**Q1.  What is Java bytecode ?**

Ans. “Java bytecode” is the usual name for the machine language of the Java Virtual Machine. Java programs are compiled into Java bytecode, which can then be executed by the JVM.

**Q2.  Why is Java considered Portable Language ?**

Ans. Java is a portable-language because without any modification we can use Java byte-code in any platform(which supports Java). So this byte-code is portable and we can use in any other major platforms.

**Q3.  How Java provide high Performance ?**

Ans. Java uses Just-In-Time compiler to enable high performance. Just-In-Time compiler is a program that turns Java bytecode into instructions that can be sent directly to the processor.

**Q4.  What is Byte Code ? Why Java's intermediary Code is called Byte Code ?**

Ans. Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system. Its called Byte Code because each instruction is of 1-2 bytes.

Sample instructions in Byte Code -

1: istore\_1

2: iload\_1

3: sipush 1000

6: if\_icmpge 44

9: iconst\_2

10: istore\_2

**Q5. Which memory segment holds the Byte Code ?**

Ans. Code Segment.

**Q6.  Is JVM an overhead ?**

Ans. Yes and No. JVM is an extra layer that translates Byte Code into Machine Code. So Comparing to languages like C, Java provides an additional layer of translating the Source Code.

C++ Compiler - Source Code --> Machine Code

Java Compiler - Source Code --> Byte Code   ,  JVM - Byte Code --> Machine Code

Though it looks like an overhead but this additional translation allows Java to run Apps on all platforms as JVM provides the translation to the Machine code as per the underlying Operating System.

1) Define Constructor?

2) How many types of Constructors are in Java?

There are two types of constructors in Java:

1. **Default Constructor (Non-parameterized Constructor)**
2. **Parameterized Constructor**

3) Do we have a copy constructor in Java?

Unlike C++, there is no explicit copy constructor in Java. However, we can achieve the functionality of a copy constructor in Java by copying the values from one object to another, just like the copy constructor.

The following are some methods to copy the values from one object to another:

* By constructor
* By assigning the values of one object to another
* By clone() method of Object class

### 5) Is there any method to call a sub-class constructor from a superclass constructor?

The subclass constructor has its own private data members, so Java does not provide any way to access the sub-class constructor from a super class constructor. However, we can call a superclass constructor from a sub-class constructor by using the super keyword.

### 6) Can we have a constructor in the Interface?

No, we cannot have constructors in the Java interface.

### 7) Explain Constructor Chaining?

Constructor Chaining is a way to call one constructor from another constructor with respect to the current object. It can be achieved in the following two ways:

**From base class:** We can use the super keyword to call a constructor from the base class.

**Within the same class:** We can call a constructor within the same class by using **this()** keyword.

### 8) What happens if we provide a return type to a constructor?

If we provide a return type to a constructor, it will function as a general method. But, the compiler will display a warning message, "**This method has a Constructor name**".

9) What is a private constructor?

Like methods, we can have the private constructors in Java. To make or create a constructor as private, use the **private** keyword while declaring it. It can only be accessed within that class.

The following are some usage scenarios when we need a private constructor:

* Internal Constructor chaining
* Singleton class design pattern

10) Why constructors in Java cannot be static?

The constructors cannot be static in Java. When we declare a method as static, it means the method belongs to the class and not to a specific object. But the constructor is always invoked to the reference of objects. So, there is no sense in making a constructor static.

11) Can we make a constructor final?

No, we cannot make a constructor final. If we made a constructor final, it would throw a compile-time error "**modifier final not allowed**".

12) Can we make a constructor abstract?

a body, which really makes no sense. It is automatically called at the time of object creation. So, it cannot be a block without a body.

13) what will happen when a constructor is declared as protected?

Generally, when we declare a method as protected, other classes can access that method in a different package by using inheritance only. But, when we declare a constructor protected, it behaves slightly differently than a method. The protected constructor can only be accessed by using a super keyword according to Java language standards.

14) Why constructor name is similar to the class name?

When we create an object of a class using a new keyword, it should have information about that particular class. That is why the constructor's name must be similar to the class name.

15) Why return type is not allowed for the constructor?

The return type is not allowed in the constructor because if we provide a return type in the constructor, it will act as the normal method. So, to differentiate between constructor and method block, the return type is not allowed in constructors.

**What is No-arg constructor?**  
Constructor without arguments is called no-arg constructor. Default constructor in java is always a no-arg constructor.

**How a no – argument constructor is different from**[default Constructor](https://www.geeksforgeeks.org/g-fact-50/)**?**  
If a class contains no constructor declarations, then a default constructor with no formal parameters and no throws clause is implicitly declared.

If the class being declared is the primordial class Object, then the default constructor has an empty body. Otherwise, the default constructor simply invokes the superclass constructor with no arguments.

**What are**[private constructors](https://www.geeksforgeeks.org/private-constructors-and-singleton-classes-in-java/)**and where are they used?**  
Like any method we can provide access specifier to the constructor. If it’s made private, then it can only be accessed inside the class.  
The major scenarios where we use private constructor:

* + Internal Constructor chaining
  + Singleton class design pattern

**When do we need**[Constructor Overloading](https://www.geeksforgeeks.org/constructor-overloading-java/)**?**  
Sometimes there is a need of initializing an object in different ways. This can be done using constructor overloading. Different constructors can do different work by implementing different line of codes and are called based on the type and no of parameters passed.  
According to the situation , a constructor is called with specific number of parameters among overloaded constructors.

**Do we have destructors in Java?**  
No, Because Java is a garbage collected language you cannot predict when (or even if) an object will be destroyed. Hence there is no direct equivalent of a destructor.

**1. What is Inheritance in Java?**

Ans: The technique of creating a new class by using an existing class functionality is called inheritance in Java. In other words, inheritance is a process where a child class acquires all the properties and behaviors of the parent class.

**2. Why do we need to use inheritance?**  
Or, What is the purpose of using inheritance?

Ans: Inheritance is one of the main pillars of OOPs concept. Some objects share certain properties and behaviors. By using inheritance, a child class acquires all properties and behaviors of parent class.

There are the following reasons to use inheritance in java.

* We can reuse the code from the base class.
* Using inheritance, we can increase features of class or method by overriding.
* Inheritance is used to use the existing features of class.
* It is used to achieve runtime polymorphism i.e method overriding.

**3. What is Is-A relationship in Java?**

Ans: Is-A relationship represents Inheritance. It is implemented using the “extends” keyword. It is used for code reusability.

**4. What is super class and subclass?**

Ans: A class from where a subclass inherits features is called superclass. It is also called base class or parent class.

A class that inherits all the members (fields, method, and nested classes) from other class is called subclass. It is also called a derived class, child class, or extended class.

**5. How is Inheritance implemented/achieved in Java?**

Ans: Inheritance can be implemented or achieved by using two keywords:

1. **extends:** extends is a keyword that is used for developing the inheritance between two classes and two interfaces.
2. **implements:** implements keyword is used for developing the inheritance between a class and interface.

**6. Write the syntax for creating the subclass of a class?**

Ans: A subclass can be created by using the “extends” keyword. The syntax for declaring a subclass of class is as follows:

class subclassName extends superclassName

{

// Variables of subclass

// Methods of subclass

}

where class and extends are two keywords.

**7. Which class in Java is superclass of every other class?**

Ans: In Java, Object class is the superclass of every other class.

**9. Can a class extend itself?**

Ans: No, a class cannot extend itself

**11. Can a class extend more than one class?**

Ans: No, one class can extend only a single class.

**12. Are constructor and instance initialization block inherited to subclass?**

Ans: No, constructor and instance initialization block of the superclass cannot be inherited to its subclass but they are executed while creating an object of the subclass.

**13. Are static members inherited to subclass in Java?**

Ans: Static block cannot be inherited to its subclass.

A static method of superclass is inherited to the subclass as a static member and non-static method is inherited as a non-static member only.

**14. Can we extend (inherit) final class?**

Ans: No, a class declared with final keyword cannot be inherited.

**15. Can a final method be overridden?**

Ans: No, a final method cannot be overridden.

**16. Can we inherit private members of base class to its subclass?**

Ans: No.

**17. What is order of calling constructors in case of inheritance?**

**Ans: In case of inheritance, constructors are called from the top to down hierarchy.**

**18. Which keyword do you use to define a subclass?**  
Or, which keyword is used to inherit a class?

Ans: extends keyword.

**19. What are the advantages of inheritance in Java?**

Ans: The advantages of inheritance in java are as follows:

* We can minimize the length of duplicate code in an application by putting the common code in the superclass and sharing it amongst several subclasses.
* Due to reducing the length of code, the redundancy of the application is also reduced.
* Inheritance can also make application code more flexible to change.

**20. What are the types of inheritance in Java?**

Ans: The various types of inheritance are as follows:

a. Single inheritance  
b. Multi-level inheritance  
c. Hierarchical inheritance  
d. Multiple inheritance  
e. Hybrid inheritance

**27. How will you restrict a member of a class from inheriting its subclass?**

Ans: We can restrict members of a class by declaring them private because the private members of superclass are not available to the subclass directly. They are only available in their own class.

**28. Can we access subclass members if we create an object of superclass?**

Ans: No, we can access only superclass members but not the subclass members.

**29. Can we access both superclass and subclass members if we create an object of subclass?**

Ans: Yes, we can access both superclass and subclass members.

**30. What happens if both superclass and subclass have a field with the same name?**

Ans: Only subclass members are accessible if an object of subclass is instantiated.

**43. How many ways to implement relationships among classes in Java?**

Ans: There are four ways to make relationships among classes. They are as follows:

a. Association  
b. Aggregation  
c. Composition  
d. Inheritance

**44. What is Association in OOPs concept?**

Ans: Association in Java is one of the core concepts of OOPs. It establishes the relation between two classes that are independent of one another.

In association, all objects have their own life cycle and there is no ownership between objects. Association can be unidirectional or bidirectional.

**45. In the OOPs concept, what is meant by Aggregation?**

Ans: Aggregation in java represents has-a relationship. It is a special form of association that represents an ownership relation between two objects.

In aggregation, two aggregated objects have their own life cycle but one of the objects is the owner of has-a relationship.

**46. What is meant by Composition in OOPs?**

Ans: Composition in java is one of the core concepts of OOPs and a more restrictive case of aggregation. It represents has-a relationship that contains an object and cannot exist on its own.

In simple words, if a class object holds an object of another class, it is called composition. It establishes strong relationship between two objects than aggregation.

**47. Which is more tightly bound? Inheritance or Composition?**

Ans: Composition is more tightly bound than inheritance.

**48. What is the difference between Inheritance and Composition?**

Ans: Inheritance represents Is-A relationship between subclass and its superclass whereas, composition represents Has-A relationship between classes.

**49. When will you prefer composition over inheritance?**

Ans: Inheritance can be used only when there is a perfect IS-A relationship between the superclass and subclass definitions. But in case of any confusion, prefer composition over inheritance.

**50. How aggregation and composition both are different concepts?**

Ans: In the OOPs concept, aggregation and composition both are types of association relations. A composition establishes a strong relationship between classes.

If the composite object is destroyed, all its parts are destroyed. For example, a car has a steering wheel. If car object is destroyed, there is no meaning to be the existence of steering wheel.

Aggregation establishes a weaker relationship between classes than composition. For example, a library has students. If a library is destroyed, students still exist. Hence, library and student are related by aggregation.

**Array**

Exceptions

1. **ArrayStoreException**
2. **ArrayIndexOutOfBoundsException**
3. NegativeArraySizeException
4. **What is ArrayStoreException in java? When you will get this exception?**
   1. ArrayStoreException is a run time exception which occurs when you try to store non-compatible element in an array object. The type of the elements must be compatible with the type of array object. For example, you can store only string elements in an array of strings. if you try to insert integer element in an array of strings, you will get ArrayStoreException at run time.

**2) Can you pass the negative number as an array size?**

No. You can’t pass the negative integer as an array size. If you pass, there will be no compile time error but you will get NegativeArraySizeException at run time.

**3) Can you change the size of the array once you define it? OR Can you insert or delete the elements after creating an array?**

No. You can’t change the size of the array once you define it. You can not insert or delete the elements after creating an array. Only you can do is change the value of the elements.

**4) What is an anonymous array? Give example?**

Anonymous array is an array without reference. For example,

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | **public** **class** MainClass  {  **public** **static** **void** main(String[] args)      {          //Creating anonymous arrays            System.out.println(**new** **int**[]{1, 2, 3, 4, 5}.length);    //Output : 5            System.out.println(**new** **int**[]{21, 14, 65, 24, 21}[1]);   //Output : 14      }  } |

**6) There are two array objects of int type. one is containing 100 elements and another one is containing 10 elements. Can you assign array of 100 elements to an array of 10 elements?**

Yes, you can assign array of 100 elements to an array of 10 elements provided they should be of same type. While assigning, compiler checks only type of the array not the size.

**9) What are the different ways of copying an array into another array?**

There are four methods available in java to copy an array.

1) Using for loop

2) Using Arrays.copyOf() method // **int**[] b = Arrays.copyOf(a, a.length);

3) Using System.arraycopy() method // **int**[] b = a.clone();

4) Using clone() method // System.arraycopy(a, 0, b, 0, a.length);

**11) How do you check the equality of two arrays in java? OR How do you compare the two arrays in java?**

You can use Arrays.equals() method to compare one dimensional arrays and to compare multidimensional arrays, use Arrays.deepEquals() method.

**12) What is ArrayIndexOutOfBoundsException in java? When it occurs?**

ArrayIndexOutOfBoundsException is a run time exception which occurs when your program tries to access invalid index of an array i.e negative index or index higher than the size of the array.

**13) How do you sort the array elements?**

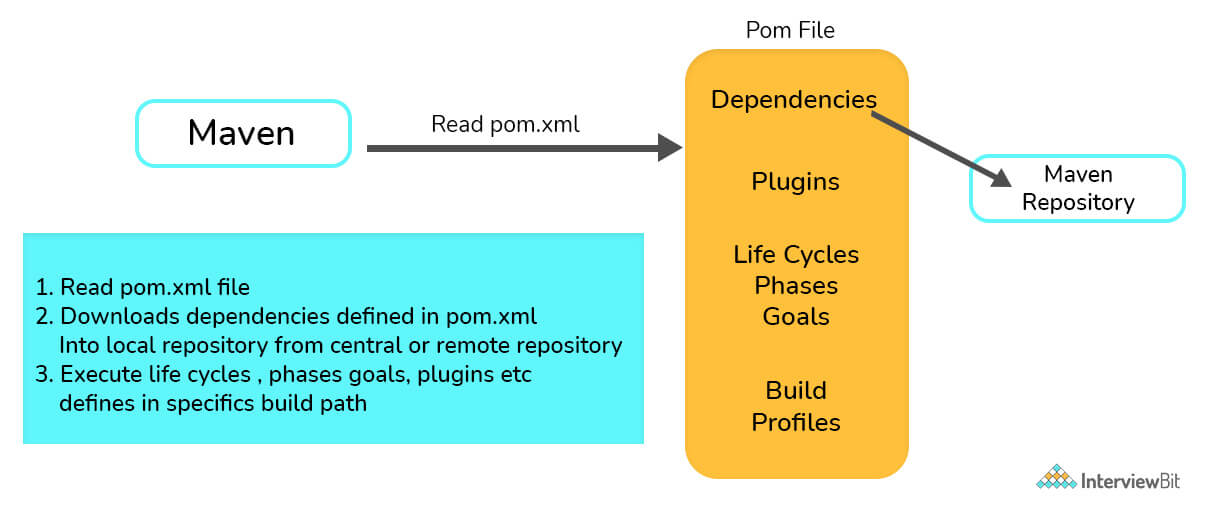
You can sort the array elements using Arrays.sort() method. This method internally uses quick sort algorithm to sort the array elements.

**1. When should one use Maven?**

The Maven Build Tool can be used in the following conditions:

* When the project has a large number of dependencies. Then, using Maven, you can easily manage those dependencies.
* When the version of a dependency changes frequently. To update dependencies, simply update the version ID in the pom file.
* Maven makes it simple to handle continuous builds, integration, and testing.
* When you need a quick way to generate documentation from source code, this is the tool you use. It helps in compiling source code, and then packaging it into JAR or ZIP files.

**3. How does Maven work?**



Maven works in three steps:

* Reading the pom.xml file is the first step.
* The dependencies mentioned in pom.xml are then downloaded from the central repository into the local repository.
* Finally, it builds and generates a report based on the requirements, as well as handles life cycles, phases, goals, plugins, and other tasks.

**5. What elements are used for creating a pom.xml file?**

The following elements are necessary for creating a pom.xml file:

* **project**- The root element of the pom.xml file is the project.
* **modelVersion**- It identifies which version of the POM model you're working with. For Maven 2 and Maven 3, use version 4.0.0.
* **groupId**- groupId is the project group's identifier. It is unique, and you will most likely use a group ID that is similar to the project's root Java package name.
* **artifactId**- It is used for naming the project you're working on.
* **version**- The version number of the project is contained in the version element. If your project has been released in multiple versions, it is helpful to list the versions.

Other Pom.xml File Elements

* **dependencies**- This element is used to establish a project's dependency list.
* **dependency**- dependency is used inside the dependencies tag to define a dependency. The groupId, artifactId, and version of each dependency are listed.
* **name**- This element is used to give our Maven project a name.
* **scope**- This element is used to specify the scope of this maven project, which can include compile, runtime, test, among other things.
* **packaging**- The packaging element is used to package our project into a JAR, WAR, and other output formats.

**6. What are the different types of Maven repositories? Discuss.**

The three types of repositories of Maven are:

* Local repository
* Central repository
* Remote repository

Maven scans these repositories for dependencies. Maven looks in the Local repository first, then the Central repository, and finally the Remote repository if the Remote repository is defined in the POM.

* **Local Repository**: Local repository is a directory on the developer's device. The local repository contains all of Maven's dependencies. Even though several projects rely on dependencies, Maven only needs to download them once.
* **Central Repository**: The Maven community has built the central Maven repository. Maven searches this central repository for any dependencies that aren't available in your local repository. The dependencies are subsequently downloaded into your local repository by Maven.
* **Remote Repository**: Maven may download dependencies from a remote repository hosted on a web server. It is frequently used to host internal organization projects. The dependencies are subsequently downloaded into your local repository by Maven.
* **Q #5) What is a build tool responsible for?**
* **Answer**: Build tools are primarily responsible for producing the source code (provided the auto-generated code is utilized). It creates project documentation from the source code.
* It compiles and packages the code in the form of JAR/ZIP file. Finally, places that code in the local, central, or remote repository.
* **Q #6) What do you mean by POM and what does it contain?**
* **Answer:**Project Object Model or pom forms the elementary part of the working of Maven. While performing a Maven goal or a task, the pom file residing in the present directory is searched and referred to collect information on the project.
* It is basically an xml file. It contains configuration details like plugins, goals, developers, dependencies, profiles, versions, and mailing lists.

**Q #16)** **What are Maven Plugins?**

**Answer:**Maven plugins are the basic component of a Maven framework. Each of the plugins has a specific task to be performed.

**Maven generally performs the following functionalities:**

* Generates jar files.
* Generates war files.
* Compiles the code.
* Executes unit testing of code.
* Generates documentation of the project.
* Generates customized reports.

**Maven plugin gives a group of goals that can be run with the following command syntax:**

**mvn [plugin-name]:[goal-name]**

**Q #17) What are the different types of Maven Plugins?**

**Answer:**

**The different types of Maven plugins are listed below:**

* **Building Plugins:**These plugins are used at the time of build and are defined in the building element of the pom file.
* **Reporting Plugins:**These plugins are used at the time of site generation and are defined in the reporting element of the pom file.
* **Q #25) What do you mean by Maven’s External Dependencies?**
* **Answer:**Once Maven reads through the pom file, it gets to know the list of dependencies for the project. It searches for these dependencies in Local, Central and Remote repositories. In case any of the dependency not found in any of the repositories, then Maven utilizes the external dependency.