**Day 2**

**Third\_highest:**

import java.util.\*;

public class third\_highest {

public static void main(String[] arg){

Scanner s = new Scanner(System.in);

int n;

System.out.println("Enter size of array ");

n=s.nextInt();

System.out.println("Enter values of array ");

int [] a= new int[n];

for(int i=0;i<n;i++)

a[i] = s.nextInt();

int h1=a[0];

for(int i=1;i<n;i++)

{

if(a[i]>h1)

h1=a[i];

}

int h2=Integer.MIN\_VALUE;

for(int i=0;i<n;i++)

{

if(a[i]==h1)

continue;

else

{

if(a[i]>h2)

h2=a[i];

}

}

if(h2==Integer.MIN\_VALUE)

h2=h1;

int h3=Integer.MIN\_VALUE;

for(int i=0;i<n;i++)

{

if(a[i]==h1 || a[i]==h2)

continue;

else

{

if(a[i]>h3)

h3=a[i];

}

}

if(h3==Integer.MIN\_VALUE)

h3=h2;

System.out.println("Third highest element of array is "+h3);

}

**Pant :**

import java.util.\*;

public class pant {

public static void main(String[] arg){

Scanner s = new Scanner(System.in);

int n;

System.out.println("Enter size of pant ");

n=s.nextInt();

if(n%2==0)

{

for(int i=n-1;i>=0;i--)

{

for(int j=0;j<2\*n;j++)

{

if(j<=i)

System.out.print("\*");

else

{

if(j+i>=2\*n-1)

System.out.print("\*");

else

System.out.print(" ");

}

}

System.out.println();

}

}

else

{

for(int i=n-1;i>=0;i--)

{

for(int j=0;j<2\*n-1;j++)

{

if(j<=i)

System.out.print("\*");

else

{

if(j+i>=2\*n-2)

System.out.print("\*");

else

System.out.print(" ");

}

}

System.out.println();

}

}

}

}

**Inverted\_pant:**

import java.util.\*;

public class inverted\_pant {

public static void main(String[] arg){

Scanner s = new Scanner(System.in);

int n;

System.out.println("Enter size of pant ");

n=s.nextInt();

if(n%2==0)

{

for(int i=0;i<n;i++)

{

for(int j=0;j<2\*n;j++)

{

if(j<=i)

System.out.print("\*");

else

{

if(j+i>=2\*n-1)

System.out.print("\*");

else

System.out.print(" ");

}

}

System.out.println();

}

}

else

{

for(int i=0;i<n;i++)

{

for(int j=0;j<2\*n-1;j++)

{

if(j<=i)

System.out.print("\*");

else

{

if(j+i>=2\*n-2)

System.out.print("\*");

else

System.out.print(" ");

}

}

System.out.println();

}

}

}

}

**Diamond:**

import java.util.Scanner;

public class diamond {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of rows: ");

int rows = scanner.nextInt();

scanner.close();

int midRow = (rows + 1) / 2;

// Upper half of the diamond

for (int i = 1; i <= midRow; i++) {

for (int j = 1; j <= midRow - i; j++) {

System.out.print(" ");

}

for (int j = 1; j <= i; j++) {

System.out.print("\* ");

}

System.out.println();

}

// Lower half of the diamond

for (int i = midRow - 1; i >= 1; i--) {

for (int j = 1; j <= midRow - i; j++) {

System.out.print(" ");

}

for (int j = 1; j <= i; j++) {

System.out.print("\* ");

}

System.out.println();

}

}

}

**Exception handling:**

import java.util.Scanner;

public class task {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter the value of a: ");

int a = scanner.nextInt();

System.out.print("Enter the value of b: ");

int b = scanner.nextInt();

int result = a / b;

System.out.println("Result: " + result);

}

catch (ArithmeticException e) {

System.out.println("Division is not allowed");

System.out.println(e);

}

catch (Exception e) {

System.out.println(e.getClass());

}

finally {

scanner.close();

}

}

}

**String\_descending\_order:**

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

public class task2 {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("enter size of string");

int n = s.nextInt();

String[] str = new String[n];

for (int i=0;i<n;i++)

str[i] = s.next();

// for (int i=0;i<n;i++)

// System.out.println(str[i]);

Arrays.sort(str, new Comparator<String>() {

@Override

public int compare(String s1, String s2) {

double d1 = Double.parseDouble(s1);

double d2 = Double.parseDouble(s2);

return Double.compare(d2, d1);

}

});

for (String st : str) {

System.out.println("output:")

System.out.println(st);

}

}

}

**Divisor\_addition:**

import java.util.Scanner;

// Define the interface

interface DivisorAddition {

int divisorSum(int n);

}

// Implement the interface in a class

class DivisorAdditionImpl implements DivisorAddition {

@Override

public int divisorSum(int n) {

int sum = 0;

for (int i = 1; i <= n; i++) {

if (n % i == 0) {

sum += i;

}

}

return sum;

}

}

// Example usage

public class task3 {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("enter number:");

int number = s.nextInt();

DivisorAdditionImpl divisorAddition = new DivisorAdditionImpl();

int sum = divisorAddition.divisorSum(number);

System.out.println("Sum of divisors of " + number + ": " + sum);

}

}

**Class:**

import java.util.Scanner;

class FirstClass {

public int sumOfIntegers(int num1, int num2) {

return num1 + num2;

}

}

class SecondClass extends FirstClass {

}

class ThirdClass {

public void printSum() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first number: ");

int num1 = scanner.nextInt();

System.out.print("Enter the second number: ");

int num2 = scanner.nextInt();

scanner.close();

SecondClass secondObj = new SecondClass();

int result = secondObj.sumOfIntegers(num1, num2);

System.out.println("The sum of the integers is: " + result);

}

}

public class class1 {

public static void main(String[] args) {

ThirdClass thirdObj = new ThirdClass();

thirdObj.printSum();

}

}

**Bank:**

class SBIBankAccount {

protected double balance;

public void deposit(double amount) {

balance += amount;

System.out.println("Deposit successful. Current balance: " + balance);

}

public void withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

System.out.println("Withdrawal successful. Current balance: " + balance);

} else {

System.out.println("Insufficient funds. Withdrawal canceled.");

}

}

public void openAccount() {

System.out.println("Account opened successfully.");

}

}

class PremiumAccount extends SBIBankAccount {

public void deposit(double amount) {

if (amount >= 5000) {

balance += amount;

System.out.println("Deposit successful. Current balance: " + balance);

} else {

System.out.println("Premium account deposit should be at least 5000.");

}

}

public void withdraw(double amount) {

if (balance - amount >= 5000) {

balance -= amount;

System.out.println("Withdrawal successful. Current balance: " + balance);

} else {

System.out.println("Minimum balance of 5000 should be maintained in a premium account.");

}

}

public void openAccount() {

System.out.println("Premium account opened successfully.");

}

}

class JhandhanAccount extends SBIBankAccount {

public void deposit(double amount) {

balance += amount;

System.out.println("Deposit successful. Current balance: " + balance);

}

public void withdraw(double amount) {

if (balance - amount >= 0) {

balance -= amount;

System.out.println("Withdrawal successful. Current balance: " + balance);

} else {

System.out.println("Jhandhan account balance cannot go below 0.");

}

}

public void openAccount() {

System.out.println("Jhandhan account opened successfully.");

}

}

public class bank {

public static void main(String[] args) {

// Test the classes

PremiumAccount premiumAccount = new PremiumAccount();

premiumAccount.openAccount();

premiumAccount.deposit(5000);

premiumAccount.withdraw(2000);

System.out.println();

JhandhanAccount jhandhanAccount = new JhandhanAccount();

jhandhanAccount.openAccount();

jhandhanAccount.deposit(3000);

jhandhanAccount.withdraw(4000);

}

}

**Banking\_system:**

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

// Account interface

interface Account {

void deposit(double amount);

void withdraw(double amount);

double calculateInterest();

double getBalance();

}

// SavingsAccount class implementing Account interface

class SavingsAccount implements Account {

private double balance;

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

} else {

System.out.println("Insufficient balance");

}

}

public double calculateInterest() {

// Calculation logic for interest specific to SavingsAccount

return balance \* 0.05; // Assuming 5% interest rate

}

public double getBalance() {

return balance;

}

// Unique method for SavingsAccount

public void applyBonus() {

balance += 100; // Applying a bonus of 100 units

}

}

// CurrentAccount class implementing Account interface

class CurrentAccount implements Account {

private double balance;

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

} else {

System.out.println("Insufficient balance");

}

}

public double calculateInterest() {

// Calculation logic for interest specific to CurrentAccount

return balance \* 0.02; // Assuming 2% interest rate

}

public double getBalance() {

return balance;

}

// Unique method for CurrentAccount

public void issueCheque(double amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Cheque issued for amount: " + amount);

} else {

System.out.println("Insufficient balance");

}

}

}

// Bank class

class Bank {

private List<Account> accounts;

public Bank() {

accounts = new ArrayList<>();

}

public void addAccount(Account account) {

accounts.add(account);

}

public void depositToAccount(int accountIndex, double amount) {

Account account = accounts.get(accountIndex);

account.deposit(amount);

}

public void withdrawFromAccount(int accountIndex, double amount) {

Account account = accounts.get(accountIndex);

account.withdraw(amount);

}

public double calculateInterestForAccount(int accountIndex) {

Account account = accounts.get(accountIndex);

return account.calculateInterest();

}

public double getAccountBalance(int accountIndex) {

Account account = accounts.get(accountIndex);

return account.getBalance();

}

}

// Main class

public class BankingSystem {

public static void main(String[] args) {

Bank bank = new Bank();

Scanner scanner = new Scanner(System.in);

// Creating SavingsAccount

System.out.print("Enter initial balance for Savings Account: ");

double savingsInitialBalance = scanner.nextDouble();

Account savingsAccount = new SavingsAccount();

bank.addAccount(savingsAccount);

bank.depositToAccount(0, savingsInitialBalance);

// Creating CurrentAccount

System.out.print("Enter initial balance for Current Account: ");

double currentInitialBalance = scanner.nextDouble();

Account currentAccount = new CurrentAccount();

bank.addAccount(currentAccount);

bank.depositToAccount(1, currentInitialBalance);

// Perform operations

System.out.print("Enter the amount to deposit to Savings Account: ");

double savingsDepositAmount = scanner.nextDouble();

bank.depositToAccount(0, savingsDepositAmount);

System.out.print("Enter the amount to withdraw from Savings Account: ");

double savingsWithdrawAmount = scanner.nextDouble();

bank.withdrawFromAccount(0, savingsWithdrawAmount);

System.out.print("Enter the amount to deposit to Current Account: ");

double currentDepositAmount = scanner.nextDouble();

bank.depositToAccount(1, currentDepositAmount);

System.out.print("Enter the amount to withdraw from Current Account: ");

double currentWithdrawAmount = scanner.nextDouble();

bank.withdrawFromAccount(1, currentWithdrawAmount);

// Display updated balances

double savingsAccountBalance = bank.getAccountBalance(0);

double currentAccountBalance = bank.getAccountBalance(1);

System.out.println("Savings Account Balance: " + savingsAccountBalance);

System.out.println("Current Account Balance: " + currentAccountBalance);

// Calculate and display interest

double savingsAccountInterest = bank.calculateInterestForAccount(0);

double currentAccountInterest = bank.calculateInterestForAccount(1);

System.out.println("Savings Account Interest: " + savingsAccountInterest);

System.out.println("Current Account Interest: " + currentAccountInterest);

// Accessing unique methods of SavingsAccount and CurrentAccount

if (savingsAccount instanceof SavingsAccount) {

SavingsAccount sa = (SavingsAccount) savingsAccount;

sa.applyBonus();

}

if (currentAccount instanceof CurrentAccount) {

CurrentAccount ca = (CurrentAccount) currentAccount;

ca.issueCheque(100);

}

// Display updated balances after applying unique methods

savingsAccountBalance = bank.getAccountBalance(0);

currentAccountBalance = bank.getAccountBalance(1);

System.out.println("Updated Savings Account Balance: " + savingsAccountBalance);

System.out.println("Updated Current Account Balance: " + currentAccountBalance);

scanner.close();

}

}