**Name – Baljinder singh**

**Roll No - 21124022**

**Information Technology Group-1**

**Java Programming Lab [ITPC-226]**

**Assignment Sheet: Week 4,5,6**

**Q1) You are required to store a table that contains substance name along with its parameters such as freezing point and boiling point using 2-D matrix where each row is corresponding to substance and columns correspond to parameters. Design a class that takes temperature input using scanner class and check whether temperature match with freezing point or boiling point of respective substance and display the result**

Code:

import java.util.Scanner;

class Prob1{

public static void main(String[] args) {

String []name = new String[5];

int [][]data = new int[5][2];

Scanner s = new Scanner(System.in);

//[]name = {"One" , "Two" , "Three" , "Four" , "Five"};

for(int i = 0; i<5 ; i++ ){

System.out.println("Enter Substance name :");

name[i] = s.next();

System.out.println("Enter Boiling and Freezing point :");

data[i][0] = s.nextInt();

data[i][1] = s.nextInt();

}

for(int i = 0; i<5 ; i++ ){

System.out.println("Substance name: " + name[i] + " " + "temps: " + data[i][0] + " " + data[i][1]);

}

System.out.println("Enter a temperature to match:");

int key = s.nextInt();

boolean check = true;

for(int i = 0; i<5 ; i++ ){

if(key == data[i][0] || key == data[i][1]){

System.out.println("Match Found for substance: " + name[i] + " " + "temps: " + data[i][0] + " " + data[i][1]);

check = false;

}

}

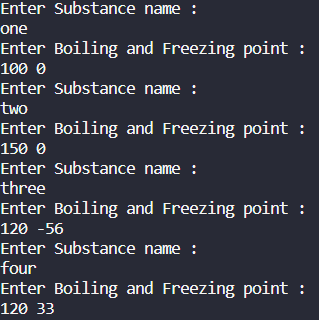
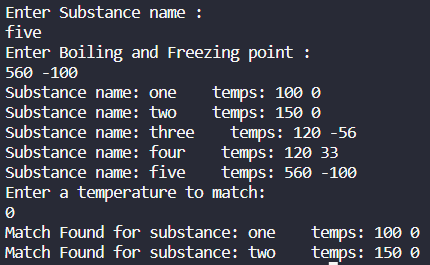
if(check)

System.out.println("No Match Found");

}

}

Output:

**Q2) Take a JaggedArray representing 2-D matrix as an input from the user. Write a program that transpose the Jagged matrix. Merge the original and transpose Ragged matrix. Display all the matrices.**

Code:

import java.util.\*;

public class Prob2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int rows = scanner.nextInt();

int[][] jaggedMatrix = new int[rows][];

for (int i = 0; i < rows; i++) {

System.out.print("Enter the number of elements in row " + (i+1) + ": ");

int cols = scanner.nextInt();

jaggedMatrix[i] = new int[cols];

System.out.print("Enter the elements in row " + (i+1) + ": ");

for (int j = 0; j < cols; j++) {

jaggedMatrix[i][j] = scanner.nextInt();

}

}

int[][] transposedMatrix = new int[jaggedMatrix[0].length][];

for (int i = 0; i < transposedMatrix.length; i++) {

transposedMatrix[i] = new int[jaggedMatrix.length];

for (int j = 0; j < jaggedMatrix.length; j++) {

if (jaggedMatrix[j].length > i) {

transposedMatrix[i][j] = jaggedMatrix[j][i];

} else {

transposedMatrix[i][j] = 0;

}

}

}

int[][] mergedMatrix = new int[rows][];

for (int i = 0; i < rows; i++) {

mergedMatrix[i] = new int[jaggedMatrix[i].length + transposedMatrix[i].length];

System.arraycopy(jaggedMatrix[i], 0, mergedMatrix[i], 0, jaggedMatrix[i].length);

System.arraycopy(transposedMatrix[i], 0, mergedMatrix[i], jaggedMatrix[i].length, transposedMatrix[i].length);

}

System.out.println("Original jagged matrix:");

displayMatrix(jaggedMatrix);

System.out.println("Transposed jagged matrix:");

displayMatrix(transposedMatrix);

System.out.println("Merged jagged matrix:");

displayMatrix(mergedMatrix);

}

public static void displayMatrix(int[][] matrix) {

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix[i].length; j++) {

System.out.print(matrix[i][j] + " ");

}

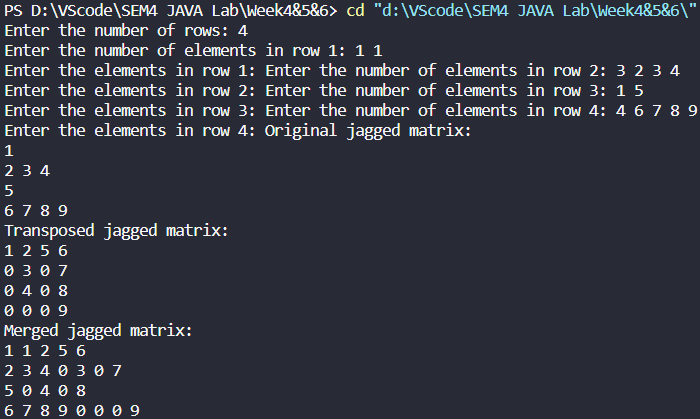
System.out.println();

}

}

}

Output:



**Q3) Write java program to create array of string of 10 elements. Further sort the array in ascending order. The string comparison can be with or without case sensitive**

Code:

import java.util.\*;

public class StringArraySorting {

public static void main(String[] args) {

String[] strArray = {"apple", "banana", "Cherry", "date", "Eggplant", "fig", "Grape", "Honey", "kiwi", "lemon"};

// Sorting the array in ascending order (case-insensitive)

for (int i = 0; i < strArray.length - 1; i++) {

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

if (strArray[i].compareToIgnoreCase(strArray[j]) > 0) {

String temp = strArray[i];

strArray[i] = strArray[j];

strArray[j] = temp;

}

}

}

// Printing the sorted array

System.out.println("Sorted Array:");

for (String str : strArray) {

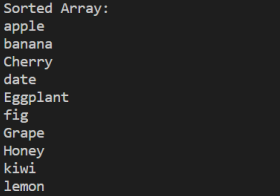
System.out.println(str);

}

}

}

Output:



**Q4) You are given a paragraph consisting of words, Convert this para graph to tokens (word) to be stored in array of string in sorted order where sorted array does not have duplicate words.**

Code:

import java.util.Arrays;

import java.util.\*;

public class Tokenizer {

public static void main(String[] args) {

String paragraph = "This is a sample paragraph. It contains some words that may be repeated.";

String[] words = paragraph.split("\\s+");

Arrays.sort(words);

int uniqueCount = 0;

for (int i = 1; i < words.length; i++) {

if (!words[i].equals(words[uniqueCount])) {

uniqueCount++;

words[uniqueCount] = words[i];

}

}

String[] uniqueWords = Arrays.copyOfRange(words, 0, uniqueCount + 1);

for (String word : uniqueWords) {

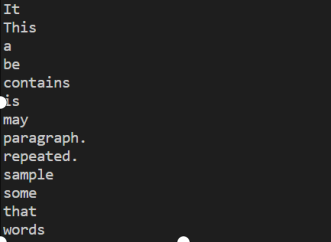
System.out.println(word);

}

}

}

Output:



**Q5) Write a Java programs to demonstrate (i) 0-arguments constructor (ii) parameterized constructor (iii) constructor overloading (iv) constructor calling using super keyword (v) Automatic constructor creation of parent class while creating the instance of child class. Also represent various error messages due to inappropriate use of constructor.**

Code

import java.util.\*;

public class ConstructorDemo {

// 0-arguments constructor

public ConstructorDemo() {

System.out.println("0-arguments constructor called");

}

// Parameterized constructor

public ConstructorDemo(String message) {

System.out.println("Parameterized constructor called with message: " + message);

}

// Constructor overloading

public ConstructorDemo(int number) {

System.out.println("Constructor with int parameter called with number: " + number);

}

// Constructor calling using super keyword

public static class ChildConstructorDemo extends ConstructorDemo {

public ChildConstructorDemo() {

super("Message from child constructor");

System.out.println("Child constructor called");

}

}

public static void main(String[] args) {

ConstructorDemo obj1 = new ConstructorDemo();

ConstructorDemo obj2 = new ConstructorDemo("Hello world");

ConstructorDemo obj3 = new ConstructorDemo(42);

ChildConstructorDemo obj4 = new ChildConstructorDemo();

// Demonstrate automatic constructor creation of parent class while creating the instance of child class

ChildConstructorDemo obj5 = new ChildConstructorDemo() {

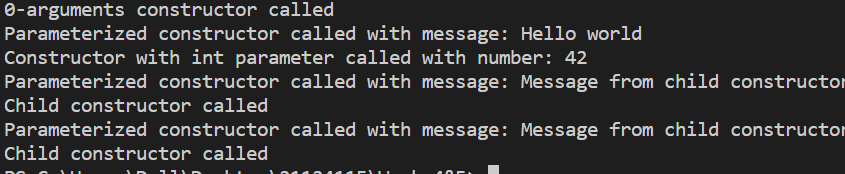
public void doSomething() {

System.out.println("Doing something");

}};

}}

Output:



**Q6) You are given a paragraph consisting continuous alphabets, convert this para graph to tokens based on occurrence of particular characters. Store the tokens in array of string in sorted order where sorted array does not have duplicate words.**

Code:

import java.util.\*;

public class prob6{

public static String[] tokenizeParagraph(String paragraph, char delimiter) {

Set<String> words = new HashSet<>();

StringBuilder sb = new StringBuilder();

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

if (c == delimiter) {

String word = sb.toString().trim();

if (!word.isEmpty()) {

words.add(word);

}

sb.setLength(0);

} else {

sb.append(c);

}}

// Add the last word (if any)

String lastWord = sb.toString().trim();

if (!lastWord.isEmpty()) {

words.add(lastWord);

}

// Convert the set to a sorted array

String[] result = words.toArray(new String[0]);

Arrays.sort(result);

return result;

}

public static void main(String[] args) {

String paragraph = "HellojTherejJava";

String[] token = tokenizeParagraph(paragraph, 'j');

System.out.println(Arrays.toString(token));

}}

Output:



**Q7) Apply the concept of Abstract class. You are required to write a base class Student having details (name, registration number, father’s name, address (current address, permanent address), and contact details (phone number, email Id). UGStudent, PGStudent, and PhDStudent extend the Student class they have details about the past qualifying degree and marks. Make each attribute protected and provide suitable getter and setter to access attributes.**

Code:

abstract class Student

{

Course course\_obj;

Payment payment\_obj;

protected String name,father\_name,curr\_address,per\_address,email\_id;

protected int regNo,phoneNum;

}

class UGStudent extends Student{

protected String pastQualifying="class12th";

protected float CGPA\_12th;

void setName(String s)

{

name=s;

}

void setFatherName(String s)

{

father\_name=s;

}

void setCurrAdd(String s)

{

curr\_address=s;

}

void setPerAdd(String s)

{

per\_address=s;

}

void setEmail(String s)

{

email\_id=s;

}

void setRegNum(int s)

{

regNo=s;

}

void setPhoneNum(int s)

{

phoneNum=s;

}

void setCGPA(float f)

{

CGPA\_12th=f;

}

String getName(){

return name;

}

String getFatherName(){

return father\_name;

}

String getCurrAdd(){

return curr\_address;

}

String getPerAdd(){

return per\_address;

}

String getEmail(){

return email\_id;

}

int getRegNum(){

return regNo;

}

int getPhoneNum(){

return phoneNum;

}

String getPastQualifying(){

return pastQualifying;

}

float getPastCgpa(){

return CGPA\_12th;

}

}

class PGStudent extends Student{

protected String pastQualifying="UG";

protected float CGPA\_UG;

void setName(String s)

{

name=s;

}

void setFatherName(String s)

{

father\_name=s;

}

void setCurrAdd(String s)

{

curr\_address=s;

}

void setPerAdd(String s)

{

per\_address=s;

}

void setEmail(String s)

{

email\_id=s;

}

void setRegNum(int s)

{

regNo=s;

}

void setPhoneNum(int s)

{

phoneNum=s;

}

void setCGPA(float f)

{

CGPA\_UG=f;

}

String getName(){

return name;

}

String getFatherName(){

return father\_name;

}

String getCurrAdd(){

return curr\_address;

}

String getPerAdd(){

return per\_address;

}

String getEmail(){

return email\_id;

}

int getRegNum(){

return regNo;

}

int getPhoneNum(){

return phoneNum;

}

String getPastQualifying(){

return pastQualifying;

}

float getPastCgpa(){

return CGPA\_UG;

}

}

class PhDStudent extends Student{

protected String pastQualifying="PG";

protected float CGPA\_PG;

void setName(String s)

{

name=s;

}

void setFatherName(String s)

{

father\_name=s;

}

void setCurrAdd(String s)

{

curr\_address=s;

}

void setPerAdd(String s)

{

per\_address=s;

}

void setEmail(String s)

{

email\_id=s;

}

void setRegNum(int s)

{

regNo=s;

}

void setPhoneNum(int s)

{

phoneNum=s;

}

void setCGPA(float f)

{

CGPA\_PG=f;

}

String getName(){

return name;

}

String getFatherName(){

return father\_name;

}

String getCurrAdd(){

return curr\_address;

}

String getPerAdd(){

return per\_address;

}

String getEmail(){

return email\_id;

}

int getRegNum(){

return regNo;

}

int getPhoneNum(){

return phoneNum;

}

String getPastQualifying(){

return pastQualifying;

}

float getPastCgpa(){

return CGPA\_PG;

}

}

**Q8) Apply the concept of Abstract class. (Continuation with Q. 7) Write another base class Course which stores details about the courses registered by a Student. Course has attributes such as code, name, and credit. UGCourse and PGCourse extends the Course class. The course code follows the following syntax PGxxxx and UG-xxxx here x is a number which uniquely identifies a course. Make each attribute protected and provide suitable getter and setter to access attributes.**

Code:

abstract class Course

{

protected String code,name;

protected int credit;

}

class UGCourse extends Course

{

void setCode(int s)

{

code="UG"+ String.valueOf(s);

}

void setName(String s)

{

name=s;

}

void setCredit(int s)

{

credit=s;

}

String getName(){

return name;

}

String getCode(){

return code;

}

int getCredit(){

return credit;

}

}

class PGCourse extends Course

{

void setCode(int s)

{

code="PG"+ String.valueOf(s);

}

void setName(String s)

{

name=s;

}

void setCredit(int s)

{

credit=s;

}

String getName(){

return name;

}

String getCode(){

return code;

}

int getCredit(){

return credit;

}

}

**Q9) Apply the concept of Abstract class. (Continuation with Q. 8) Write another base class Payment which stores the payment details of the registered Student. Payment has a single attribute paymentID which is unique for each payment. PaymentDD, PaymentUPI, and PaymentNB extends the Payment class, these classes stores details of payment using Demand Draft, UPI, and Net Banking. Add relevant attributes to these classes to store payment details. Make each attribute protected and provide suitable getter and setter to access attributes.**

Code:

abstract class Payment{

protected int PaymentID;

}

class PaymentDD extends Payment{

protected String PayMethod="Demand Draft";

protected double Amount;

void setPaymentID(int s)

{

PaymentID=s;

}

void setAmount(double s)

{

Amount=s;

}

String getPayMethod()

{

return PayMethod;

}

int getPaymentID()

{

return PaymentID;

}

double getAmount()

{

return Amount;

}

}

class PaymentUPI extends Payment{

protected String PayMethod="UPI";

protected double Amount;

void setPaymentID(int s)

{

PaymentID=s;

}

void setAmount(double s)

{

Amount=s;

}

String getPayMethod()

{

return PayMethod;

}

int getPaymentID()

{

return PaymentID;

}

double getAmount()

{

return Amount;

}

}

class PaymentNB extends Payment{

protected String PayMethod="Net Banking";

protected double Amount;

void setPaymentID(int s)

{

PaymentID=s;

}

void setAmount(double s)

{

Amount=s;

}

String getPayMethod()

{

return PayMethod;

}

int getPaymentID()

{

return PaymentID;

}

double getAmount()

{

return Amount;

}

}

**Q11) Write java program to demonstrate the different ways for making an object eligible for GC when it is no longer needed.**

Code:

import java.util.\*;

public class prob11 {

public static void main(String[] args) {

Object obj1 = new Object();

Object obj2 = new Object();

Object obj3 = new Object();

Object obj4 = new Object();

obj1 = null;

long[] bigArray = new long[1000000];

obj2 = null;

obj3 = null;

System.gc();

Object obj5 = new Object();

}

}

**Q12) (Continuation with Q. 11) use System.gc() method and Runtime.getRuntime().gc() method to achieve garbage collection. (You may or may not get expected outcome)**

CODE:

public class prob12 {

public static void main(String[] args) {

System.out.println("Executing Runtime.getRuntime().gc() Method");

// print when the program starts

System.out.println("Program starting...");

// run the garbage collector

System.out.println("Running Garbage Collector...");

Runtime.getRuntime().gc();

System.out.println("Completed.");

System.out.println("Executing System.gc() Method");

int arr1[] = { 0, 1, 2, 3, 4, 5 };

int arr2[] = { 0, 10, 20, 30, 40, 50 };

// copies an array from the specified source array

System.arraycopy(arr1, 0, arr2, 0, 1);

System.out.print("array2 = ");

System.out.print(arr2[0] + " ");

System.out.println(arr2[1] + " ");

// it runs the GarbageCollector

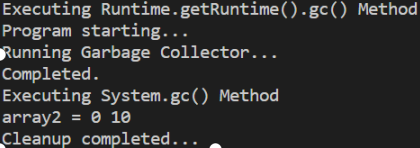
System.gc();

System.out.println("Cleanup completed...");

}

}

Output:



**Q13) Create a package P1 containing a class named Person, which has the following fields:**

**id, name, age, phoneNumber, address, and experience. It also has a method named**

**printDetail(), which prints all the details of the Persons. Create two classes that belong**

**to the same package P1, Employee and staff, that inherit the Person class.**

**The Employee and Staff classes have data member basicSalary and**

**consolidatedSalary, respectively. Create an object of both the class and print the**

**details. Assume that the salary of an individual is computed using the following**

**rules:**

**a). If an individual is an employee with experience below five years, then the salary**

**= basicSalary + 100% of basic salary as DA +30% of Basic salary as HRA.**

**b). If an individual is an employee and has an experience above five years, then the**

**net salary = basic salary + 100% of basic salary as DA +30% of Basic salary as**

**HRA + 20% of basic salary \* (no. of years of experience above 5 years).**

**c. The salary of staff is equal to the consolidatedSalary.**

CODE:

class Person

{

int id;

String name;

int age;

long phoneno;

String address;

int experience;

Person(int id, String name, int age, long phoneno, String address, int experience)

{

this.id = id;

this.name = name;

this.age = age;

this.phoneno = phoneno;

this.address = address;

this.experience = experience;

}

void printDetails()

{

System.out.println("Person details are :");

System.out.println(id +" "+ name + " " + age + " " + phoneno + " " + address + " " + experience);

}

}

class Employee extends Person

{

int basicSalary;

long salary;

Employee(int id, String name, int age, long phoneno, String address, int experience, int basicSalary)

{

super(id,name,age,phoneno,address,experience);

this.id = id;

this.name = name;

this.age = age;

this.phoneno = phoneno;

this.address = address;

this.experience = experience;

this.basicSalary = basicSalary;

}

double getSalary(int experience)

{

if(experience < 5)

{

return basicSalary + basicSalary + 0.3\*basicSalary;

}

else

{

return basicSalary + basicSalary + 0.3\*basicSalary + 0.2\*basicSalary \* (this.experience - 5);

}

}

void printDetails()

{

System.out.println("Person details are :");

System.out.println(id + name + age + phoneno + address + experience + getSalary(experience));

}

}

class Staff extends Person

{

int consolidatedSalary;

long salary;

Staff(int id, String name, int age, long phoneno, String address, int experience, int consolidatedSalary)

{

super(id,name,age,phoneno,address,experience);

this.id = id;

this.name = name;

this.age = age;

this.phoneno = phoneno;

this.address = address;

this.experience = experience;

this.consolidatedSalary = consolidatedSalary;

}

double getSalary(int experience)

{

return consolidatedSalary;

}

void printDetails()

{

System.out.println("Person details are :");

System.out.println(id +" "+name+" " + age +" "+ phoneno + " " + address + " " + experience + " " + getSalary(experience));

}

}

class prob13

{

public static void main(String[] args)

{

Employee E = new Employee(1,"ANON",19,233333,"Jalandhar",8,10000);

Staff S;

E.printDetails();

}

}

Output:

