**Assignment no3**

1. **What is Java?**

Java is a programming language that James Gosling developed in 1991 initially with Sun Microsystems. Then released the advance version of it that is JDK 1.0 later in 23 Jan 1996. It was initially known as Oak language named after an Oak tree and then renamed JAVA. It is a flexible, reliable, portable and powerful language.

1. **What is a package in Java? List down various advantages of packages.**

A java package is a group of similar types of classes, interfaces and sub-packages.Packages are group of modules. Modules contain number of classes.

Advantages:

1. The advantage of using packages in java is we create our own folders in drives and import them whenever needed using import keyword.
2. Also we need not to write the same functions over and again in our new code.
3. **Explain JDK, JRE and JVM?**
4. JDK- JDK stands for Java Development Kit used for developing Java

applets and apps. It is basically a software development

environment.JDK includes Java Runtime Environment (JRE), a

compiler called (javac), a document generator (Javadoc), an

archiver (jar), and other tools required in development.

1. JRE- JRE refers to a runtime environment in which java bytecode can

be executed. It implements the JVM (Java Virtual Machine) and

provides all the class libraries and other support files that JVM

uses at runtime. So JRE is a software package that contains what

is required to run a Java program. Basically, it’s an

implementation of the JVM which physically exists.

1. JVM-It is the tool necessary to compile, document and package Java

programs. The JDK completely includes JRE which contains

tools for Java programmers. The Java Development Kit is

provided free of charge. Along with JRE, it includes an

interpreter, a compiler (javac), documentation generator and

other tools needed in Java development. In short, it contains JRE

+ development tools.

1. **Explain public static void main (String args[]) in Java.**

**Public:** It is an method/access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program.

**Static:** You can make a method static by using the keyword static. We should call the main() method without creating an object.

**Void:** In Java, every method has the return type. Void keyword acknowledges the compiler that main() method does not return any value.

**Main():** It is a default signature which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the main() method.

**String args[]:** The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array. Here, agrs[] is the array name, and it is of String type. It means that it can store a group of string. Values passed to the main() method is called arguments. These arguments are stored into args[] array, so the name args[] is generally used for it.

1. **What are the differences between C++ and Java?**

|  |  |  |
| --- | --- | --- |
|  | JAVA | C++ |
| a. | It is platform independent programming language. | It is platform dependent programming language |
| b. | It uses compiler and interpreter both. | It only uses compiler. |
| c. | Java don’t support the pointer concept. | C++ supports the pointer concept. |
| d. | Java supports method overloading and don’t support operator overloading. | C++ supports both method and operator overloading. |
| e. | It supports multiple inheritance. | It does not support multiple inheritance. |

**6. Why Java is platform independent?**

Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere(WORA).

**7. What are wrapper classes in Java?**

The **wrapper class in Java** provides the mechanism to convert primitive into object and object into primitive. Since J2SE 5.0, **autoboxing** and **unboxing** feature convert primitives into objects and objects into primitives automatically. The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing. Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc. Let us see the different scenarios, where we need to use the wrapper classes.

* Change the value in Method: Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
* Serialization: We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
* Synchronization: Java synchronization works with objects in Multithreading.
* java.util package: The java.util package provides the utility classes to deal with objects.
* Collection Framework: Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

The eight classes of the *java.lang* package are known as wrapper classes in Java. The list of eight wrapper classes are given below:

|  |  |
| --- | --- |
| Primitive Type | Wrapper class |
| boolean | [Boolean](https://www.javatpoint.com/java-boolean) |
| char | [Character](https://www.javatpoint.com/post/java-character) |
| byte | [Byte](https://www.javatpoint.com/java-byte) |
| short | [Short](https://www.javatpoint.com/java-short) |
| int | [Integer](https://www.javatpoint.com/java-integer) |
| long | [Long](https://www.javatpoint.com/java-long) |
| float | [Float](https://www.javatpoint.com/java-float) |
| double | [Double](https://www.javatpoint.com/java-double) |

## Autoboxing

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

Since Java 5, we do not need to use the valueOf() method of wrapper classes to convert the primitive into objects.

## Unboxing

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing. Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives.

## Custom Wrapper class in Java

Java Wrapper classes wrap the primitive data types, that is why it is known as wrapper classes. We can also create a class which wraps a primitive data type. So, we can create a custom wrapper class in Java.

**8. Why pointers are not used in Java?**

1. Memory access via pointer arithmetic: this is fundamentally unsafe. Java has a robust security model and disallows pointer arithmetic for the same reason. It would be impossible for the Virtual Machine to ensure that code containing pointer arithmetic is safe without expensive runtime checks.

2. Security: By not allowing pointers, Java effectively provides another level of abstraction to the developer. No pointer support make Java more secure because they point to memory location or used for memory management that loses the security as we use them directly.

3. Passing argument by reference: Passing a reference which allows you to change the value of a variable in the caller's scope. Java doesn't have this, but it's a pretty rare use case and can easily be done in other ways. This is in general equivalent to changing a field in an object scope that both the caller and callee can see.

4. Manual memory management: you can use pointers to manually control and allocate blocks of memory . This is useful for some bigger applications like games, device drivers etc. but for general purpose Object Oriented programming it is simply not worth the effort. Java instead provides very good automatic Garbage Collection (GC) which takes care of memory management.

**9. List some features of Java?**

A list of most important features of Java language is given below.

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic

**10.Why is Java Architectural Neutral?**

### Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

**11.How java enabled high performance in java?**

Java uses Just-In-Time compiler (JIT) to enable high performance. Just-In-Time compiler is a program that turns Java bytecode, which is a program that contains instructions that must be interpreted into instructions that can be sent directly to the processor.

Java enables the high performance as Java has Just-In-Compiler, which can compile same functionality byte code in same time. A Java program takes less time to compile than other programming language’s compiler.

The JIT compiler helps improve the performance of Java programs by compiling bytecodes into native machine code at run time. The JIT compiler is enabled by default. When a method has been compiled, the JVM calls the compiled code of that method directly instead of interpreting it.

**12. Why java considered as dynamic?**

Java is a dynamic language. ... It also supports functions from its native languages, i.e., C and C++. Java supports dynamic compilation and automatic memory management (garbage collection).

Java is considered dynamic because of Bytecode. The source code which is written in one platform that code can be executed in any platform. It loads the class file during runtime only. Hence, any thing that happens in runtime is dynamic.

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

**13What is java virtual machine and how it is considered as java independent feature?**

The JVM has two primary functions: to allow Java programs to run on any device or operating system (known as the "Write once, run anywhere" principle), and to manage and optimize program memory.

The JVM is the specification for a software program that executes code and provides the runtime environment for that code.

Java is a high level programming language. A program written in high level language cannot be run on any machine directly. First, it needs to be translated into that particular machine language. The javac compiler does this thing, it takes java program (.java file containing source code) and translates it into machine code (referred as byte code or .class file).

Java Virtual Machine (JVM) is a virtual machine that resides in the real machine (your computer) and the machine language for JVM is byte code. This makes it easier for compiler as it has to generate byte code for JVM rather than different machine code for each type of machine. JVM executes the byte code generated by compiler and produce output. JVM is the one that makes java platform independent.

So, now we understood that the primary function of JVM is to execute the byte code produced by compiler. Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems. Which means that the byte code generated on

Windows can be run on Mac OS and vice versa. That is why we call java as platform independent language. The same thing can be seen in the diagram below. Java Virtual Machine (JVM) is a virtual machine that resides in the real machine (your computer) and the machine language for JVM is byte code. This makes it easier for compiler as it has to generate byte code for JVM rather than different machine code for each type of machine. JVM executes the byte code generated by compiler and produce output. JVM is the one that makes java platform independent machine code for each type of machine. JVM executes the byte code generated by compiler and produce output. JVM is the one that makes java platform independent.

So, now we understood that the primary function of JVM is to execute the byte code produced by compiler. Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems. Which means that the byte code generated on Windows can be run on Mac OS and vice versa. That is why we call java as platform independent language.

**13.List two java IDE's?**

An Integrated Development Environment (IDE) is an application which facilitates the application development. In general, an IDE is a graphical user interface (GUI)-based workbench. It is designed to help a developer in building software applications with an environment combined with all the required tools at hand.

Best java IDE's:

1. eclipse
2. NetBeans
3. **why java called as platform?**

Java Platform is a collection of programs that help programmers to develop and run Java programming applications efficiently. It includes an execution engine, a compiler, and a set of libraries in it. It is a set of computer software and specifications.

The exact meaning of platform means it’s bride between two pilliors. When comeback to java in technical language...

When run java program it will generate the .class file and that .calsss file will work on the any operating systems(Ex: windows, Linux , unix etc) without any problems. .Class file with is the bridge between hard and software to work and execute the problem. So the .class file generation can be done on java only, so that why people used to call java as platform:

Java Platform is a collection of programs that help programmers to develop and run Java programming applications efficiently. It includes an execution engine, a compiler, and a set of libraries in it. It is a set of computer software and specifications.

The exact meaning of platform means it’s bride between two pilliors. When comeback to java in technical language...

When run java program it will generate the .class file and that .calsss file will work on the any operating systems(Ex: windows, Linux , unix etc) without any problems. .Class file with is the bridge between hard and software to work and execute the problem. So the .class file generation can be done on java only, so that why people used to call java as plat

When run java program it will generate the .class file and that .calsss file will work on the any operating systems(Ex: windows, Linux , unix etc) without any problems. .Class file with is the bridge between hard and software to work and execute the problem. So the .class file generation can be done on java only, so that why people used to call java as platform

1. **Is Java Pure-Object oriented Language ?**

Pure Object Oriented Language or Complete Object Oriented Language are Fully Object Oriented Language which supports or have features which treats everything inside program as objects. It doesn’t support primitive datatype(like int, char, float, bool, etc.). There are seven qualities to be satisfied for a programming language to be pure Object Oriented. They are:

Encapsulation/Data Hiding

Inheritance

Polymorphism

Abstraction

All predefined types are objects

All user defined types are objects

All operations performed on objects must be only through methods exposed at the objects.

Example: Smalltalk

Why Java is not a Pure Object Oriented Language?

Java supports property 1, 2, 3, 4 and 6 but fails to support property 5 and 7 given above. Java language is not a Pure Object Oriented Language as it contain these properties:

Primitive Data Type ex. int, long, bool, float, char, etc as Objects: Smalltalk is a “pure” object-oriented programming language unlike Java and C++ as there is no difference between values which are objects and values which are primitive types. In Smalltalk, primitive values such as integers, booleans and characters are also objects.

In Java, we have predefined types as non-objects (primitive types).

nt a = 5;

System.out.print(a);

The static keyword: When we declares a class as static then it can be used without the use of an object in Java. If we are using static function or static variable then we can’t call that function or variable by using dot(.) or class object defying object oriented feature.

Wrapper Class: Wrapper class provides the mechanism to convert primitive into object and object into primitive. In Java, you can use Integer, Float etc. instead of int, float etc. We can communicate with objects without calling their methods. ex. using arithmetic operators.

String s1 = "ABC" + "A" ;

Even using Wrapper classes does not make Java a pure OOP language, as internally it will use the operations like Unboxing and Autoboxing. So if you create instead of int Integer and do any mathematical operation on it, under the hoods Java is going to use primitive type int only.

public class BoxingExample

{

public static void main(String[] args)

{

Integer i = new Integer(10);

Integer j = new Integer(20);

Integer k = new Integer(i.intValue() + j.intValue());

System.out.println("Output: "+ k);

}

}

In the above code, there are 2 problems where Java fails to work as pure OOP:While creating Integer class you are using primitive type “int” i.e. numbers 10, 20.While doing addition Java is using primitive type “int”.

1. **Which version of java have u learned? Name some of the new features added to it.**

We learned 1.8 version of java

Follwing are some new features added by java 1.8

1) Parallel Operations:- Multiple threads can work at a same time

2) Functional Interface & Lambda Expressions: - An interface with exactly one abstract method becomes Functional Interface. Adding functional capabilities.

3) Method referencing: Now we can refer methods by their name

4) Data time API :- more improved and enhanced version data API is available

5) Default method − Interface to have default method implementation.

6) New tools and compilers are added like "jdeps"

7) JAVASCRIPT engine: new javascript engine named Nashorn is added to execute javascript codes

8) Stream API :- are introduces to run parallel and pipelined processes.

**18.WhatgivesJavaits'writeonceandrunanywhere'nature?**

Oneoftheinitial"killerfeature"ofJavawassupposedtobethewriteonce, runanywherenatureofit. Earlier, itisnotpracticallypossibletohavedifferentversionsofanapplicationfordifferentdevicesbecausethedeviceshavevarietyofCPUs, operatingsystemandbrowsers. Thesamecodemustworkonallthecomputers, thereforeweneedaportablecode . Portabilityreferstotheabilitytorunaprogramondifferentmachines. "Javaisportable" , meansthatyoucanrunJavabytecodeonanyhardwarethathasacompliantJVM (JavaVirtualMachine).

TheJavaCompilercompilesajavaprogram (.javafile) andconvertsitintoclassfiles (.class) thatcontainbytecodes , whichistheintermediatelanguagebetweensourcecodeandmachinecode . Thesebytecodesarenotplatformspecific, sowiththehelpofJVM (Javavirtualmachine), thejavaprogramcanrunonwidevarietyofplatforms. TheJVM (Javavirtualmachine) isplatformdependenti.eitsimplementationdiffersfromplatformtoplatform (likewindows, linuxatc.), buttheseallJVMscanexecutethesamejavabytecode .

Thisissomethingwhichcanbetermedas'writeonceandrunanywhere' . So, thismeansJavacanbedevelopedonanydevice, compiledintoastandardbytecodeandbeexpectedtorunonanydeviceequippedwithaJVM (Javavirtualmachine).

1. **Difference between path and classpath.**

PATH:-

1) It is environment variable used to locate JDK libraries like "java" or "javac" command use to run or compile code.

2) To set path we need to include JDK\_HOME/bin file.

3) It cannot be overridden by any JAVA settings.

4) PATH environment variable is used by operating system to find any binary or command typed in the shell,

CLASSPATH:-

1) It is an environment variable is used by System to locate and load compile Java byte codes stored in the .class file.

2) To set CLASSPATH we need to include all directories containing all JAR files and .class files which are required by our JAVA application.

3) It can be overridden by providing command line option -classpath or -cp to both "java" and "javac" commands.

4) It is used by class loaders to load JAVA files.

1. **What is the signature of main function in java ?**

The signature of the main method is public static void main(String[] ags). public static void main(String a[]) is the main entry point signature for a typical Java program. So you should get on with this method signature. Java Runtime tries to find a method with name "main" with argument types "String[]".

**21. What is Difference Between JDK and JRE?**

Development tools e.g.javac,java,etc.

Set of Libraries e.g.rt.jar etc

JVM

Other files

JRE

JDK

JRE : Java Runtime Environment

- It is also written as Java RTE

- set of software tools which are used for developing Java applications

- used to provide the runtime environment.

- It is the implementation of JVM

- It physically exists

- JRE is platform dependent.

- It contains a set of libraries + other files that JVM uses at runtime.

- The implementation of JVM is also actively released by other

companies besides Sun Micro Systems.

JDK: Java Development Kit

- software development environment which is used to develop Java

applications and applets.

- It physically exists

- JDK is platform dependent.

- It contains JRE + development tools

- contains a private Java Virtual Machine (JVM) and a few other resources

such as an interpreter/loader (java), a compiler (javac), an archiver (jar),

a documentation generator (Javadoc), etc.

to complete the development of a Java Application.

- JDK is an implementation of any one of the below given Java

Platforms released by Oracle Corporation:

1.Standard Edition Java Platform

2.Enterprise Edition Java Platform

3.Micro Edition Java Platform

**22. What is JVM? What it does?**

* JVM - Java Virtual Machine

- an abstract machine

- JVM is platform dependent.

It is a specification that provides runtime environment in which java bytecode can be executed.

It is **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.

It is **An implementation** Its implementation is known as JRE (Java Runtime Environment).

It is **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

* JVM performs operation like,

Loads code

Verifies code

Executes code

Provides runtime environment

* JVM provides definitions for the:

Memory area

Class file format

Register set

Garbage-collected heap

Fatal error reporting etc.

**23. Why JVM is called as “Virtual Machine”?**

**Ans:**

JVM is called virtual because it is not a physical device and it does not physically exists.

When compile java program it creates class file which is called as byte code. JVM reads this byte code and converts this byte code to underlaying OS understandable code.

There is no real hardware which interprets the byte code.

**24. What are main components of JVM? Explain them OR Explain**

**JVM Architecture.**

JVM Lang class

JVM language Stacks

Native method Stacks

PC registers

Heap

Method area

Native Method libraries

Native Method Interface

Executive Engine

Fig: **JVM Architecture**

Here are the important components of JVM:

1) Class Loader

The class loader is a subsystem used for loading class files. It performs three major functions viz. Loading, Linking, and Initialization.

2) Method Area

JVM Method Area stores structure of class like metadata, the code for Java methods, and the constant runtime pool.

3) Heap

All the Objects, arrays, and instance variables are stored in a heap. This memory is shared across multiple threads.

4) JVM language Stacks

Java language Stacks store local variables, and its partial results. Each and every thread has its own JVM language stack, created concurrently as the thread is created. A new frame is created when method is invoked, and it is removed when method invocation process is complete.

5) PC Registers

PC registers store the address of the Java virtual machine instruction, which is currently executing. In Java, each thread has its separate PC register.

6) Native Method Stacks

Native method stacks hold the instruction of native code depends on the native library. It allocates memory on native heaps or uses any type of stack.

7) Execution Engine

It is a type of software that is used to test software, hardware, or complete systems. The test execution engine never carries any information about the tested product.

8) Native Method interface

The Native Method Interface is a programming framework. It allows Java code, which is running in a JVM to call by libraries and native applications.

9) Native Method Libraries

Native Libraries is a collection of the Native Libraries (C, C++), which are needed by the Execution Engine.

1. **What is difference between java compiler javac and jit?**

Java Compiler (javac):

- Java compiler refers to a program which translates

Java

- language source code into the Java Virtual Machine

(JVM) bytecodes.

- javac is the tool which can transform code to the Java

bytecode.

- The javac command is included in the Java

Development Kit (JDK) distributions.

- The Java compiler and the rest of the standard Java

toolchain places the following restrictions on the code:

Source code is held in files with the suffix ".java"

Bytecodes are held in files with the suffix ".class"

JIT (Just-In-Time Interpreter):

- JIT stands for Java-In-Time Compiler.

- The JIT compilation is also known as dynamic compilation.

- JIT is a integral part of the JVM that optimizes the performance of the application.

- JIT is the tool which can transform bytecode to the binary

code.

- It accelerates execution performance many times over the

previous level.

- It is a long-running, computer-intensive program that provides

the best performance environment.

- It optimizes the performance of the Java application at compile or run time.

- The term Java interpreter refers to a program which

implements the JVM specification and actually executes the

bytecodes.

**26. Is Empty .java file name a valid source file name?**

Yes empty .java file name works but class must not be public ,it means that it must be default.

E.g when we create a one file with class x then save it as .java and while compile it just compile as javac .java

When it compile JVM run. You can execute it has java x.here x is class name.

Let’s take a simple example:

//save by .java only  
class A{  
public static void main(String args[]){  
System.out.println(“Hello java”);  
}  
}

1. Compile by ---- --🡪javac .java  
2.Run by -------🡪java A  
3.Compile it by------🡪 javac .java

4. Run it by --------🡪java A

1. **Is JRE different for different Platforms ?**

JRE is an acronym for Java Runtime Environment. It is used to provide runtime environment it provides JVM +libraries (which are needed by JVM), and other components to run applets and applications written in Java.

JRE compiles java file i.e. .java file and converts it into .class file i.e. byte code.

Complied class file of java file will be same no matter if the java was complied on windows/Linux/Mac from algorithmic perspective yes complier is same across the platform.

But java compiler /JRE installed on one as will not work on other OS will not work on other OS.that is why oracle gives option like “Jdk for windows, Jdk foe Linux/Mac while downloading.

**28. Difference between C++ and java in terms of object creation.**

|  |  |
| --- | --- |
| C++ | Java |
| C++ allows you to create on object anywhere /every where on the heap ,stack etc. | Java force you allocates objects on the heaps always; |
| In c++ only object created on the stack are released automatically | The JVM execute that java code log all objects being created so that memory allocated foe them can later be freed automatically by garbage collection when objects are not referenced any more |

**29. Who invokes main () function?**

Ans:-

The java tool launches a java application .it does this by starting a java runtime environment, loading a specified class and invoking that class’s main method. The methods declaration is like that:

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

JVM uses the main () as a starting point for program.

JVM calls the main method because it is the entry point to the class that has to be loaded in order to execute the class.

**30. What is .class file known as?**

Ans:-

A java class file is a file(with .class filename extension)containing java byte code that can be executed on java virtual machine(JVM).A java class file is usually produced by a java complier from java programming language source file.

.class file is called ad byte code in java.

**31. Can we define more than one public class in a java source code? what is the rule of public class and file name?**

No, while defining multiple classes in a single Java file you need to make sure that only one class among them is public. If you have more than one public class a single file a compile-time error will be generated.

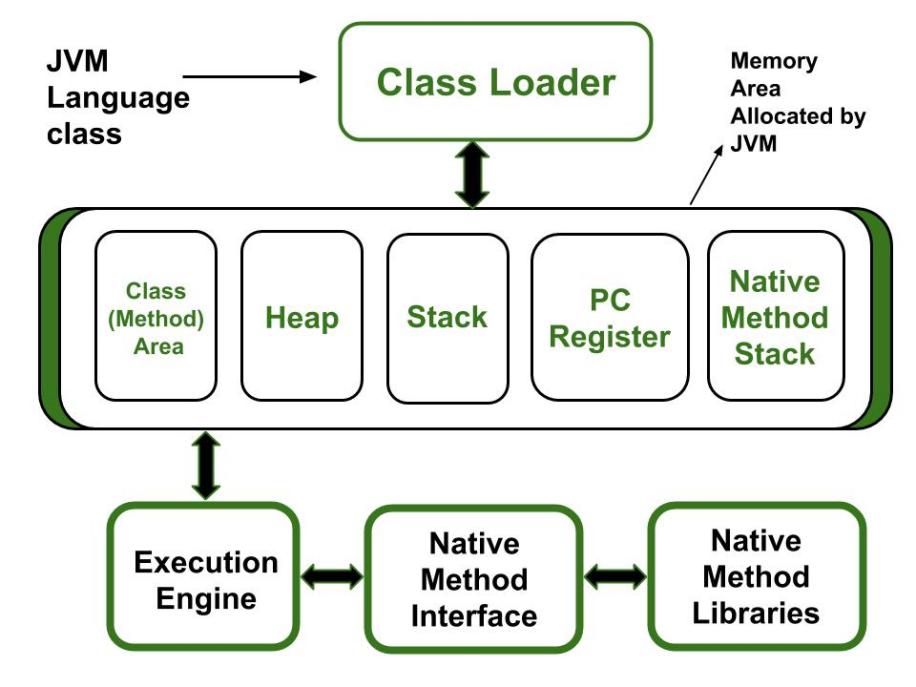
Name the file with the class name that contains main method.

1. **What is JIT compiler?**

**JIT** stands for **Java-In-Time Compiler**. **JIT** in Java is an integral part of the **JVM**. JIT compilers interact with the Java Virtual Machine (JVM) at run time and compile suitable bytecode sequences into native machine code and improve performance. It accelerates execution performance.

**33.How many types of memory areas are allocated by JVM?**

The memory in java devided into 5 areas.



1. **Class (Method) Area:** It stores class level data of every class such as the runtime constant pool, field and method data, the code for methods.
2. **Heap:** It is used to allocate memory to objects at run time
3. **Stack:**
   * Each thread has a private JVM stack, created at the same time as thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.
   * Java Stack stores frames and a new frame is created each time at every invocation of the method.  
     A frame is destroyed when its method invocation completes
4. **Program Counter Register:** Each JVM thread which carries out the task of a specific method has a program counter register associated with it. The non-native method has a PC which stores the address of the available JVM instruction whereas, in a native method, the value of the program counter is undefined. PC register is capable of storing the return address or a native pointer on some specific platform.
5. **Native method Stacks:** Also called as C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when its created and it can be of a fixed or dynamic nature.

**34.What is the rule for local member in java?**

* Local variables cannot use any of the access levels as they exist inside the Method only.
* The Final is the only nonaccess modifier that can be applied to a local variable.
* Local variables don’t get a default value, hence local variables need to be initiated before they can be used.

1. **What are the various access specifiers in Java?**

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

**1.Public:** The classes, methods, or variables which are defined as public, can be accessed by any class or method.

**2.Protected:** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.

**3.Default:** Default is accessible within the package only. By default, all the classes, methods, and variables are of default scope.

**4.Private:** The private class, methods, or variables defined as private can be accessed within the class only

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

1. **What is the rule for local member in java ?**

Ans-

:-Local variables cannot use any of the [access level](https://javabeginnerstutorial.com/core-java-tutorial/access-modifier-in-java/) since their scope is only inside the method.

**:-Final** is the Only [Non Access Modifier](https://javabeginnerstutorial.com/core-java-tutorial/non-access-modifiers-in-java/) that can be applied to a local variable.

:-Local variables are not assigned a default value, hence they need to be initialized.

1. **What is native code?**

Native code is computer programming (code) that is compiled to run with a particular [processor](https://whatis.techtarget.com/definition/processor) (such as an Intel [x86](https://searchwindowsserver.techtarget.com/definition/x86)-class processor, AMD processor) and its set of [instruction](https://whatis.techtarget.com/definition/instruction)s.Native code can also be distinguished from [bytecode](https://whatis.techtarget.com/definition/bytecode) (sometimes called interpreted code), a form of code that can be said to run in a [virtual machine](https://searchservervirtualization.techtarget.com/definition/virtual-machine) (for example, the [Java Virtual Machine](https://www.theserverside.com/definition/Java-virtual-machine-JVM)). The virtual machine is a program that converts the platform-generalized bytecode into the native code that will run in a specific processor.

1. **Why there is no sizeof operator in java ?**

:-Because the [size of primitive types](http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html) is explicitly mandated by the Java language. There is no variance between JVM implementations.

:-Moreover, since allocation is done by the new operator depending on its argument there is no need to specify the amount of memory needed.

:-It would sure be convenient sometimes to know how much memory an object will take so you could estimate things like max heap size requirements but I suppose the Java Language/Platform designers did not think it was a critical aspect.

1. **What kinds of programs u can develop using Java ?**

:- Application servers.

:-Web applications.

:-Unit tests.

:-Mobile applications.

:-Desktop applications.

:-Enterprise applications.

:-Scientific applications.

:- Business applications.

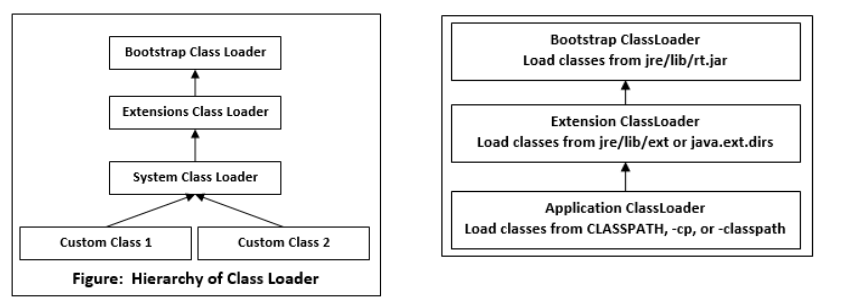
1. **U have reference type as a member of class. What is the default value it gets?**

The default value of a reference type variable is null when they are not initialized. Null means not refering to any object.

1. What is the job done by classloader ?

java ClassLoader is an abstract class. It belongs to a java.lang package.  
It loads classes from different resources. Java ClassLoader is used to load the  
classes at run time. In other words, JVM performs the linking process at  
runtime. Classes are loaded into the JVM according to need.  
If a loaded class depends on another class, that class is loaded as well. When  
we request to load a class, it delegates the class to its parent. In this way,  
uniqueness is maintained in the runtime environment. It is essential to execute  
a Java program.

42.Explain the hierarchy of classloaders in java.



Bootstrap Class Loader: It loads standard JDK class files from rt.jar and other  
core classes. It is a parent of all class loaders. It doesn't have any parent. When  
we call String.class.getClassLoader() it returns null, and any code based on it  
throws NullPointerException. It is also called Primordial ClassLoader. It loads  
class files from jre/lib/rt.jar. For example, java.lang package class.  
Extensions Class Loader: It delegates class loading request to its parent. If the  
loading of a class is unsuccessful, it loads classes from jre/lib/ext directory or  
any other directory as java.ext.dirs. It is implemented by  
sun.misc.Launcher$ExtClassLoader in JVM

System Class Loader: It loads application specific classes from the  
CLASSPATH environment variable. It can be set while invoking program using -  
cp or classpath command line options. It is a child of Extension ClassLoader. It  
is implemented by sun.misc.Launcher$AppClassLoader class. All Java  
ClassLoader implements java.lang.ClassLoader.

1. **What is the role played by Bytecode Verifier ?**

The bytecode verifier acts as a sort of gatekeeper: it ensures that code  
passed to the Java interpreter is in a fit state to be executed and can run  
without fear of breaking the Java interpreter. Imported code is not allowed to  
execute by any means until after it has passed the verifier's tests.

**44.What are the memory areas allocated by JVM ?**

JVM has five memory locations namely -  
**Heap** - Runtime storage allocation for objects (reference types).  
**Stack** - Storage for local variables and partial results. A stack contains  
frames and allocates one for each thread. Once a thread gets completed,  
this frame also gets destroyed. It also plays roles in method invocation and  
returns.  
**PC Registers** - Program Counter Registers contains the address of an  
instruction that JVM is currently executing.  
**Execution Engine** - It has a virtual processor, interpreter to interpret  
bytecode instructions one by one and a JIT, just in time compiler.  
**Native method stacks** - It contains all the native methods used by the  
application.

**45.What kinds of programs u can develop using Java?**

:- Application servers.

:-Web applications.

:-Unit tests.

:-Mobile applications.

:-Desktop applications.

:-Enterprise applications.

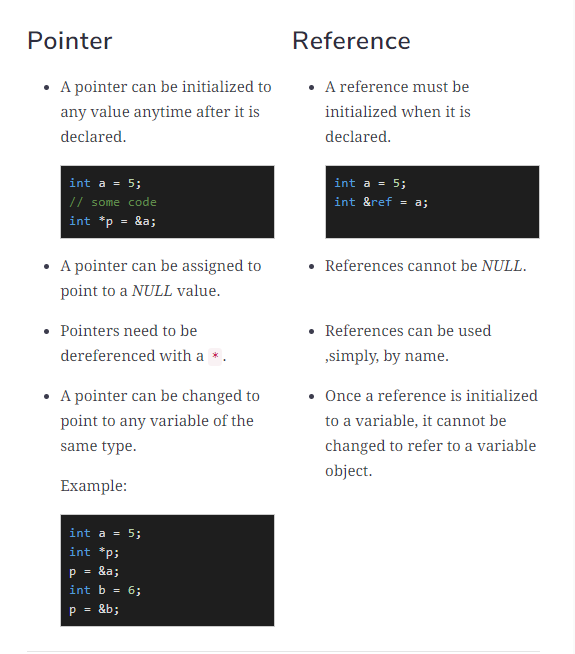
1. **When parseInt() method can be used?**

While operating upon strings, there are times when we need to convert a number represented as a string into an integer type. The method generally used to convert String to Integer in Java is **parseInt ()**. This method belongs to integer class in java.lang package .It takes a valid string as a parameter and parses it into primitive data type int. It only accepts String as a parameter and on passing values of any other data type

**47. What is finalized () method?**

* finalize () is called by the garbage collector on an object when garbage collection determines that there are no more references to the object. A subclass overrides the finalize method to dispose of system resources or to perform other cleanup.
* The Finalize method is used to perform cleanup operations on unmanaged resources held by the current object before the object is destroyed. The method is protected and therefore is accessible only through this class or through a derived class.

**48. Difference between C++ pointer and Java reference.**



**49. U have reference type as a member of class. What is the default value it gets?**

Null

The default value of a reference type variable is null when they are not initialized. Null means not refering to any object.

**50. What are the expressions allowed in switch block of java?**

A switch works with the byte, short, char, and int primitive data types, the String class and a few special classes that wrap certain primitive types:

Character, Byte, Short, and Integer (discussed in Numbers and Strings).