9/01/24

Lab Program - 5

Q. Develop a Java Program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

a) Accept deposit from customer and update the balance.
b) Display the balance.
c) Compute and deposit interest
d) Permit withdrawal and update the balance.
Check for the minimum balance, impose penalty if necessary and update the balance.

⇒

```
import java.util.Scanner;
class Account {
    String customerName;
    String accountNumber;
    String accountType;
    double balance;
Account (String customerName, String accountNumber,
        String accountType, double balance)
        {
```

```
this.customerName = customerName;
this.accountNumber = accountNumber;
this.accountType = accountType;
sc = new Scanner(System.in);
}
void deposit(){
System.out.println("Enter the deposit amount:");
int dipo = sc.nextInt();
balance += dipo;
}
void withdrawal(){
System.out.println("Enter the withdrawal amount:");
int with = sc.nextInt();
if(with > balance){
System.out.println("Insufficient Balance");
}
else{
balance = with;}
}
void display(){
System.out.println("Customer name:" + customerName);
System.out.println("Account number:" + accountNumber);
System.out.println("Balance:" + balance);
}
void applyinterest(){}
}
package java_lab;
public class Cur_acct extends Account{
Cur_acct(String customerName, int accountNumber,
String accountType)
{super(customerName, accountNumber, accountType);}
```

```
void withdrawal(){
  System.out.println("Enter the withdrawal amount:");
  int with = sc.nextInt();
  if (balance <= 2000){
     double pen = balance/(0.06);
  System.out.println("Insufficient balance penalty to
     be paid:" +pen);
  balance += pen;
}
else{
     balance = with;
  }
}
}
```

```
package java_lab;
public class Sav_acct extends Account{
  Sav_acct(String CustomerName, Account Number,
        accountType)
    {
Super(CustomerName, AccountNumber, accountType);
  }
void applyinterest(){
  System.out.println("Enter the interest rate:");
  int rate = sc.nextInt();
  double interest = balance*(rate/100);
  balance += interest;
  System.out.println("Balance after interest:"+balance);
}
}
```

```
package java_lab;
```

```
import java.util.Scanner;
public class Bank{
public static void main(String args[]){
 Scanner sc= new Scanner(System.in);
System.out.println("Enter customer name:");
String customerName= sc.nextLine();
System.out.println("Enter account number:");
int accountNumber = sc.nextLine();
Account Ca = new
Cur_acct (customerName, account number, "Current account");
Account Sa = new
Sav_acct (customerName, accountNumber, "Saving account");
int choice;
while(true){
System.out.println("---- Menu ----");
System.out.println("1. Deposit \n2. Withdrawal \n
  3. Compute interest for saving account \n 4. Display
  account details \n5. Exit");
choice = sc.nextInt();
switch(choice){
case 1;
 System.out.println("Enter the type of account:
    \n1. SavingAccount \n2. Current account");
int acc = sc.nextInt();
if(acc==1){
Sa.deposit();
}
else{
  Ca.deposit();
}
break;
```

```
Case 2:
    System.out.println("Enter the type of account:\n
    1. Saving Account \n 2. Current Account");
    int acc1 = sc.nextInt();
    if (acc1 == 1){
    sa.withdrawal();
    }
    else {
    ca.withdrawal();
    }
    break;
Case-3:
    sa.applyinterest();
    break;
Case-4:
    System.out.println("Enter the type of account:\n
    1. Saving account \n 2. Current Account");
    int acc2 = sc.nextInt();
    if (acc2 == 1){
      sa.display();
    }
    else{
    ca.display();
    }
    break;
Case 5: Break;
default: System.out.println("Invalid Choice");
        break; }
if (choice == 5){
        break; } }
}}
```

Output:

Enter Customer name:
Shreya
Enter account number-
22
--Menu--
1. Deposit
2. Withdrawal
3. Compute interest for saving account.
4. Display account details
5. Exit
1
Enter the type of account:
1. Saving Account 2. Current Account.
1.
Enter deposite amount: 5000
2
Enter the type of account:
1. Saving account 2. Current Account 1.
Enter the withdrawal amount, 2000
4.
Enter the type of account: 1.
Customer name: Shreya
Account Number: 22
Type of account: Saving Account
Balance: 3000
2.
Enter the withdrawal amount: 6000
Insufficient balance. Penalty to be paid: 833.334

16/01/24

Lab Program 6

1. String Constructors -
Output - ABCDEF
CDE

2. String length -
The length of the string is 13

String literal -
Hello World

String Concatenate -
str1 : Java
str2 : Programming
str3 : Java programming

3. class string
{
public static void main (String arg[])
{
char c[] = {'J', 'A', 'V', 'A'};
String s1 = new String (c);
String s2 = new String (s1);
System.out.println (s1);
System.out.println (s2);
}}

Output -
JAVA
JAVA

13. Write a Java Program to create an abstract class Bird with abstract methods fly() and makeSound(). Create subclasses Eagle and Hawk that extend the Bird class and implement the respective methods to describe how each birds flies and makes a sound.

```
abstract class Bird{
    abstract void fly();
    abstract void makesound();
}
class Eagle extends Bird{
  void fly(){
      System.out.println("Eagle can fly very high.");
  }
void makeSound(){
   System.out.println("Eagle makes a screech sound");
  }
}
class Hawk extends Bird{
   void fly(){
      System.out.println("Hawk can fly moderately high.");
   }
  void makeSound(){
     System.out.println("Hawk makes a shrill sound.");
  }
}
public class Main{
   public static void main(String[] args){
      Bird bird1 = new Eagle();
      bird1.fly();
      bird1.makeSound();
```

```
        Bird bird2 = new Hawk();
        bird2.fly();
        bird2.makeSound();
    }
}
```

Output -
Eagle can fly very high.
Eagle makes a screech sound.
Hawk can fly moderately high.
Hawk makes a shrill sound.

8. Demonstrate startsWith() to give output true and false.

=>

```
public class Main
{
    Public static void main (String arg[]){
        String test = "teststrings";
        String pattern = "te";
    System.out.println(test.startsWith(pattern));
        pattern = "est";
        System.out.println("test.startsWith(pattern));
    }
}
```

Output —
True
False

Generics -

Write a Java program to create a generic class Stack which hold 5 integers and 5 doubles values.

Input -

```
import java.util.EmptyStackException;
public class stack <T> {
    private int maxSize;
    private int top;
    private Object [] stackArray;

    public Stack (int size){
    maxSize = size;
    stackArray = new Object [maxSize];
    top = -1;
    }
public void push (T value){
    if (top < maxSize -1) {
      top++;
      stackArray [top] = value;
    }
    else {
      throw new RuntimeException ("Stack is full");
    }
}
@SuppressWarnings("Unchecked")
public T pop(){
    if (!isEmpty ()) {
      return (T) stackArray [top--];
    } else {
      throw new EmptyStackException();
    }
}
```

```
public boolean isEmpty(){
    return (top == -1);
}
public int size(){
    return top+1;
}
public static void main (String args[]){
    Stack<Integer> intStack = new Stack<>();
    Stack<Double> doubleStack = new Stack<>();
    for (int i=0; i<5; i++){
        intStack.push(i);
        doubleStack.push((double) i);
    }
System.out.println("Integer Stack:");
for (int i=0; i<5; i++){
    System.out.println(intStack.pop());
}
System.out.println("Double Stack:");
for (int i=0; i<5; i++){
    System.out.println(doubleStack.pop());
}}}
```

Output -

Integer Stack:	Double Stack:
4	4.0
3	3.0
2	2.0
1	1.0
0	0.0

16/01/24