**What is JAVA? What it is used for?**

**Java** is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies.

It is a computing platform for application development.

Java is fast, secure, and reliable, therefore, it is widely used for developing Java applications in laptops, data centres, game consoles, scientific supercomputers, cell phones, etc.

USES:

* It is used for developing Android Apps
* Helps you to create Enterprise Software
* Wide range of Mobile java Applications
* Scientific Computing Applications
* Use for Big Data Analytics
* Java Programming of Hardware devices
* Used for Server-Side Technologies like Apache, JBoss, GlassFish, etc.

**How to install java in your machine?**

**Step 1)** Go to [link](https://www.oracle.com/java/technologies/javase-downloads.html). Click on JDK Download for Java download JDK 8.

**Step 2)** Next,

1. Accept License Agreement
2. Download Java 8 JDK for your version JDK download 64 bit.

**Step 3)**When you click on the Installation link the popup will be open. Click on I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE development kit and you will be redirected to the login page. If you don’t have an oracle account you can easily sign up by adding basics details of yours.

**Step 4)**Once the Java JDK 8 download is complete, run the exe for install JDK. Click Next

**Step 5)** Select the PATH to install Java in Windows. You can leave it Default. Click next. Once you install Java in windows, click Close.

**What is java, javac, javap command for?**

* java -used to execute the byte code of java. It takes byte code as input and runs it and produces the output.
* javac - responsible to compile Java programs, it takes .java file as input and produces byte code.
* javap - can be used to help developers understand the mechanism of the Java compiler

**What is jar and war?**

**JAR** stands for **Java ARchive.jar files:** The .jar files **contain libraries, resources and accessories files** like property files.

WAR stands for Web application Archive. The war file **contains the web application** that can be deployed on any servlet/jsp container. The .war file **contains jsp, html, javascript** and other files necessary for the development of web applications.

**What is platform independent in java?**

The meaning of platform-independent is that the java compiled code (byte code) can run on all operating systems.  In other words, you can write your code once and then run it anywhere, on any platform that provides the environment to run it. This environment is the Java Virtual Machine (JVM). The JVM should be present to execute the code.

**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.

**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);

}

}

**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);  }  } |

**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]);

}

}

}

}

**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.

**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.

**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);

}

}

**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);

}

}

**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

**Create a java application as mentioned above flow**

Welcome to College Management

A: Add Student Result

X: Exit

a

Enter student id:

10

Enter student Name:

durga

Enter marks in hindi:

1

Enter marks in english:

1

Enter marks in maths:

1

Enter marks in science:

1

Enter marks in social:

1

Student added successfully.

Do you want to continue (Y/N):

C: Check student Result

A: Add Student Result

X: Exit

c

Enter student id to check result:

1

student id not found

Do you want to continue (Y/N):

y

C: Check student Result

A: Add Student Result

X: Exit

c

Enter student id to check result:

10

Student Result{id='10', name='null', marks=Subject Marks{hindi=1, english=1, maths=1, science=1, social=1}, result='Pass', total=5, percentage=100.0}

Do you want to continue (Y/N):

n

Bye

Process finished with exit code 1

|  |
| --- |
| **package** com.pack;  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.Scanner;  **import** java.util.stream.Collectors;  **public** **class** StudentApp {  **public** **static** **void** main(String[] args) {  List<Employee> empList = **new** ArrayList<>();  List<Employee> dlst = empList.stream().distinct().collect(Collectors.*toList*());  System.***out***.println("Welcome to College Management");  **char** choice;  String result;  **int** count = 0;  **do** {  **if** (count > 0) {  System.***out***.println("C : Check Student Result");  }  System.***out***.println("A : Add Student Result");  System.***out***.println("X : Exit");  Scanner sc = **new** Scanner(System.***in***);  String c = sc.next();  **switch** (c.toUpperCase()) {  **case** "A":  System.***out***.println("Enter student id");  **int** id = sc.nextInt();  System.***out***.println("Enter student Name");  String name = sc.next();  System.***out***.println("Enter marks in Hindi");  **int** hindi = sc.nextInt();  System.***out***.println("Enter marks in English");  **int** eng = sc.nextInt();  System.***out***.println("Enter marks in Maths");  **int** maths = sc.nextInt();  System.***out***.println("Enter marks in Science");  **int** science = sc.nextInt();  System.***out***.println("Enter marks in Social");  **int** social = sc.nextInt();  **int** total = hindi+eng+maths+science+social;  **float** per = (**float**) total\*100 / 500;    **if** (per > 50) {  result="Pass";  } **else** {  result="Fail";  }  dlst.add(**new** Employee(id, name,hindi,eng,maths,science,social,result,total,per));  System.***out***.println("Student added successfully");  count++;  **break**;  **case** "C":  System.***out***.println("Enter Student Id to Check result: ");  **int** in=sc.nextInt();  **for**(Employee e:dlst)  {  **if**(e.getEmpId()==(in))  {  dlst.stream().filter(emp->emp.getEmpId()==in).forEach(System.***out***::println);  System.*exit*(0);  }  }  System.***out***.println("No such data found");  **break**;  **case** "X":  System.*exit*(1);  **default**:  System.***out***.println("Invalid choice");  **break**;  }  System.***out***.println("Do you want to continue (Y/N)");  choice = sc.next().charAt(0);  } **while** (choice == 'Y' || choice == 'y');  System.***out***.println("Bye");  System.*exit*(1);    }  @Override  **public** String toString() {  **return** "MyClass []";  }    }  **class** Employee {  **private** **int** empId;  **private** String empNm;  **private** **int** m1, m2, m3, m4, m5;  **private** String r;  **private** **float** per;  **private** **int** t;  Scanner sc = **new** Scanner(System.***in***);  **public** Employee() {  }  **public** Employee(**int** empId, String empNm, **int** m1, **int** m2, **int** m3, **int** m4, **int** m5, String r,**int** t, **float** per ) {  **super**();  **this**.empId = empId;  **this**.empNm = empNm;  **this**.m1 = m1;  **this**.m2 = m2;  **this**.m3 = m3;  **this**.m4 = m4;  **this**.m5 = m5;  **this**.per=per;  **this**.r=r;  **this**.t=t;  }  **public** **int** getEmpId() {  **return** empId;  }  **public** **void** setEmpId(**int** empId) {  **this**.empId = empId;  }  **public** String getEmpNm() {  **return** empNm;  }  **public** **void** setEmpNm(String empNm) {  **this**.empNm = empNm;  }  **public** **int** getm1() {  **return** m1;  }  **public** **void** setm1(**int** m1) {  **this**.m1 = m1;  }  **public** **int** getm2() {  **return** m2;  }  **public** **void** setm2(**int** m2) {  **this**.m2 = m2;  }  **public** **int** getm3() {  **return** m3;  }  **public** **void** setm3(**int** m3) {  **this**.m3 = m3;  }  **public** **int** getm4() {  **return** m4;  }  **public** **void** setm4(**int** m4) {  **this**.m4 = m4;  }  **public** **int** getm5() {  **return** m5;  }  **public** **void** setm5(**int** m5) {  **this**.m5 = m5;  }  **public** String getR() {  **return** r;  }  **public** **void** setR(String r) {  **this**.r = r;  }  **public** **float** getPer() {  **return** per;  }  **public** **void** setPer(**float** per) {  **this**.per = per;  }  **public** **int** getT() {  **return** t;  }  **public** **void** setT(**int** t) {  **this**.t = t;  }  @Override  **public** String toString() {  **return** "Student Result {Id=" + empId + ", name=" + empNm + ",marks=Subject Marks{ Hindi=" + m1 + ", English=" + m2 + ", maths=" + m3 + ", Science="  + m4 + ", Social=" + m5 + "}, result=" + r +", total=" + t + ", percentage=" + per + "}";  }    } |

**What is garbage collector and how it works?**

Garbage collection in Java is the process by which Java programs perform automatic memory management. Java programs compile to byte code that can be run on JVM. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

Automatic garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects.

**What is Heap space?**

**The Java heap is the area of memory used to store objects instantiated by applications running on the JVM.**When the JVM is started, heap memory is created and any objects in the heap can be shared between threads as long as the application is running.

**What is java memory model?**

The Java memory model specifies how the Java virtual machine works with the computer's memory (RAM). The Java virtual machine is a model of a whole computer so this model naturally includes a memory model -Java memory model.  The Java memory model specifies how and when different threads can see values written to shared variables by other threads, and how to synchronize access to shared variables when necessary.

**What is young and old generations? What is Eden and Survivor space?**

* The *Young Generation* is where all new objects are allocated and aged. When the young generation fills up, this causes a minor garbage collection. A young generation full of dead objects is collected very quickly. Some survived objects are aged and eventually move to the old generation.
* The Old Generation is used to store long surviving objects. Typically, a threshold is set for young generation object and when that age is met, the object gets moved to the old generation. Eventually the old generation needs to be collected. This event is called a major garbage collection.
* **Eden Space**: The pool from which memory is initially allocated for most objects.
* **Survivor Space**: The pool containing objects that have survived the garbage collection of the Eden space.