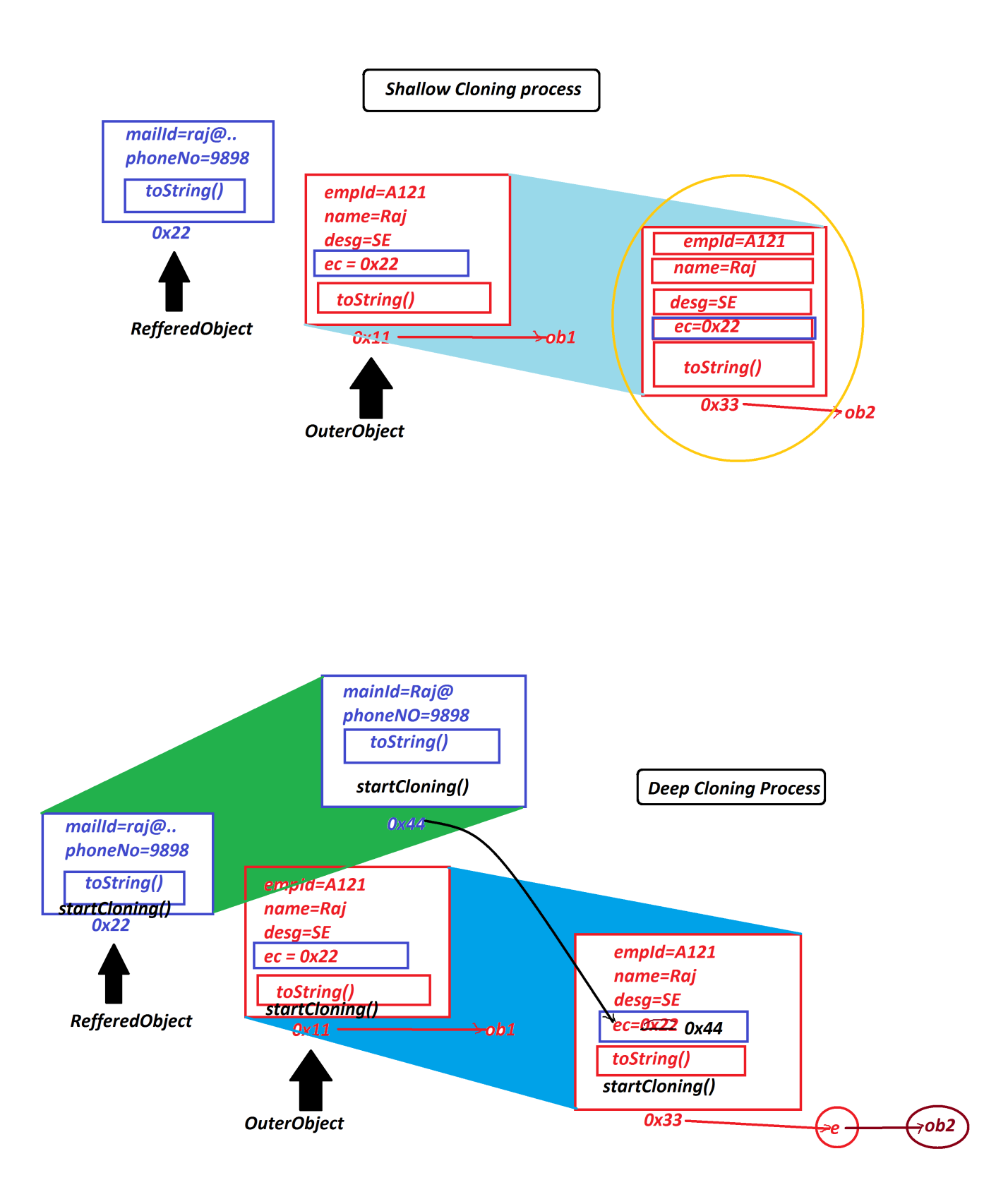
***====hashCodes===***

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

***hashCode of EmpContact Object : 1480010240***

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

***diagram:***

******

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

***Note:***

***=>In the process of performing Deep Cloning process the reffered classes also***

***must be implemented from "java.lang.Cloneable" interface and the classes must be***

***declared with User defined Object return\_type method.***

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

***define "Cloneable"?***

***=>"Cloneable" is an empty interface from java.lang package and specify the***

***Cloning process.***

***=>This "Cloneable" interface also known as "Marker Interface" or Tagging***

***Interface.***

***Note:***

***=>Cloning process cannot be perfomed without implementing from "Cloneable"***

***interface.***

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

***Advantage of Cloning process:***

***=>Part of protection and Security,Cloning process is used to take the backup of***

***an objects.***

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

***Note:***

***=>All Collection<E> and Map<K,V> objects are Serializable and Cloneable Objects,***

***except PriorityQueue<E>,which means PriorityQueue<E> object is Serializable but***

***Cloneable.***

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

***4.equals():***

***=>equals() method will compare two objects and generate boolean result.***

***5.wait()***

***6.notify()***

***7.notifyAll():***

***=>These three methods are used to establish communication b/w threads.***

***8.getClass():***

***=>getClass() method is used to display the class name of an object.***

***9.finalize():***

***=>finalize() method will check the object is eligible for garbage collection***

***process or not***

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

***faq:***

***define Garbage Collection Process?***

***=>The process of identifying anonymous objects and destroying is known as***

***Garbage Collection process.***

***=>The objects which are created without name are known as Anonymous Objects.***

***=>This garbage Collection Process is performed by ExecutionEngine using***

***predefined method "gc()".***

***=>This gc() method is part of ExecutionEngine and executes contineously like***

***Daemon thread.***

***Behaviour of gc() method:***

***=>gc() will identify all anonymous objects and call finalize() to check the***

***objects are eligible for Garbage Collection or not,then thay are destroyed.***

***Note:***

***=>This gc() method available from "Runtime" class and "System" class.***

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

***Ex:***

***Display.java***

***package test;***

***public class Display {***

***public void m2() {***

***System.out.println("=====m2()====");***

***new Test().m1();***

***}***

***}***

***Test.java***

***package test;***

***public class Test {***

***public void m1() {***

***System.out.println("====m1()====");***

***}***

***}***

***DemoObject3.java***

***package maccess;***

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

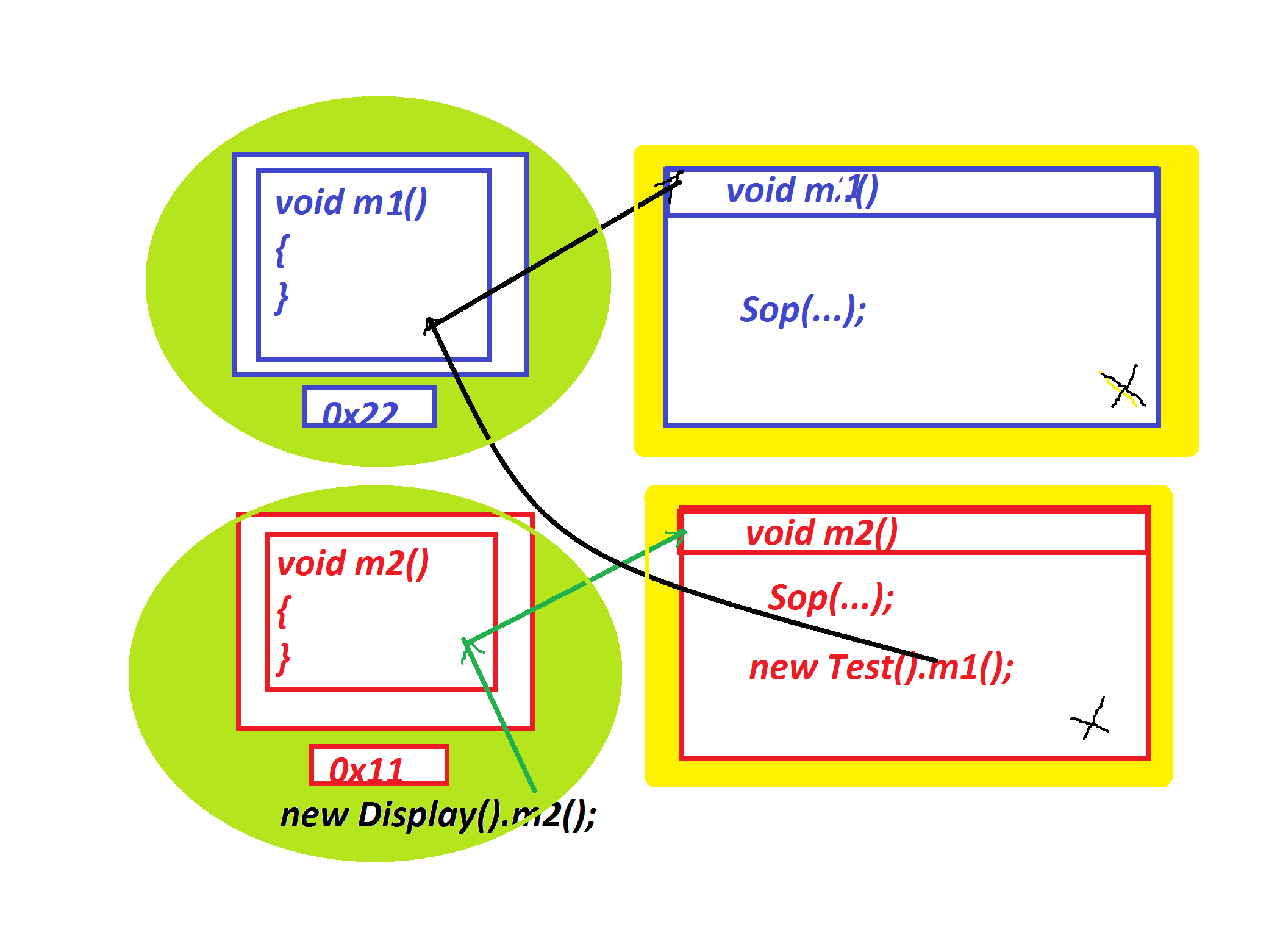
***public class DemoObject3 {***

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

***new Display().m2();***

***}***

***}***

******