JAVA INTERVIEW QUESTIONS

1. JDK - Java Development Kit

* It is the tool necessary to compile, document, and package Java programs.
* It contains JRE + development tools.

1. JRE - Java Runtime Environment

* JRE refers to a runtime environment in which Java bytecode can be executed.
* It’s an implementation of the JVM which physically exists.

1. JVM – Java Virtual Machine

* It is an abstract machine. It is a specification that provides a run-time environment in which Java bytecode can be executed.
* JVM follows three notations: Specification, Implementation, and Runtime Instance.

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

main() in Java is the entry point for any Java program. It is always written as public static void main(String[] args).

* public: Public is an access modifier used to specify who can access this method. Public means that this Method will be accessible by any Class.
* static: It is a keyword in java that identifies it as class-based. main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, the main is not made static then the compiler will throw an error as main() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.
* void: It is the return type of the method. Void defines the method which will not return any value.
* main: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
* String args[]: It is the parameter passed to the main method.

1. Why Java is platform-independent?

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

1. Why Java is not 100% Object-oriented?

Java is not 100% Object-oriented because it makes use of eight primitive data types such as boolean, byte, char, int, float, double, long, and short which are not objects.

1. What are wrapper classes in Java?

Wrapper classes convert the Java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they “wrap” the primitive data type into an object of that class. Refer to the below image which displays the different primitive types, wrapper classes, and constructor arguments.

1. What are constructors in Java?

In Java, a constructor refers to a block of code that is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and it is automatically called when an object is created.

There are two types of constructors:

* Default Constructor: In Java, a default constructor is the one that does not take any inputs. Its main purpose is to initialize the instance variables with the default values. Also, it is majorly used for object creation.
* Parameterized Constructor: The parameterized constructor in Java, is the constructor which is capable of initializing the instance variables with the provided values. In other words, the constructors which take the arguments are called parameterized constructors.

1. What is a singleton class in Java and how can we make a class singleton?

Singleton class is a class whose only one instance can be created at any given time, in one JVM. A class can be made singleton by making its constructor private.

1. What is the difference between equals() and == in Java?

* Equals() method is defined in Object class in Java and used for checking the equality of two objects defined by business logic.
* “==” or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects.

1. When can you use the super keyword?

In Java, the super keyword is a reference variable that refers to an immediate parent class object.

The uses of the Java super Keyword are-

* To refer to an immediate parent class instance variable, use super.
* The keyword super can be used to call the method of an immediate parent class.
* Super() can be used to call the constructor of the immediate parent class.

1. What is the importance of reflection in Java?

Reflection is a runtime API for inspecting and changing the behavior of methods, classes, and interfaces. Java Reflection is a powerful tool that can be beneficial. Java Reflection allows you to analyze classes, interfaces, fields, and methods during runtime without knowing what they are called at compile time. Reflection can also be used to create new objects, call methods, and get/set field values. External, user-defined classes can be used by creating instances of extensibility objects with their fully qualified names. Debuggers can also use reflection to examine private members of classes.

1. How do not allow serialization of attributes of a class in Java?

The NonSerialized attribute can be used to prevent member variables from being serialized.

You should also make an object that potentially contains security-sensitive data non-serializable if possible. Apply the NonSerialized attribute to certain fields that store sensitive data if the object must be serialized. If you don’t exclude these fields from serialization, the data they store will be visible to any programs with serialization permission.

1. Can you call a constructor of a class inside another constructor?

Yes, we can call a constructor of a class inside another constructor. This is also called constructor chaining. Constructor chaining can be done in 2 ways-

* Within the same class: For constructors in the same class, this() keyword can be used.
* From the base class: The super() keyword is used to call the constructor from the base class.

1. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.

An array generally contains elements of the primitive data types such as int, float, etc. In such cases, the array directly stores these elements at contiguous memory locations. While an ArrayList does not contain primitive data types. An ArrayList contains the reference of the objects at different memory locations instead of the object itself. That is why the objects are not stored at contiguous memory locations.

1. Why is it said that the length() method of the String class doesn’t return accurate results?

The length() method of the String class doesn’t return accurate results because

it simply takes into account the number of characters within the String. In other words, code points outside of the BMP (Basic Multilingual Plane), that is, code points having a value of U+10000 or above, will be ignored.

The reason for this is historical. One of Java’s original goals was to consider all text as Unicode; yet, Unicode did not define code points outside of the BMP at the time. It was too late to modify char by the time Unicode specified such code points.

1. Stack memory

* Stack memory is used only by one thread of execution.
* Stack memory can’t be accessed by other threads.
* Follows LIFO manner to free memory.
* Exists until the end of execution of the thread.
* Stack memory only contains local primitive and reference variables to objects in heap space.

1. Heap memory

* Heap memory is used by all the parts of the application.
* Objects stored in the heap are globally accessible.
* Memory management is based on the generation associated with each object.
* Heap memory lives from the start till the end of application execution.
* Whenever an object is created, it’s always stored in the Heap space.

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

Packages in Java are the collection of related classes and interfaces which are bundled together. By using packages, developers can easily modularize the code and optimize its reuse. Also, the code within the packages can be imported by other classes and reused.

* Packages help in avoiding name clashes
* They provide easier access control on the code
* Packages can also contain hidden classes which are not visible to the outer classes and only used within the package
* Creates a proper hierarchical structure which makes it easier to locate the related classes

1. What is the JIT compiler in Java?

Just-In-Time compiler is a program that helps in converting the Java bytecode into instructions that are sent directly to the processor. By default, the JIT compiler is enabled in Java and is activated whenever a Java method is invoked. The JIT compiler then compiles the bytecode of the invoked method into native machine code, compiling it “just in time” to execute. Once the method has been compiled, the JVM summons the compiled code of that method directly rather than interpreting it. This is why it is often responsible for the performance optimization of Java applications at the run time.

1. Local variable

In Java, a local variable is typically used inside a method, constructor, or block and has only local scope. Thus, this variable can be used only within the scope of a block. The best benefit of having a local variable is that other methods in the class won’t be even aware of that variable.

1. Instance variable

An instance variable in Java is a variable that is bounded to its object itself. These variables are declared within a class, but outside a method. Every object of that class will create its copy of the variable while using it. Thus, any changes made to the variable won’t reflect in any other instances of that class and will be bound to that particular instance only.

1. What is the final keyword in Java?

Final is a special keyword in Java that is used as a non-access modifier.

1. Break

* Can be used in switch and loop (for, while, do-while) statements
* It causes the switch or loop statements to terminate the moment it is executed
* It terminates the innermost enclosing loop or switches immediately

1. Continue

* Can be only used with loop statements
* It doesn’t terminate the loop but causes the loop to jump to the next iteration
* A continue within a loop nested with a switch will cause the next loop iteration to execute

1. What is an infinite loop in Java?

An infinite loop is an instruction sequence in Java that loops endlessly when a functional exit isn’t met. This type of loop can be the result of a programming error or may also be a deliberate action based on the application behavior. An infinite loop will terminate automatically once the application exits.

1. What is Java String Pool?

Java String pool refers to a collection of Strings that are stored in heap memory. In this, whenever a new object is created, the String pool first checks whether the object is already present in the pool or not. If it is present, then the same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned.

1. Static

Is mainly used for memory management. Called without the class/ object instance. It uses early binding, the memory is fixed in the ram.

1. Non-static

Is everything by default. Called using an instance of class/ object. Uses late binding, the memory is not fixed in the ram.

1. Explain the term “Double Brace Initialisation” in Java?

Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialisation. The inner class has a reference to the enclosing outer class, generally using the ‘this’ pointer. It is used to do both creation and initialization in a single statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

1. What is constructor chaining in Java?

In Java, constructor chaining is the process of calling one constructor from another concerning the current object. Constructor chaining is possible only through legacy where a subclass constructor is responsible for invoking the superclass constructor first. There could be any number of classes in the constructor chain. Constructor chaining can be achieved in two ways:

* Within the same class using this()
* From base class using super()

1. Why Java Strings are immutable?

In Java, string objects are immutable which simply means once the String object is created its state cannot be modified. Whenever you try to update the value of that object instead of updating the values of that particular object, Java creates a new string object. Java String objects are immutable as String objects are generally cached in the String pool. Since String literals are usually shared between multiple clients, action from one client might affect the rest. It enhances the security, caching, synchronization, and performance of the application.

1. What is the collection class in Java? List down its methods and interfaces.

In Java, the collection is a framework that acts as an architecture for storing and manipulating a group of objects. Using Collections you can perform various tasks like searching, sorting, insertion, manipulation, deletion, etc. Java collection framework includes the following:

* Interfaces
* Classes
* Methods

1. What is an association?

Association is a relationship where all object have their lifecycle and there is no owner.

Let’s take the example of Teacher and Student. Multiple students can associate with a single teacher and a single student can associate with multiple teachers but there is no ownership between the objects and both have their lifecycle. These relationships can be one to one, one to many, many to one, and many to many.

1. What do you mean by aggregation?

An aggregation is a specialized form of Association where all object has their lifecycle but there is ownership and a child object can not belong to another parent object.

Let’s take the example of Department and teacher. A single teacher can not belong to multiple departments, but if we delete the department teacher’s object will not destroy.

1. What is a composition in Java?

A composition is again a specialized form of Aggregation and we can call this a “death” relationship. It is a strong type of Aggregation. Child objects do not have their lifecycle and if the parent object deletes all child objects will also be deleted. Let’s take again an example of a relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different houses if we delete the house room will automatically delete.

1. What is a marker interface?

In simpler terms, an empty interface is called the Marker interface. The most common examples of Marker interfaces in Java are Serializable, Cloneable.

1. What is object cloning in Java?

Object cloning in Java is the process of creating an exact copy of an object. It means the ability to create an object with a similar state as the original object. To achieve this, Java provides a method clone() to make use of this functionality. This method creates a new instance of the class of the current object and then initializes all its fields with the same contents as the corresponding fields. To object clone(), the marker interface java.lang.Cloneable must be implemented to avoid any runtime exceptions. One thing you must note is Object clone() is a protected method, thus you need to override it.

1. What is a servlet?

* Java Servlet is server-side technology to extend the capability of web servers by providing support for dynamic response and data persistence.
* The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing our own servlets.
* All servlets must implement the javax.servlet.Servlet interface, which defines servlet lifecycle methods. When implementing a generic service, we can extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet() and doPost(), for handling HTTP-specific services.
* Most of the time, web applications are accessed using the HTTP protocol, and that’s why we mostly extend HttpServlet class. The Servlet API hierarchy is shown below image.

1. What is a Request Dispatcher?

RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP, or another servlet in the same application. We can also use this to include the content of another resource in the response.

There are two methods defined in this interface:

* void forward()
* void include()

1. What is the life cycle of a servlet?

* Servlet is loaded
* Servlet is instantiated
* Servlet is initialized
* Service the request
* Servlet is destroyed

1. How do cookies work in Servlets?

* Cookies are text data sent by the server to the client and it gets saved at the client’s local machine.
* Servlet API provides cookies support through javax.servlet.http.Cookie class that implements Serializable and Cloneable interfaces.
* HttpServletRequest getCookies() method is provided to get the array of Cookies from the request, since there is no point in adding a Cookie to a request, there are no methods to set or add a cookie to a request.
* Similarly HttpServletResponse addCookie(Cookie c) method is provided to attach cookies in the response header, there are no getter methods for cookies.

1. What are the different methods of session management in servlets?

A session is a conversational state between client and server and it can consist of multiple requests and responses between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session-id) is passed between server and client in every request and response.

Some of the common ways of session management in servlets are:

* User Authentication
* HTML Hidden Field
* Cookies
* URL Rewriting
* Session Management API

1. POJO – Plain Old Java Object is an ordinary Java object that does not follow any of the major Java object models, conventions, or frameworks.
2. API – Application Programming Interface is software, that gives the possibility for 2 applications to communicate.
3. SOLID:

* Single responsibility – 1 method = 1 functionality
* Open – closed – open for extensions, closed for modifications
* Liskov substitution – subclass should be able to be replaced by the superclass
* Interface segregation – big interface made into smaller ones
* Dependency inversion – abstraction over concretization

1. Compile time – the time when the source code is converted to executable code.
2. Runtime – the time when the executable code is started running.