1. **What is metaspace and heap memory?**

Metaspace is the memory, which virtual machine uses to store class metadata. Class metadata are the runtime representation of java classes within a JVM process.

Heap memory is the run time data area from which the memory for all java class instances and arrays is allocated. This memory is allocated for all class instances and array.

1. **Generate multiples of 2 until 20 using recursive function.**

**package** com.pack;

**public** **class** table {

**static** **void** mul\_table(**int** N, **int** i)

{

**if** (i > 10)

**return** ;

System.***out***.println(N \* i);

*mul\_table*(N, i + 1);

}

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

{

**int** N = 2;

*mul\_table*(N, 1);

}

}

1. **Check if two strings are equal or not.**

|  |
| --- |
| **package** com.pack;  **import** java.util.Scanner;  **public** **class** CompString {    **public** **static** **void** main(String[] args) {  Scanner scanner = **new** Scanner(System. ***in***);    System.***out***.print("Enter first string : ");  String str1 = scanner.nextLine();    System.***out***.print("Enter second string : ");  String str2 = scanner.nextLine();    **boolean** areTwoStringsEqual = str1.equals(str2);    System.***out***.print("Two strings are equal : "+areTwoStringsEqual);  }  } |

1. **Print the character count in a string say string s ="helloworld" print h-0, e-1, l-3,o-2**

**package** com.lambda;

**public** **class** OccurenceOfChar {

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

String str = "helloworld";

**int**[] freq= **new** **int**[128];

**for**(**char** ch:str.toCharArray()){

freq[ch]++;

}

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

**if**(freq[i] != 0) {

System.***out***.println("char: " + (**char**) i + "count = " + freq[i]);

}

}

}

}

1. **Why java is platform independent?**

* Whenever we compile a code in java, javac compiles the code and it creates an intermediate code called Byte Code.
* This creates a .class file, which is considered as byte code. This byte code is not executable. JVM acts as an interpreter and then executes the byte code generated by javac.
* The byte code generated by source code compilation would run in any operating system, but the JVM present in a machine differs for each operating system. This is how java is considered a platform-independent programming language.

1. **Can we create class as final?**

Yes, a class can be made final by using the final keyword. A final class cannot be extended by any other class. The final class cannot be inherited and so the final keyword is commonly used with a class to prevent inheritance.

1. **Consider we have employee class with empid, empname and salary and list of employees get the the highest salary paid employee data.**

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(100, "Smith", 50000), new Employee(200, "John", 30000),

new Employee(300, "Tom", 40000));

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.**

**package** com.lambda;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**public** **class** MyInterface{

**public** **static** <D> ArrayList<D> removeDuplicates(ArrayList<D> list)

{

ArrayList<D> newList = **new** ArrayList<D>();

**for** (D element : list)

{

**if** (!newList.contains(element))

{

newList.add(element);

}

}

**return** newList;

}

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

{

ArrayList<Integer>

list = **new** ArrayList<>(Arrays .*asList*(5,2,4,6,5,4,7));

System.***out***.println("ArrayList with duplicates: " + list);

ArrayList<Integer>

newList = *removeDuplicates*(list);

System.***out***.println("ArrayList with duplicates removed: " + newList);

}

}

1. **Can we write try and finally without catch block? What is the use?**

Yes, It is possible to have a try block without a catch block by using a final block. A final block will always execute even there is an exception occurred in a try block, except System.exit() it will execute always.

**Example**

public class TryBlockWithoutCatch {

   public static void main(String[] args) {

      try {

         System.out.println("Try Block");

      } finally {

         System.out.println("Finally Block");

      }

   }

}

**Output**

Try Block

Finally Block