9/11/24

Lab 5 & Develop a Java program to create Bank that ..... savings acc & current acc.
Include necessary methods
(a) Accept deposit & update the balance.
(b) display
(c) Compute & deposite interest
(d) Permit withdrawal & update the balance.
(e) Check for min balance, impose penalty if needed & updt

```
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;
        this.balance = balance;
    }

    void deposit (double amount)
    {
        balance += amount;
        System.out.println("Deposit successful. Updated
                balance: " + balance);
    }
```

```
void displayBalance()
{
    System.out.println("Account Type: " + AccountType);
    System.out.println("Custom Name:" + customName);
    System.out.println("Account Number:" + accountNumber);
    System.out.println("Balance: " + Balance);
}
}

class SavAct extends Account
{
    SavAct(String customName, String accountNumber,
                                    double balance)
    {
        super(customerName, accountNumber, "Savings", balance);
    }
    void computeInterest(double interestRate)
    {
        double interest = balance * interestRate;
        deposit(interest);
        System.out.println("Interest computed &
            deposited. Updated Balance: " + balance);
    }
    void withdraw(double amount)
    {
        if (amount <= balance)
            balance -= amount;
            System.out.println("withdrawn successfully.
                updated balance:" + balance);
        else
            System.out.println("Insufficient funds.
                Withdrawal failed");
    }
}
```

__/__/____

```
class CurAcct extends Account
{
    double ServiceCharge;
    CurAcct(String customerName, String accountNumber, double balance,
                                                    double serviceCharge)
    {
        super(customerName, accountNumber, "current", balance);
        this.serviceCharge = serviceCharge;
    }
    void withdraw(double amount)
    {
        if (amount <= balance)
            balance -= amount;
            System.out.println("withdrawal successful. Updated
                balance: " + balance);
        else
            System.out.println("Insufficient funds, Withdrawal fail");
    }
}
public class Main
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter Customer Name: ");
        String customerName = s.nextLine();
        System.out.println("Enter account number: ");
        String accountNumber = s.nextLine();
        System.out.println("Enter initial balance: ");
        double initialBalance = s.nextDouble();

        SavAcct savingsAccount = new SavAcct(customerName,
                                    accountNumber, initialBalance);
        CurAcct currentAcct = new CurAcct(customerName,
                                    accountNumber, initialBalance
                                    ,50);

        int choice;
        double amount;
        double interestRate = 0.05;
```

__/__/____

```
do {
    Sys.o.pln("\nMenu:"\n 1. Deposit \n 2. Display Balance
        \n 3. Compute Interest (Savings Acc) \n
        4. Withdraw \n 5. Exit \n 6. Your Choice:");
    choice = s.nextInt();
switch(choice){
    case 1:
        Sys.o.p("Enter the amount to deposit:");
        amount = s.nextDouble();
        Sys.o.p("Choose account type (1. Savings / 2. Current):");
        int accountTypeChoice = s.nextInt();
        switch (accountTypeChoice)
        {       case 1:
                    savingssAccount.deposit(amount);
                    break;
                case 2:
                    currentAccount.deposit(amount);
                    break;
                default: Sys.o.p("Invalid Input");
        }
        break;

    case 2:
        Sys.o.p("Choose account type (1. Savings (2. Current):");
        int displayChoice = s.nextInt();
        switch (displayChoice)
        {
            case 1: savingsAccount.displayBalance();
                    break;
            case 2: currentAccount.displayBalance();
                    break;
            default:
                    Sys.o.pln("Invalid Input");
        }
        break;

    case 3:
        Sys.out.println("Enter the interestRate:");
        interestRate = s.nextDouble();
        SavingsAccount.computeInterest(interestRate);
        break;
```

__/__/____

```
case 4:
    sy.out.p("enter the amount to withdraw: ");
    amount = s.nextDouble();
    s.o.p("Choose account type (1.savings) 2.Current)
    int withdrawChoice = s.nextInt();
    switch (withdraw)
    {
        case 1:
            SavingsAccount.withdraw(amount)
            break;
        case 2:
            CurrentAccount.withdraw(amount)
            break;
        default:
            S.O.P("Invalid Input");
    }
    break;

case 5:
    system.out.println("Exiting the program....")
    break.
default:
    system.out.println("Invalid choice. PleaseTryag
}
}while(choice != 5);
s.close();
}
}
```

O/P →

__/__/____

O/P

Enter customer name: Shreya.
Enter account number: 12345789
Enter initial balance: 155879

Menu:
1. Deposit
2. display Balance
3. Compute Interest (Savings Account)
4. withdraw
5. Exit
Enter your choice: 1
Enter the amount to deposit: 11000
Choose account Type (1. Savings / 2. Current): 2
deposit successful. Updated balance: 166879

Menu:
1. deposit
2. display Balance
3. Compute Interest (Savings Account)
4. withdraw
5. Exit
Enter your choice: 3
Enter the interest rate: 0.2
deposit successful. Updated balance: 187054.8
Interest computed and deposited. Upload balance: 187054.8

10/01/24

16/1/24

1. Demonstrate various string constructor with proper java program

```
class String {
    public static void main (String args[]) {
        char c[] = {'J', 'a', 'v', 'a'};
        String s1 = new String(c);
        String s2 = new String(s1);
        System.out.println(s1);
        System.out.println(s2);
    }}
```

O/P: Java
Java

8. Demonstrate startswith() to give output true or false.

```
public class Main {
    public static void main (String args[]) {
        String test = "testString";
        String pattern = "te";
        System.out.println(test.startswith(pattern));
        pattern = "est";
        System.out.println(test.startswith(pattern));
    }
}
```

O/P:
True
False

19. Write a Java program to create an abstract class Bird with abstract method fly() and makeSound(). Create subclass Eagle and Hawk that extend the Bird class and implement the respective methods to describe how each bird 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 moderatly 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();
    }
}
```

O/P:

Eagle can fly very high
Eagle makes a screech sound
Hawk can fly moderatlly high
Hawk makes a shrill sound

Write a Java program to create a generic class Stack which holds 5 integers and 5 double values.

```
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<X(5);
   Stack<Double> doubleStack = new Stack<>(5);

  for(int i=0 ; i<5; i++) {
      intStack.push(i);
      doubleStack.push(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());}
}}
```

__/__/____

O/P:

Integer Stack:

4

3

2

1

0

Double Stack:

4.0

3.0

2.0

1.0

0.0

17/01/24