WEEK-4 JAVA ASSIGNMENT

Q. 1 A person wants to create a Bank account, so he goes to Bank and the Bank requires following details to create a Bank account. Details such as first Name, last Name, age and address (Note: A descent address should comprise of Name, Door No, Street, City and Postal Code).

Having given that the Bank would create an account with following details: Account Number and the Bank would also request the person to maintain a minimum balance of Rs.500, which he has to deposit for his account to get activated.

Solution:-

```
import java.util.Scanner
public class BankApp

public static void main String   args
   Scanner s = new Scanner System in
   Bank myBank = new Bank

   int user_choice = 2


   do
      //display menu to user
      //ask user for his choice and validate it (make sure it is between 1 and 6)
      System out println
      System out println "1) Open a new bank account"
      System out println "2) Deposit to a bank account"
      System out println "3) Withdraw to bank account"
      System out println "4) Print account balance"
      System out println "5) Quit"
      System out println
      System out print "Enter choice [1-5]: "
      user_choice = s nextInt
      switch user_choice
         case 1:
            System out println "Enter a customer name"
               String cn = s next
```

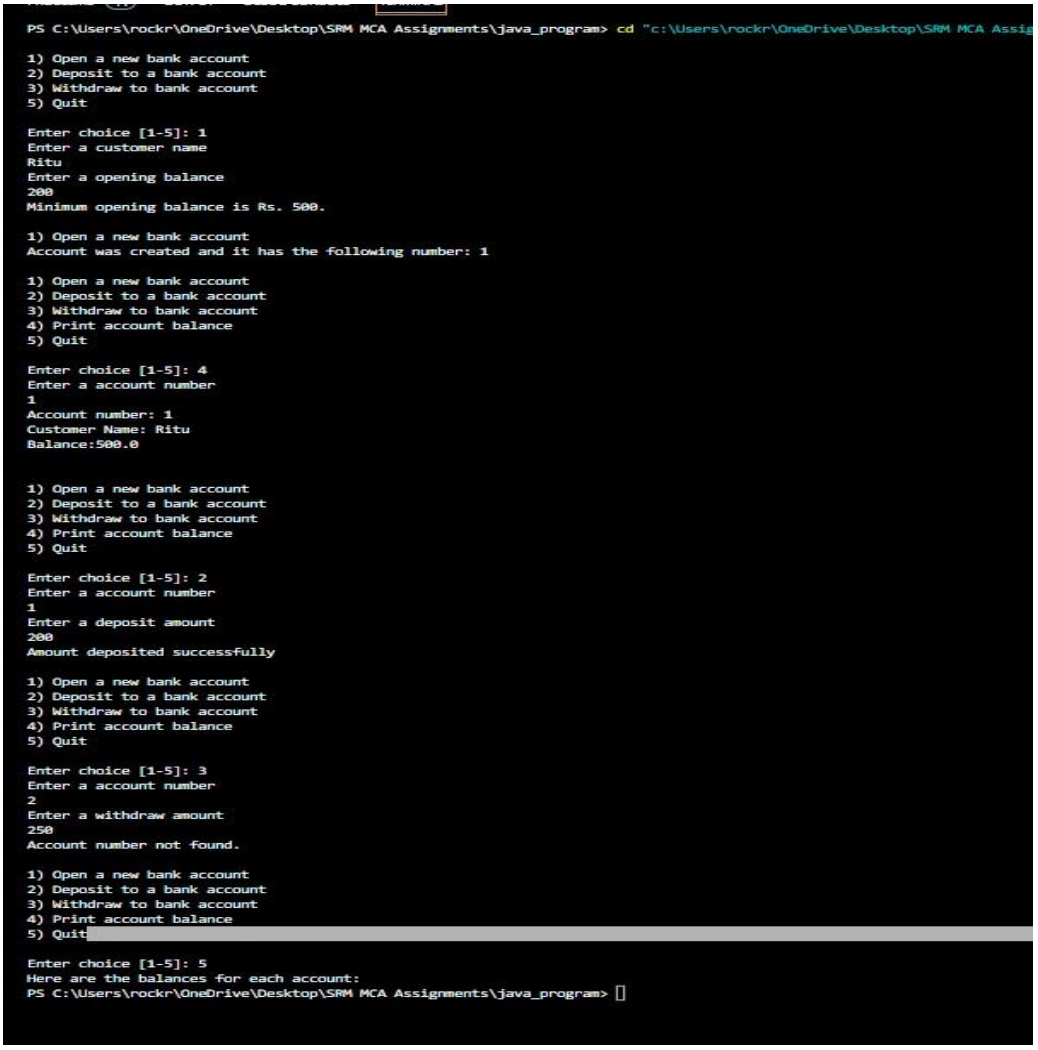
```
            System out println "Enter a opening balance"
            double d = s nextDouble
            if  d<500
               System out println  "Minimum opening balance is Rs. 500."

            else
            System out println  "Account was created and it has the following number: " +
myBank openNewAccount  cn  d

            break
        case 2: System out println "Enter a account number"
            int an = s nextInt
            System out println "Enter a deposit amount"
            double da = s nextDouble
            myBank depositTo an da
            break
        case 3: System out println "Enter a account number"
            int acn = s nextInt
            System out println "Enter a withdraw amount"
            double wa = s nextDouble
            myBank withdrawFrom  acn wa
            break
        case 4: System out println "Enter a account number"
            int anum = s nextInt
            myBank printAccountInfo  anum
            break
        case 5:
            System out println "Here are the balances " + "for each account:"
        case 6:
           System exit  0


while user_choice != '6'


static class Bank
private BankAccount    accounts    // all the bank accounts  at  this  bank
private int numOfAccounts = 5      // the number of bank accounts at this bank

// Constructor: A new Bank object initially doesn't contain any accounts.
public Bank
```

```
    accounts = new BankAccount 5
    numOfAccounts = 0


// Creates a new bank account using the customer name and the opening balance given as
parameters
// and returns the account number of this new account. It also adds this account into the
account list
// of the Bank calling object.
public int openNewAccount String customerName  double openingBalance

    BankAccount b = new BankAccount customerName openingBalance
    accounts numOfAccounts  = b
    numOfAccounts++
    return b getAccountNum


// Withdraws the given amount from the account whose account number is given. If the
account is
// not available at the bank, it should print a message.
public void withdrawFrom  int accountNum double amount
    for  int i =0  i<numOfAccounts  i++
        if accountNum == accounts i getAccountNum
            accounts i  withdraw amount
            System out println "Amount withdrawn successfully"
            return


    System out println  "Account number not found."


// Deposits the given amount to the account whose account number is given. If the account is
not
// available at the bank, it should print a message.
public void depositTo  int accountNum double amount
    for int i =0  i<numOfAccounts  i++
        if accountNum == accounts i getAccountNum
            accounts i  deposit amount
            System out println "Amount deposited successfully"
            return
```

```
    System out println  "Account number not found."


// Prints the account number, the customer name and the balance of the bank account whose
// account number is given. If the account is not available at the bank, it should print a
message.
public void printAccountInfo  int accountNum
    for  int i =0 i<numOfAccounts i++
            if accountNum == accounts i getAccountNum
                System out println  accounts  i getAccountInfo
                return


    System out println  "Account number not found."


public void printAccountInfo  int accountNum int n
    for  int i =0 i<numOfAccounts i++
                if accountNum == accounts i getAccountNum
                    System out println  accounts  i getAccountInfo
                    return


    System out println  "Account number not found."









  static class BankAccount

      private  int accountNum
      private String customerName
      private double balance
      private static int noOfAccounts=0

      public String getAccountInfo
```

```
        return "Account number: " + accountNum + "\nCustomer Name: " + customerName +
"\nBalance:" + balance +"\n"



    public BankAccount String abc double xyz
      customerName = abc
      balance = xyz
      noOfAccounts ++
      accountNum = noOfAccounts


  public int getAccountNum
     return accountNum

  public void deposit double amount

     if amount<=0
        System out println "Amount to be deposited should be positive"
       else
        balance = balance + amount



  public void withdraw double amount

     if amount<=0
         System out println "Amount to be withdrawn should be positive"

     else

        if  balance < amount
           System out println "Insufficient balance"
         else
           balance = balance - amount




//end of class
```

Output:-



Q.2 What is object and class in Java with example?

Solution:-

CLASS:-

1. Class is a set of object which shares common characteristics/ behavior and common properties/ attributes.
2. Class is not a real world entity. It is just a template or blueprint or prototype from which objects are created.
3. Class does not occupy memory.
4. Class is a group of variables of different data types and group of methods.

A class in java can contain:

- data member
- method
- constructor
- nested class and
- interface

Syntax to declare a class:

```
access_modifier  class<class_name>
{
   data member;
   method;
   constructor;
   nested class;
   interface;
}
```

Example of class:-

- Animal
- Student
- Bird
- Vehicle
- Company

Program class example:-

```
class Student
{
   int id;//data member (also instance variable)
   String name; //data member (also instance variable)

   public static void main(String args[])
   {
      Student s1=new Student();//creating an object of Student
      System.out.println(s1.id);
      System.out.println(s1.name);
    }
}
```

Output
0

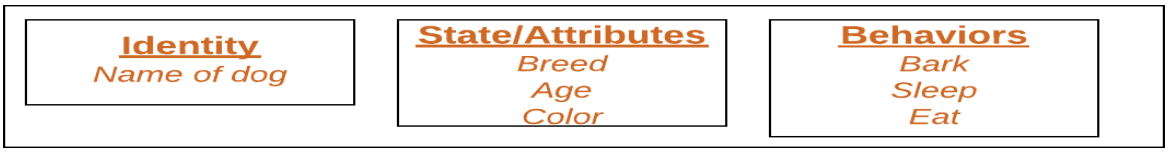
null

OBJECT:-

It is a basic unit of Object-Oriented Programming and represents real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

Example of an object: dog



Objects correspond to things found in the real world. For example, a graphics program may have objects such as “circle”, “square”, and “menu”. An online shopping system might have objects such as “shopping cart”, “customer”, and “product”.

```
// Class Declaration

public class Dog
{
   // Instance Variables
   String name;
   String breed;
   int  age;
   String color;

   // Constructor Declaration of Class
   public Dog(String name, String breed,
              int age, String color)
   {
      this.name = name;
      this.breed = breed;
```

```
    this.age = age;
    this.color = color;
}

// method 1
public String getName()
{
    return name;
}

// method 2
public String getBreed()
{
    return breed;
}

// method  3
public  int  getAge()
{
    return age;
}

// method 4
public String getColor()
{
    return color;
}

@Override
public String toString()
{
    return("Hi my name is "+ this.getName()+
          ".\nMy breed,age and color are " +
          this.getBreed()+"," + this.getAge()+
          ","+ this.getColor());
}

public static void main(String[] args)
{
    Dog tuffy = new Dog("tuffy","papillon", 5, "white");
    System.out.println(tuffy.toString());
}
```

```
}
```

Output:

```
Hi my name is tuffy.

My breed,age and color are papillon,5,white
```

Q3.What are classes and Objects?

Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

To create a class, use the keyword class:

Main.java

Create a class named "Main" with a variable x:

```
public class Main {

  int x = 5;

}
```

Create an Object

In Java, an object is created from a class. We have already created the class named Main, so now we can use this to create objects.

To create an object of Main, specify the class name, followed by the object name, and use the keyword new:

Example

Create an object called "myObj" and print the value of x:

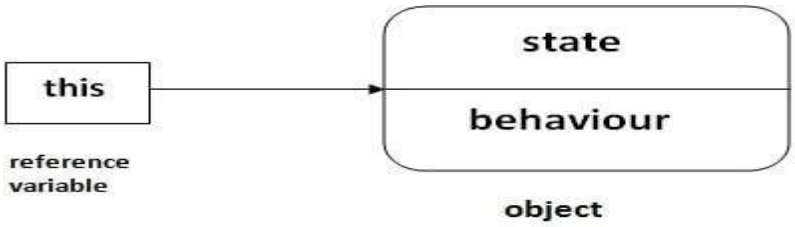
```
public class Main {

  int x = 5;
```

```
  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Q.4 What is the use of 'this' keyword?

Solution:-

There can be a lot of usage of **Java this keyword**. In Java, this is a **reference variable** that refers to the current object.



The this keyword refers to the current object in a method or constructor.

The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter). If you omit the keyword in the example above, the output would be "0" instead of "5".

Usage of Java this keyword

Here is given the 6 usage of java this keyword.

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

Q.5 What is the super() function?

Solution:-

Super Keyword in Java

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

1. super is used to refer immediate parent class instance variable.

We can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields.

```
class Animal{
String color="white";
}
class Dog extends Animal{
String color="black";
void   printColor(){
System.out.println(color);//prints color of Dog class
System.out.println(super.color);//prints color of Animal class
}
}
class TestSuper1{
public static void main(String args[]){
Dog d=new Dog();
d.printColor();
}}
```

Output

```
black
white
```

2. super can be used to invoke parent class method

The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
void bark(){System.out.println("barking...");}
void work(){
super.eat();
bark();
}
}
class TestSuper2{
public static void main(String args[]){
Dog d=new Dog();
d.work();
}}
```

Output:-

```
eating...
barking...
```

3. super is used to invoke parent class constructor.

The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

```
class  Animal{
Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
Dog(){
super();
System.out.println("dog is created");
}
```

```
}
class TestSuper3{
public static void main(String args[]){
Dog d=new Dog();
}}
```

Output:-

animal is created
dog is created

Q.6 Can a top-level class be private or protected?

Solution:-

No, we cannot declare a top-level class as **private or protected**. It can be either **public** or **default (**no modifier**).** If it does not have a modifier, it is supposed to have a default access.

Syntax

```
// A top level class
  public class TopLevelClassTest {
    // Class body
}
```

- If a **top-level class** is declared as **private** the compiler will complain that the **modifier private is not allowed here.** This means that a **top-level class cannot be a private**, the same can be applied to **protected** access specifier also.
- **Protected** means that the member can be accessed by any class in the **same package and by subclasses** even if they are in **another package**.
- The **top-level** classes can only have **public, abstract** and **final** modifiers, and it is also possible to not define any class modifiers at all. This is called **default/package** accessibility.
- We can declare the **inner classes** as **private** or **protected**, but it is not allowed in **outer classes**.
- More than one **top-level class** can be defined in a Java source file, but there can be at most one **public top-level class** declaration. The file name must match the name of the public class.

Declare the class as Protected

Example

```
protected class ProtectedClassTest {
  int i = 10;
```

```
  void show() {
    System.out.println("Declare top-level class as protected");
  }
}
public class Test {
  public static void main(String args[]) {
    ProtectedClassTest pc = new ProtectedClassTest();
    System.out.println(pc.i);
    pc.show();
    System.out.println("Main class declaration as public");
  }
}
```

Output:-

modifier protected not allowed here

Declare the class as Private

Example

```
private class PrivateClassTest {
  int x = 20;
  void show() {
    System.out.println("Declare top-level class as private");
  }
}
public class Test {
  public static void main(String args[]) {
    PrivateClassTest pc = new PrivateClassTest();
    System.out.println(pc.x);
    pc.show();
    System.out.println("Main class declaration as public");
  }
}
```

Output:-

modifier private not allowed here