NAME- ARMAN MANDAL

Roll No:- 00191050106 1

1.

a) In Java programs are compiled into byte code and the byte code is platform-independent.

The byte code is executed by the Java Virtual Machine and Java virtual machine is platform dependent.

So here the compiler creates a .class file that is readable for JVM (Java Virtual Machine)

b) To create deep copy we need to override the clone() method of object class.

d) X $ Y => means X is the main class, with in this class Y is another inner class.

e) There are four types of access specifier for a class private, public, protected and default

If we doesnot specify access specifier then it makes default access specifier.

for make a private member,

private member;

for protected protected member;

for public public member;

f)

(i) Abstract class doesn't support multiple inheritance but Interface supports multiple inheritance

(ii) Abstract class can have final, non-final, Static and non-static ~~method~~ variables. But Interface has only Static and final variable.

g)

1) It ensures that the ~~over~~ user want to override a methode present in Superclass or Interface, so signature of the methode can not be changed by mistake, until if we write wrong signature of the methode then compile time error will occure.

2) It improves the readability of the code

h) Checked exception are the subclass of the Exception class. These type of exception can be handled by the try-catch block otherwise the program will give a compilation error.

```
try(){
  .
  .
}
catch(){
  .
  .
}
finalized(){
  .
  .
}
```

1)

```
import java.io.File;

public class Demo{

    public static void main(String [] args){

      try{

      File f = new File("demo1.txt");

      f.createNewFile();

      System.out.println("Is directory?" + f.isDirectory());

      }

      catch(Exception e){

      }

}
```

Thats how we can check.

---

2)

a) start() method of thread class is implemented as when it is called a new thread is created and code inside run() method is executed in the new Thread.

We cannot create an object of Runnable as it is an interface. we can make a class implement Runnable and then make a Runnable reference refer to an object of that class and call run() with that reference. This will simply execute the override run() method as defined in the current thread.

b)

```
class Account {
    private String ac_no;
    private int balance;
    void Set_data() { } // for set the account no and balance
    void get_data() { } // for showing the data
    int ret_balance() { } // this function will return balance.
    void credit() { } // this function will credit money and update balance,
    void debit() { } // this will debit money and update balance
}
class AccountList implement Runnable {
    ArrayList<Account> a = new ArrayList<Account>();
    void add_acc() { } // this will add account to arraylist
    void show_acc() { } // this will search the account and call get_data();
    public void run() {
        Switch(String I) {
        case "Add account": add_acc();
        case "show account": show_acc()
        Synchronized (a) {
            case "credit": credit();
        }
```

synchronized (a) {

case "delit": delit();

}

}

}

3)

a) A Software system deals with basic objects of type like

We can organize the system as follows

Class Name: Employee, DEPT, product, Stock info, Sale info, purchase info, after that we have to write some interface like,

a) Employee management

b) Sales Management

c) purchase Management

The function for Employee management interface are

1) Set_record

2) Get_record

3) Display Record

The Function for Sales Management

1) Stock update

2) Search_quantity

The function for purchase Management

1) Stock update 2) Search_quantity

we have to implement all the function from individual class.

3)
b) ~~class Imap friend~~

```
class Information{
 String name;
 String date_of_birth;
 void getdata(){} - takes data from user
 void showdata(){} - show the data
}

boolean check_month();

 boolean get_month(int month){
  String month= new String()
  String month= date_of_birth.Substring(3,5);
  int j = Integer.parseInt(month);
  if(j==month){
  return True;
  }
  else{
  return False; // Code to find if month is
                   matched
  }
}

class List{
 Information[]= new Information[5];
 void getdata() - // get information of friend
 void putdata() // put the list of friend
                   in output screen
 void check_for_month(int m){
  for(int i=0; i<5; i++){
   if(Information[i].getmonth(m)==true){
    Information[i].showdata();
  }
 }
 // print the name of friend born in
    same month.
```