1. what is metaspace and heap memory

Ans: **Metaspace** is a new memory space – starting from the Java 8 version; it has replaced the older PermGen memory space. The most significant difference is how it handles memory allocation. Specifically, this native memory region grows automatically by default.

**Heap memory**, also known as “dynamic” memory, is an alternative to local stack memory. Local memory is quite automatic. Local variables are allocated automatically when a function is called, and they are deallocated automatically when the function exits.

1. generate multiples of 2 until 20 using recursive function

Ans: **package** com.pack;

**import** java.util.Scanner;

**public** **class** tableoftwo {

**static** **void** MultiplicationTable(**int** number, **int** i) {

System.***out***.println(number + "\*" + i + "=" + number \* i);

**if** (i < 10) {

*MultiplicationTable*(number, i + 1);

}

}

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

Scanner sc = **new** Scanner(System.***in***);

**int** number;

System.***out***.println("Enter a number: ");

number = sc.nextInt();

System.***out***.println("Multiplication Table of " + number + " is: ");

*MultiplicationTable*(number, 1);

sc.close();

}

}

1. check if two strings are equal or not

Ans: **package** com.pack;

**import** java.util.Scanner;

**public** **class** twostring {

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

String str1, str2;

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("Enter first String");

str1 = scanner.nextLine();

System.***out***.println("Enter second String");

str2 = scanner.nextLine();

**if** (str1.equals(str2))

System.***out***.print("Equal Strings");

**else**

System.***out***.print("UnEqual Strings");

}

}

**4**. print the character count in a string say

string s ="helloworld" print h-1, e-1, l-3,o-2

Ans: **package** com.pack;

**public** **class** CharCount {

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

String str = "helloworld";

**int** count[] = **new** **int**[256];

**int** len = str.length();

**for** (**int** i = 0; i < len; i++) {

count[str.charAt(i)]++;

}

**char** ch[] = **new** **char**[str.length()];

**for** (**int** i = 0; i < len; i++) {

ch[i] = str.charAt(i);

**int** find = 0;

**for** (**int** j = 0; j <= i; j++) {

**if** (str.charAt(i) == ch[j]) {

find++;

}

}

**if** (find == 1) {

System.***out***.println(str.charAt(i) + " - " + count[str.charAt(i)]);

}

}

}

}

1. why java is platform independent

Ans: Java is platform-independent because it does not depend on any type of platform. Hence, Java is platform-independent language. In Java, programs are compiled into byte code and that byte code is platform-independent. ... Any machine to execute the byte code needs the Java Virtual Machine.

1. can we create class as final

Ans: You can declare some or all of a class's methods final. You use the final keyword in a method declaration to indicate that the method cannot be overridden by subclasses. The Object class does this—a number of its methods are final .

The main purpose of using a class being declared as final is to prevent the class from being subclassed. If a class is marked as final then no class can inherit any feature from the final class. You cannot extend a final class. If you try it gives you a compile time error.

**7**. considder we have employee class with empid, empname and salary

and list of employees get the the highest salary paid employee data

Ans:

package com.pack;

import java.util.Arrays;

import java.util.List;

public class Employee {

private long id;

private String name;

private int salary;

public Employee(long id, String name, int salary) {

this.id = id;

this.name = name;

this.salary = salary;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

@Override

public String toString() {

return "Employee [id=" + id + "," + " name=" + name + "," + " salary=" + salary + "]";

}

}

class FindEmployee {

public static void main(String[] args) {

List<Employee> employees = Arrays.asList(new Employee(101, "Jay", 5000), new Employee(109, "Atul", 3000),

new Employee(111, "Sourav", 4400));

int maxSalary = employees.stream().map(Employee::getSalary).max(Integer::compare).get();

System.out.println("Max salary of the employee:" + maxSalary);

System.out.print("Employee details:");

employees.stream().filter(emp -> emp.getSalary() == maxSalary).forEach(System.out::println);

}

}

1. consider a list of duplicate values remove duplicate value and get unique values from the list

Ans: **package** com.pack;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.stream.Collectors;

**public** **class** Listp {

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

List<Integer> lst = **new** ArrayList<>();

lst.add(1);

lst.add(3);

lst.add(5);

lst.add(3);

lst.add(5);

lst.add(4);

List<Integer> dlst = lst.stream().distinct().collect(Collectors.*toList*());

System.***out***.println(dlst);

}

}

1. can we write try and finally without catch block what is the use

Ans: Yes, we can have try without catch block by using finally block. You can use try with finally. As you know finally block always executes even if you have exception or return statement in try block except in case of System.

1. Welcome to College of Management

Ans:

**package** com.pack;

**import** java.util.Scanner;

**public** **class** Application {

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

System.***out***.println("Welcome to College of Management");

**try** (Scanner sc = **new** Scanner(System.***in***)) {

**int** studentId;

**boolean** n = **true**;

String name = "Aditya";

**int** hindi;

**int** english;

**int** maths;

**int** science;

**int** social;

**int** total;

**double** percentage;

**while** (n) {

System.***out***.println("Do you want to continue ");

**char** go = sc.next().charAt(0);

**if** (go == 'y') {

System.***out***.println("enter valid choice");

System.***out***.println("c - check student result");

System.***out***.println("a - add student result");

System.***out***.println("x - exit");

**char** choice = sc.next().charAt(0);

**switch** (choice) {

**case** 'a':

System.***out***.println("Add student result");

System.***out***.println("StudentId-");

studentId = sc.nextInt();

System.***out***.println("marks in hindi-");

hindi = sc.nextInt();

System.***out***.println("marks in english-");

english = sc.nextInt();

System.***out***.println("marks in maths-");

maths = sc.nextInt();

System.***out***.println("marks in science-");

science = sc.nextInt();

System.***out***.println("marks in social-");

social = sc.nextInt();

System.***out***.println("Student added successfully");

**break**;

**case** 'c':

System.***out***.println("Check student result");

System.***out***.println("Enter Student ID=");

studentId = sc.nextInt();

**if** (studentId != 38) {

System.***out***.println("Student id not found");

} **else** {

hindi = 50;

english = 90;

maths = 60;

science = 40;

social = 85;

total = hindi + english + maths + science + social;

percentage = (total \* 100) / 500.00;

System.***out***.println("Student Result-");

System.***out***.println("id-" + studentId);

System.***out***.println("studentname-" + name);

System.***out***.println("marks in hindi-" + hindi);

System.***out***.println("marks in english-" + english);

System.***out***.println("marks in maths-" + maths);

System.***out***.println("marks in science-" + science);

System.***out***.println("marks in social-" + social);

**if** (percentage >= 50)

System.***out***.println("result-pass");

**else**

System.***out***.println("result-fail");

System.***out***.println("total-" + total);

System.***out***.println("percentage-" + percentage);

}

**break**;

**case** 'x':

**return**;

**default**:

System.***out***.println("Invalid Input");

}

} **else** {

System.***out***.println("bye");

**break**;

}

}

}

}

}