WEEK-05

### Question 1

### Create a class Mobile with constructor and a method basicMobile0.

### Create a subclass CameraMobile which extends Mobile class , with constructor and a method newFeature0.

### Create a subclass AndroidMobile which extends CameraMobile, with constructor and a method androidMobile0.

### display the details of the Android Mobile class by creating the instance. .

### class Mobile{

### class CameraMobile extends Mobile {

### class AndroidMobile extends CameraMobile {

### expected output:

### Basic Mobile is Manufactured

### Camera Mobile is Manufactured

### Android Mobile is Manufactured

### Camera Mobile with 5MG px

### Touch Screen Mobile is Manufactured

### For example:

### 

### Program:

### class Mobile{

### public Mobile(){

### System.out.println("Basic Mobile is Manufactured");

### }

### void basicMobile(){

### 

### }

### }

### class CameraMobile extends Mobile{

### public CameraMobile(){

### System.out.println("Camera Mobile is Manufactured");

### }

### void newFeature(){

### System.out.println("Camera Mobile with 5MG px");

### }

### }

### class AndroidMobile extends CameraMobile{

### public AndroidMobile(){

### 

### System.out.println("Android Mobile is Manufactured");

### }

### void androidMobile(){

### System.out.println("Touch Screen Mobile is Manufactured");

### }

### }

### class prog{

### public static void main(String[] args){

### AndroidMobile mobile=new AndroidMobile();

### mobile.newFeature();

### mobile.androidMobile();

### }

### }

### 

### Question 2

### Create a class known as "BankAccount" with methods called deposit() and withdraw().

### Create a subclass called SavingsAccount that overrides the withdraw0 method to prevent withdrawals if the account

### balance falls below one hundred.

### For example:

### 

### Program:

### class BankAccount {

### // Private field to store the account number

### private String accountNumber;

### // Private field to store the balance

### private double balance;

### // Constructor to initialize account number and balance

### BankAccount(String accountNumber, double balance){

### this.accountNumber=accountNumber;

### this.balance=balance;

### }

### 

### 

### 

### 

### // Method to deposit an amount into the account

### public void deposit(double amount) {

### // Increase the balance by the deposit amount

### this.balance+=amount;

### 

### }

### // Method to withdraw an amount from the account

### public void withdraw(double amount) {

### // Check if the balance is sufficient for the withdrawal

### 

### if (balance >= amount) {

### // Decrease the balance by the withdrawal amount

### balance -= amount;

### 

### } else {

### // Print a message if the balance is insufficient

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

### }

### }

### // Method to get the current balance

### public double getBalance() {

### // Return the current balance

### return this.balance;

### }

### }

### class SavingsAccount extends BankAccount {

### // Constructor to initialize account number and balance

### public SavingsAccount(String accountNumber, double balance) {

### // Call the parent class constructor

### super(accountNumber,balance);

### 

### }

### // Override the withdraw method from the parent class

### @Override

### public void withdraw(double amount) {

### // Check if the withdrawal would cause the balance to drop below $100

### if (getBalance() - amount < 100) {

### // Print a message if the minimum balance requirement is not met

### System.out.println("Minimum balance of $100 required!");

### } else {

### // Call the parent class withdraw method

### super.withdraw(amount);

### }

### }

### }

### class prog {

### 

### public static void main(String[] args) {

### // Print message to indicate creation of a BankAccount object

### System.out.println("Create a Bank Account object (A/c No. BA1234) with initial balance of $500:");

### // Create a BankAccount object (A/c No. "BA1234") with initial balance of $500

### BankAccount BA1234 = new BankAccount("BA1234", 500);

### // Print message to indicate deposit action

### System.out.println("Deposit $1000 into account BA1234:");

### // Deposit $1000 into account BA1234

### BA1234.deposit(1000);

### // Print the new balance after deposit

### 

### System.out.println("New balance after depositing $1000: $"+BA1234.getBalance());

### 

### // Print message to indicate withdrawal action

### System.out.println("Withdraw $600 from account BA1234:");

### // Withdraw $600 from account BA1234

### BA1234.withdraw(600);

### // Print the new balance after withdrawal

### System.out.println("New balance after withdrawing $600: $" + BA1234.getBalance());

### // Print message to indicate creation of another SavingsAccount object

### System.out.println("Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300:");

### // Create a SavingsAccount object (A/c No. "SA1000") with initial balance of $300

### SavingsAccount SA1000 = new SavingsAccount("SA1000", 300);

### // Print message to indicate withdrawal action

### System.out.println("Try to withdraw $250 from SA1000!");

### // Withdraw $250 from SA1000 (balance falls below $100)

### SA1000.withdraw(250);

### // Print the balance after attempting to withdraw $250

### System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance());

### }

### }

### 

### Question 3

### create a class called College with attribute String name, constructor to initialize the name attribute , a method called

### Admitted(). Create a subclass called CSE that extends Student class, with department attribute , Course() method to sub

### class. Print the details of the Student.

### College:

### String collegeName;

### public College() { }

### public admitted() { }

### Student:

### String studentName;

### String department;

### public Student(String collegeName, String studentName,String depart) { }

### public toString0

### Expected Output:

### A student admitted in REC

### CollegeName : REC

### StudentName : Venkatesh

### Department : CSE

### For example:

### 

### Program:

### class College

### {

### protected String collegeName;

### public College(String collegeName) {

### // initialize the instance variables

### this.collegeName=collegeName;

### }

### public void admitted() {

### System.out.println("A student admitted in "+collegeName);

### }

### }

### class Student extends College{

### String studentName;

### String department;

### public Student(String collegeName, String studentName,String depart) {

### super(collegeName);

### // initialize the instance variables

### this.studentName=studentName;

### this.department=depart;

### 

### 

### }

### public String toString(){

### // return the details of the student

### return "CollegeName : "+collegeName+"\nStudentName : "+studentName + "\nDepartment : "+department;

### }

### }

### class prog {

### public static void main (String[] args) {

### Student s1 = new Student("REC","Venkatesh","CSE");

### s1.admitted();

### // invoke the admitted() method

### System.out.println(s1.toString());

### }

### }

### 

### Program:

import java.util.LinkedHashSet;

import java.util.Scanner;

import java.util.Set;

public class StringProcessor {

public static String processStrings(String input1, String input2) {

// Concatenate both strings

String combined = input1 + input2;

// Remove spaces and create a set to remove duplicates

Set<Character> charSet = new LinkedHashSet<>();

for (char c : combined.toCharArray()) {

if (c != ' ') {

charSet.add(c);

}

}

// If the set is empty, return "null"

if (charSet.isEmpty()) {

return "null";

}

// Convert set to an array and sort it in descending order

Character[] uniqueChars = charSet.toArray(new Character[0]);

java.util.Arrays.sort(uniqueChars, java.util.Collections.reverseOrder());

// Build the result string

StringBuilder result = new StringBuilder();

for (char c : uniqueChars) {

result.append(c);

}

return result.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt for the first input

String input1 = scanner.nextLine();

// Prompt for the second input

String input2 = scanner.nextLine();

// Process and display the result

String result = processStrings(input1, input2);

System.out.println(result);

scanner.close();

}

}

