Top 1000 Java Interview Question & Answers

Knowledge
Powerhouse

www.KnowledgePowerhouse.com

DEDICATIO

N

To our readers!

CONTENTS

Java Basics

1. What is the difference between JDK and JRE?

The **Java** Development Kit (**JDK**) is a software development environment used for developing **Java** applications and applets. It includes the **Java** Runtime Environment (JRE), an interpreter/loader (**java**), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in **Java** development.

The **Java** Runtime Environment (**JRE**) is a set of software tools for development of **Java** applications. It combines the **Java** Virtual Machine (JVM), platform core classes and supporting libraries. **JRE** is part of the **Java** Development Kit

2. What is Java Virtual Machine (JVM)?

JVM is a engine that provides runtime environment to drive the **Java** Code or applications. It converts **Java** bytecode into machines language. **JVM** is

a part of JRE(**Java** Run Environment). It stands for **Java** Virtual Machine.

3. What are the different types of memory areas allocated by JVM?

Heap Area,Stack Area,Method Area

PC Register,Native Area

4. What is a JIT compiler?

Just-In-Time

The Just-In-Time (JIT) compiler is a an essential part of the JRE i.e. Java Runtime Environment, that is responsible for performance optimization of java based applications at run time. Optimizations performed by JIT compilers are data-analysis, reduction of memory accesses by register allocation, translation from stack operations to register operations, elimination of common sub-expressions etc

5. How Java platform different from other platforms?

Follow WORA principle ,Write Once and Run Anywhere

Robust,Object Oriented

6. Why people say that Java is 'write once and run anywhere' language?

It is platform independent where we write once and run everywhere

7. How does ClassLoader work in Java?

In jvm it search about memory allocation of class if not alloted then it

goes to System/Application class loader then Extension class loader then BootStrap class loader until not found then it throw classNotFoundException.

8. Do you think ‘main’ used for main method is a keyword in Java?

NOT keyword, its methods

9. Can we write main method as public void static instead of public static void?

Compile time error

10. In Java, if we do not specify any value for local variables, then what will be the default value of the local variables?

Compile time error if value is not initialized

11. Let say, we run a java class without passing any arguments. What will be the value of String array of arguments in Main method?

Length of String Array is zero.

12. What is the difference between byte and char data types in Java? OOPS

Byte = signed data type,Represent negative value,-128 to 127 range

Signed data type= Signed data types can hold positive and negative numbers

Char=unsigned data type,represent positive value,0 to 65535

range

Unsigned data type= Unsigned can hold larger positive values, but no negative values.
Unsigned uses the leading bit as a part of the value, while the signed version uses the leftmost bit to identify whether the number is positive or negative. Alternatively, two's complement can be used to designate a number as positive or negative.

13. What are the main principles of Object Oriented Programming?

1.Object

2.Class

3.Encapsulation

4.Abstraction

5.Polymorphism

6.Inheritance

14. What is the difference between Object Oriented Programming language and Object Based Programming language?

1.oop = support inheritance and polymorphism,ex: java

2.obp = not support inheritance and polymorphism,ex:javascript

15. In Java what is the default value of an object reference defined as an instance variable in an Object?

null

16. Why do we need constructor in Java?

To initialize the value of object or object creation.

17. Why do we need default constructor in Java classes?

For the creation of class object.

18. What is the value returned by Constructor in Java?

No return type.

19. Can we inherit a Constructor?

No

inherits all the members (fields, methods, and nested classes) from its superclass not constructor.

20. Why constructors cannot be final, static, or abstract in Java?

Abstract = used with class and method not with constructor.

When you set a method as abstract, then “The method doesn’t or cannot have body”. A constructor will be automatically called when object is created. It cannot lack a body moreover an abstract constructor could never be implemented.

Final = cannot be overridden in sub class.

When you set a method as final, then" The method cannot be overridden by any class", but **Constructor** by JLS (**Java Language Specification**) definition can't be overridden. A constructor is not inherited, so there is no need for declaring it as **final**.

Static = method cannot be inherited in the sub class because they belong to the class in which they have been declared.

When you set a method as static, it means "The Method belong to class and not to any particular object" but a constructor is always invoked with respect to an object, so it makes no sense for a constructor to be **static**.

21. What is the purpose of 'this' keyword in java?

Current class object.

Inheritance

22. Explain the concept of Inheritance?

To inherit Parent class features in child class.

23. Which class in Java is superclass of every other class?

Object class

24. Why Java does not support multiple inheritance?

Ambigiuty problem

A show() method<-C B show() method<-C

No decision taken by compiler to call show method of class.

25. In OOPS, what is meant by composition?

Means One object always depends on another object.One object cannot exist without other object.Always a ownership.Strong Relationship

Ex: human and heart

26. How aggregation and composition are different concepts?

Aggregation :-One object exists without another object,no ownership,weak relationship

Ex: Teacher and department

Teacher can exists without department object.

Composition : - refer above question

27. Why there are no pointers in Java?

1. **Memory allocation** is done automaticaly it is managed by Jvm. So to avoid direct access to memory by user pointers dont allowed in java
2. Java Language does not use pointer for the reason that it works on **Internet**. Applets are used on the internet. and Pointers are used to

identify the address of variables,methods, and therefore even big programmer can find out the secrets of other user on the internet using pointer. so if there could be pointer in java, it could be harmful for leakage of the important information

28. If there are no pointers in Java, then why do we get NullPointerException?

In java, pointers is reference of heap memory where object gets space.

Yes, but only to objects. In fact, the only way to access an object is through a pointer because Java objects are always allocated on the heap. **Pointers in Java are called *references*** (not to be confused with C++ references, which have different semantics)

29. What is the purpose of ‘super’ keyword in java?

To refer parent class object.

30. Is it possible to use this() and super() both in same constructor?

no

31. What is the meaning of object cloning in Java?

To create exact copy of object.

1.Shallow cloning : reflect changes on original object.

2.Deep cloning: no reflection

Static

32. In Java, why do we use static variable?

1.static variable shares common memory.

2.we directly call the object using class name.

3.To share static variable value over the application.

33. Why it is not a good practice to create static variables in Java?

OOPs concept violates

34. What is the purpose of static method in Java?

when we no need to be invoked **method** using instance

35. Why do we mark main method as static in Java?

it allows main() to be called without having to instantiate a particular instance of that class.

36. In what scenario do we use a static block?

For mandatory execution of code.

37. Is it possible to execute a program without defining a main() method?

From version java 7 +,not possible throw exception.

38. What happens when static modifier is not mentioned in the signature of main method?

Throw runtime exception,no main method is found.

39. What is the difference between static method and instance method in Java?

Static method : called by using class name,can overload static method but can’t override it.

Method Overloading and Overriding

40. What is the other name of Method Overloading?

Static binding,early binding

41. How will you implement method overloading in Java?

Same method name with different number of parameter or different datatype.

42. What kinds of argument variations are allowed in Method Overloading?

1. different number of parameter

2. different data type.

43. Why it is not possible to do method overloading by changing return type of method in java?

Scenario : same no. of arguments with same datatype different

return type.

Ambiguity problem

44. Is it allowed to overload main() method in Java?

yes

45. How do we implement method overriding in Java?

Same method name with same number of parameter and same data type.

46. Are we allowed to override a static method in Java?

no

47. Why Java does not allow overriding a static method?

Method Hiding

48. Is it allowed to override an overloaded method?

yes

49. What is the difference between method overloading and method overriding in Java?

50. Does Java allow virtual functions?

Abstract class

51. What is meant by covariant return type in Java?

In overriding ,Overridden method have different return type of class name.

Polymorphism

52. What is Runtime Polymorphism?

Method Overriding

53. Is it possible to achieve Runtime Polymorphism by data members in Java?

no

54. Explain the difference between static and dynamic binding?

Static binding = Overloading

Dynamic binding = Overriding

Abstraction

55. What is Abstraction in Object Oriented programming?

To show features by hiding internal implementation

Abstraction achieved by abstract class and interface.

56. How is Abstraction different from Encapsulation?

Abstraction hides details at the **design** level, while Encapsulation hides details at the **implementation** level.

57. What is an abstract class in Java?

58. Is it allowed to mark a method abstract method without marking the class abstract?

No,it is mandatory to define abstract keyword when method is abstract.

59. Is it allowed to mark a method abstract as well as final?

No,both are opposite in nature.

Abstract use to override the class but final restrict overriding.

60. Can we instantiate an abstract class in Java?

No,

61. What is an interface in Java?

It is a concept to achieve

abstraction.

62. Is it allowed to mark an interface method as static?

no

63. Why an Interface cannot be marked as final in Java?

Interface is for implementation of method but final keyword restrict the inherit.

64. What is a marker interface?

Interface(tag interface) with no field or methods ex: serialization,cloneable

Consider Cloneable Marker Interface. If the operation is not supported by an object, it can throw an exception when such operation is attempted, like Collection.remove does when the collection does not support the remove operation (eg, unmodifiable collection) but a class claiming to implement Cloneable and throwing CloneNotSupportedException from the clone() method wouldn't be a very friendly thing to do.

65. What can we use instead of Marker interface?

marker interface in Java is used to indicate something to compiler, JVM or any other tool but **Annotation** is better way of doing same thing.

66. How Annotations are better than Marker Interfaces?

https://beetechnical.com/tech-tutorial/marker-interface

s-vs-annotations-in-java-quick-comparison/

Unlike annotations, interfaces allow us to **take advantage of polymorphism**. As a result, we can **add additional restrictions to the marker interface.**

67. What is the difference between abstract class and interface in Java?

68. Does Java allow us to use private and protected modifiers for variables in interfaces?

no

69. How can we cast to an object reference to an interface reference?

```
interface MyInterface{

   public static int num = 100;

   public void display();

}

public class InterfaceExample implements MyInterface{

   public void display() {

      System.out.println("This is the implementation of the display
method");

   }

   public void show() {

      System.out.println("This is the implementation of the show
method");

   }
```

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

        MyInterface obj = new InterfaceExample();

        obj.display();

        //obj.show();

    }

}
```

you need to cast an object reference to an interface reference. Whenever you need to call the methods of the interface only.

Final

70. How can you change the value of a final variable in Java?

No

71. Can a class be marked final in Java?

Yes,ex : String,Wrapper Class

72. How can we create a final method in Java?

By using final keyword with method.

73. How can we prohibit inheritance in Java?

By using final Keyword with method

74. Why Integer class in final in Java?

To make it immutable.

75. What is a blank final variable in Java?

Final variable whose value is not initialize.It initialize through constructor.

76. How can we initialize a blank final variable?

Through constructor

Static final variable initialize through static block.

77. Is it allowed to declare main method as final?

yes

Package

78. What is the purpose of package in Java?

Collection of classes.

79. What is java.lang package?

The package java.lang contains classes and interfaces that are essential to the Java language. These include:

- Object, the ultimate superclass of all classes in Java

- Thread, the class that controls each thread in a multithreaded program
- Throwable, the superclass of all error and exception classes in Java
- Classes that encapsulate the primitive data types in Java
- Classes for accessing system resources and other low-level entities
- Math, a class that provides standard mathematical methods
- String, the class that is used to represent strings

80. Which is the most important class in Java?

Object

81. Is it mandatory to import java.lang package every time?

no

82. Can you import same package or class twice in your class?

yes, Neither compiler nor JVM complains about it.

83. What is a static import in Java?

Which import static class

84. What is the difference between import static com.test.Fooclass and import com.test.Fooclass?

- To access a class or method from another package we need to use the **fully qualified name** or we can use **import** statements.
- The class or method should also be accessible. Accessibility is based on the **access modifiers**.
- **Private** members are accessible only within the same class. So we won't be able to access a private member even with the fully qualified name or an import statement.
- The **java.lang** package is automatically imported into our code by Java.

Static import

- **Static imports** will import all static data so that can use **without a class name**.
- A **static import** declaration has two forms, one that imports a particular static member which is known as **single static impor**t and one that imports all **static members of a class** which is known as a **static import on demand**.
- Static imports introduced in **Java5 version**.
- One of the advantages of using static imports is **reducing keystrokes and re-usability**.

Internationalization

85. What is Locale in Java?

represents a specific geographic, cultural, or political region.

86. How will you use a specific Locale in Java?

```
1. Locale locale1 = new Locale("EN", "INDIA");
```

```
2.
3.      // print locale
4.      System.out.println("Locale: " + locale1);
5.
6.      // print language
7.      System.out.println("Language: " + locale1.getDisplayLanguage());
8.
9.      // print country
10.     System.out.println("Country Name: "
11.                        + locale1.getDisplayCountry());
```

Locale: en_INDIA

Language: English

Country Name: INDIA

Serialization

87. What is the serialization?

To convert object into byte.

88. What is the purpose of serialization?

Serialization is a mechanism of converting the state of an object into a byte stream.

1. If a parent class has implemented Serializable interface then child class doesn't need to implement it but vice-versa is not true.

2. Only non-static data members are saved via Serialization process.

3. Static data members and transient data members are not saved via Serialization process.So, if you don’t want to save value of a non-static data member then make it transient.

4. Constructor of object is never called when an object is deserialized.

5. Associated objects must be implementing Serializable interface.

Why static variable not serialize ?

Because static variable is class level variable and serialization about persist state of object if static variable change the value then is also change serial version

89. What is Deserialization?

Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory.

90. What is Serialization and Deserialization conceptually?

91. Why do we mark a data member transient?

To make data not persist or not save.

92. Is it allowed to mark a method as transient?

No,it is keyword used with variable

93. How does marking a field as transient makes it possible to serialize an object?

Transient doesn’t use to turn off the serialization it only means to mark the object not to save.

https://stackoverflow.com/questions/5177013/how-does-marking-a-field-as-transient-make-it-possible-to-serialise-an-object

In this example, if class is not serialized then it better to define transient rather than to perform serialization.

This example makes Loan class not persistent by which foo became serializable.

94. What is Externalizable interface in Java?

Externalization serves the purpose of custom Serialization, where we can decide what to store in stream.

Externalizable interface present in java.io, is used for Externalization which extends Serializable interface. It consist of two methods which we have to override to write/read object into/from stream type :readExternal() and writeExternal().

95. What is the difference between Serializable and Externalizable interface?

`Serializable` is a marker interface i.e. does not contain any method.	`Externalizable` interface contains two methods `writeExternal()` and `readExternal()` which implementing classes MUST override.

`Serializable` interface pass the responsibility of serialization to JVM and it's default algorithm.	`Externalizable` provides control of serialization logic to programmer – to write custom logic.
Mostly, default serialization is easy to implement, but has higher performance cost.	Serialization done using `Externalizable`, add more responsibility to programmer but often result in better performance.
It's hard to analyze and modify class structure because any change may break the serialization.	It's more easy to analyze and modify class structure because of complete control over serialization logic.
Default serialization does not call any class constructor.	A public no-arg constructor is required while using `Externalizable` interface.

Reflection

96. What is Reflection in Java?

Reflection is an API which is used to examine or modify the behavior of methods, classes, interfaces at runtime.

- The required classes for reflection are provided under java.lang.reflect package.
- Reflection gives us information about the class to which an object belongs and also the methods of that class which can be executed by using the object.
- Through reflection we can invoke methods at runtime irrespective of the access specifier used with them.

97. What are the uses of Reflection in Java?

to describe code which is able to inspect other code in the same system

98. How can we access private method of a class from outside the class?

```
class Test

{

    // creating a private field

    private String s;
```

```
    // creating a public constructor
    public Test()  {  s = "GeeksforGeeks"; }

    // Creating a public method with no arguments
    public void method1()  {
        System.out.println("The string is " + s);
    }

    // Creating a public method with int as
argument
    public void method2(int n)  {
        System.out.println("The number is " + n);
    }

    // creating a private method
    private void method3() {
        System.out.println("Private method
invoked");
    }
```

```
}


class Demo

{

    public static void main(String args[]) throws
Exception

    {

        // Creating object whose property is to be
checked

        Test obj = new Test();



        // Creating class object from the object
using

        // getclass method

        Class cls = obj.getClass();

        System.out.println("The name of class is "
+

                           cls.getName());



        // Getting the constructor of the class
```

```
through the

        // object of the class

        Constructor constructor =
cls.getConstructor();

        System.out.println("The name of
constructor is " +


constructor.getName());



        System.out.println("The public methods of
class are : ");



        // Getting methods of the class through
the object

        // of the class by using getMethods

        Method[] methods = cls.getMethods();



        // Printing method names

        for (Method method:methods)

            System.out.println(method.getName());
```

```
        // creates object of desired method by
providing the

        // method name and parameter class as
arguments to

        // the getDeclaredMethod

        Method methodcall1 =
cls.getDeclaredMethod("method2",


int.class);



        // invokes the method at runtime

        methodcall1.invoke(obj, 19);



        // creates object of the desired field by
providing

        // the name of field as argument to the

        // getDeclaredField method

        Field field = cls.getDeclaredField("s");



        // allows the object to access the field
irrespective
```

```
        // of the access specifier used with the
field

        field.setAccessible(true);



        // takes object and the new value to be
assigned

        // to the field as arguments

        field.set(obj, "JAVA");



        // Creates object of desired method by
providing the

        // method name as argument to the
getDeclaredMethod

        Method methodcall2 =
cls.getDeclaredMethod("method1");



        // invokes the method at runtime

        methodcall2.invoke(obj);



        // Creates object of the desired method by
providing
```

```
        // the name of method as argument to the

        // getDeclaredMethod method

        Method methodcall3 =
cls.getDeclaredMethod("method3");



        // allows the object to access the method
irrespective

        // of the access specifier used with the
method

        methodcall3.setAccessible(true);



        // invokes the method at runtime

        methodcall3.invoke(obj);
```

```
    }

}
```

99. How can we create an Object dynamically at Runtime in Java?

Using Reflection

Class.forName(),class loader

If we know the name of the class & if it has a public default constructor we can create

an object in this way.

MyObject object = (MyObject) Class.forName("subin.rnd.MyObject").newInstance();

Using clone()

The clone() can be used to create a copy of an existing object.

MyObject anotherObject = new MyObject();

MyObject object = (MyObject) anotherObject.clone();

Using object deserialization

Object deserialization is nothing but creating an object from its serialized form.

ObjectInputStream inStream = new ObjectInputStream(anInputStream);

MyObject object = (MyObject) inStream.readObject();

Garbage Collection

100.What is Garbage Collection in Java?

Process to remove unused object from memory.

101. Why Java provides Garbage Collector?

For cleanup processing,Found in System and Runtime classes.

102.What is the purpose of gc() in Java?

used to invoke the garbage collector to perform cleanup processing.

103.How does Garbage Collection work in Java?

> Garbage collection is performed by a daemon thread called Garbage Collector(GC). This thread calls the finalize() method before object is garbage

collected.

104.When does an object become eligible for Garbage Collection in Java?

- Any instances that cannot be reached by a live thread.
- Circularly referenced instances that cannot be reached by any other instances.
- If an object has only lived weak references via WeakHashMap it will be eligible for garbage collection.
- The object is created inside a block and reference goes out scope once control exit that block.

105.Why do we use finalize() method in Java?

To free the memory from those objects which are not created with new keyword.

Present in Object class.

Note:-The Garbage collector of JVM collects only those objects that are created by new keyword. So if you have created any object without new, you can use finalize method to perform cleanup processing (destroying remaining objects).

106.What are the different types of References in Java?

- SoftReference object: Garbage collector is required to clear all SoftReference objects when memory runs low.
- WeakReference object: When garbage collector senses a weakly referenced object, all references to it are cleared

and ultimately taken out of memory.

- PhantomReference object: Garbage collector would not be able to automatically clean up PhantomReference objects, you would need to clean it up manually by clearing all references to it.

107.How can we reference an unreferenced object again?

Assign null to object,

assign object to other object and

by anonymous object.

108.What kind of process is the Garbage collector thread?

Garbage collector is deamon thread which free the variable from memory which are not created with new keyword.

109.What is the purpose of the Runtime class?

Clean up processing

gc() found in System and Runtime class.

110. How can we invoke an external process in Java?

https://www.tutorialspoint.com/how-to-execute-an-external-program-like-windows-media-player-in-java

111. What are the uses of Runtime class?

Inner Classes

112. What is a Nested class?

113. How many types of Nested classes are in Java?

Nested classes are divided into two categories:

1. **static nested class : Nested classes that are declared *static* are called static nested classes.**
2. **inner class : An inner class is a non-static nested class.**

S.NO	NORMAL/REGULAR INNER CLASS	STATIC NESTED CLASS
1.	**Without an outer class object existing, there cannot be an inner class object. That is, the inner class object is always associated**	**Without an outer class object existing, there may be a static nested class object. That is, static nested class object is not**

	with the outer class object.	**associated with the outer class object.**
2.	**Inside normal/regular inner class, static members can't be declared.**	**Inside static nested class, static members can be declared.**
3.	**As main() method can't be declared, regular inner class can't be invoked directly from the command prompt.**	**As main() method can be declared, the static nested class can be invoked directly from the command prompt.**
4.	**Both static and non static members of outer class can be accessed directly.**	**Only a static member of outer class can be accessed directly.**

114. Why do we use Nested Classes?

Inner classes are a security mechanism in Java. We know a class cannot be associated with the access modifier private, but if we have the class as a member of other class, then the inner class can be made private. And this is also used to access the private members of a class.

115. What is the difference between a Nested class and an

Inner class in Java?

116. What is a Nested interface?

117. How can we access the non-final local variable, inside a Local Inner class?

```
Class A {

private int x =10;

Public void sample method(){

final int y =20;

Class B {

Public void accessMainClassVar(){

South(x);

 South(y);

}

}

A a = new B();

a.accessMainClassVar();

}


Class Test {
```

Psvm(){

A a = new A();

a.sample method();

}

}

}

118. Can an Interface be defined in a Class?

Yes

```
class MyClass{

    interface MyInterfaceB{

       void myMethod();

    }

}
```

```
class NestedInterfaceDemo2 implements
MyClass.MyInterfaceB{

     public void myMethod(){

         System.out.println("Nested interface
method");

     }
```

```
    public static void main(String args[]){
      MyClass.MyInterfaceB obj=
                    new NestedInterfaceDemo2();
          obj.myMethod();
    }
}
```

119. Do we have to explicitly mark a Nested Interface public static?

Nested interfaces are static by default. **You** don't **have** to **mark** them **static explicitly** as it **would** be redundant. **Nested interfaces** declared inside class **can** take any access modifier, however **nested interface** declared inside **interface** is **public** implicitly.

120.Why do we use Static Nested interface in Java?

to resolve the namespace by grouping related interfaces
(or related interface and class) together.
1. A static nested class can directly access the static members of its outer class including private static members.
2. It acts as a direct member of its top-level class, not a member of the package in which it is declared.
3. Without an existing outer class object, there may be a chance of an existing static nested class. To access static members of the static nested class, there is no need for an outer class object.

String

121. What is the meaning of Immutable in the context of String class in Java?

Cannot be changed,it can be changed by passing reference only.

122.Why a String object is considered immutable in java?

For security purpose , if String is not immutable then changes in one place may affect String object reference by same object.

123.How many objects does following code create?

124.How many ways are there in Java to create a String object?

By new keyword and using String literals.

125.How many objects does following code create?

126.What is String interning?

To pass the heap memory object reference to constant pool area.

127.Why Java uses String literal concept?

For memory efficiency.So that same String object value refers same object.

128.What is the basic difference between a String and

StringBuffer object?

String : immutable

String buffer : mutable,Thread Safe.

129.How will you create an immutable class in Java?

the class as final

all fields private so that direct access is not allowed

Don’t provide setter methods for variables

mutable fields final
initialize all the fields via a constructor performing deep copy.

Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference.

130.What is the use of toString() method in java ?

Return string representation of object.

131. Arrange the three classes String, StringBuffer and StringBuilder in the order of efficiency for String processing operations?

Exception Handling

132.What is Exception Handling in Java?

133.In Java, what are the differences between a Checked and Unchecked?

134.What is the base class for Error and Exception classes in Java?

Throwable

135.What is a finally block in Java?

Execute at any cost System.exit()

136.What is the use of finally block in Java?

137.Can we create a finally block without creating a catch block?

Yes,with try block possible.

138.Do we have to always put a catch block after a try block?

No,try block can processed with catch block.

139.In what scenarios, a finally block will not be executed?

System.exit(0/1);

140.Can we re-throw an Exception in

Java?

no

141. What is the difference between throw and throws in Java?

142.What is the concept of Exception Propagation?

143.When we override a method in a Child class, can we throws an additional Exception that is not thrown by the Parent class method?

No

For check exception = no

For unchecked exception = yes

Multi-threading

144.How Multi-threading works in Java?

145.What are the advantages of Multithreading?

146.What are the disadvantages of Multithreading?

147.What is a Thread in Java?

148.What is a Thread's priority and how it is used in scheduling?

149.What are the differences between Pre-emptive Scheduling Scheduler and Time Slicing Scheduler?

150.Is it possible to call run() method instead of start() on a thread in Java?

Yes but it works as normal it will not create thread in jvm to run.

151. How will you make a user thread into daemon thread if it has already started?

Throw exception : `java.lang.IllegalThreadStateException`

the JVM does not wait for Daemon thread before exiting while it waits for user threads, it does not exit until unless all the user threads finish their execution.

152.Can we start a thread two times in Java?

No,throw exception on runtime.

153.In what scenarios can we interrupt a thread?

If any **thread** is in sleeping or waiting state then using **interrupt**() method, **we can interrupt** the execution of that **thread** by showing InterruptedException. A **thread** which is in the sleeping or waiting state **can** be **interrupted** with the help of **interrupt**() method of **Thread** class.

154.In Java, is it possible to lock an object for exclusive use by a thread?

Yes. You can lock an object by putting it in a “synchronized” block. The locked object is inaccessible to any thread other than the one that explicitly claimed it.

155.How notify() method is different from notifyAll() method?

Notify awake one thread in monitor but notifyall awake all thread are in wait state.

Collections

156.What are the differences between the two data structures: a Vector and an ArrayList?

157.What are the differences between Collection and Collections in Java?

158.In which scenario, LinkedList is better than ArrayList in Java?

159.What are the differences between a List and Set collection in Java?

List : duplicates Set: unique

160.What are the between a HashSet and TreeSet collection in Java?

TreeSet : natural order, not accept null because comparator use internally which not allow null values

HashSet : allow one null value. Random order.

161. In Java, how will you decide when to use a List, Set

or a Map collection?

162.What are the differences between a HashMap and a Hashtable in Java?

163.What are the differences between a HashMap and a TreeMap?

164.What are the differences between Comparable and Comparator?

165.In Java, what is the purpose of Properties file?

166.What is the reason for overriding equals() method?

Object's equals does reference equality and the entire reason we are overriding equals is to get logical equality instead of reference equality.

167.How does

method work in Java?

168.Is it a good idea to use Generics in collections?

Yes

Mixed Questions

169.What are Wrapper classes in Java?

A Wrapper class is a class whose object wraps or contains a primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store a primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

170.What is the purpose of native method in Java?

The native keyword is applied to a method to indicate that the method is implemented in native code using JNI (Java Native Interface). native is a modifier applicable only for methods and we can't apply it anywhere else. The methods which are implemented in C, C++ are called as native methods or foreign methods.

The main objectives of native keyword are:

- **To improve performance of the system.**
- **To achieve machine level/memory level communication.**
- **To use already existing legacy non-java code.**

171. What is System class?

172.What is System, out and println in System.out.println method call?

System. out. println is a Java statement that prints the argument passed, into the System. out which is generally stdout. System – is a final class in java

173.What is the other name of Shallow Copy in Java?

Shallow cloning

174.What is the difference between Shallow Copy and Deep Copy in Java?

175.What is a Singleton class?

176.What is the difference between Singleton class and Static class?

Static block does not follow OOPs concept, cannot be instantiated,doesn’t support runtime polymorphism,all its members are static. it cannot be serialized because static cannot create object.

Java Collection

177.What is the difference between Collection and Collections Framework in Java?

178.What are the main benefits of Collections Framework in Java?

Reduce programming effort by using data structure.

179.What is the root interface of Collection hierarchy in Java?

java.util Collection interface

java.lang iterable

180.What are the main differences between Collection and Collections?

181. What are the Thread-safe classes in Java Collections

framework?

Vector,Hashtable,Stack,Property

182.How will you efficiently remove elements while iterating a Collection?

183.How will you convert a List into an array of integers like- int[]?

Arrays.toArray() method

184.How will you convert an array of primitive integers int[] to a List collection?

Arrays.asList() method

185.How will you run a filter on a Collection?

```
List<Person> beerDrinkers = persons.stream()
    .filter(p -> p.getAge() > 16).collect(Collectors.toList());
```

186.How will you convert a List to a Set?

```
List<String> list = new ArrayList();
Set<String> set = new HashSet(list);
```

187.How will you remove duplicate elements from an ArrayList?

Same as above

188.How can you maintain a Collection with elements in

Sorted order?

Using comparator or comparable.

189.What is the difference between Collections.emptyList() and creating new instance of Collection?

Collections.emptyList() = immutable

190.How will you copy elements from a Source List to another list?

191. What are the Java Collection classes that implement List interface?

192.What are the Java Collection classes that implement Set interface?

193.What is the difference between an Iterator and ListIterator in Java?

List Iterator : forward and backward,iterate only list,perform add(),remove(),set() while transverse

Iterator : forward direction,iterate list,set and queue,perform remove while transversing

194.What is the difference between Iterator and Enumeration?

Enumeration : Fail safe,faster than iterator,transverse only legacy element

Iterator : Fail Fast,Slower,transverse legacy and non legacy element.

195.What is the difference between an ArrayList and a LinkedList data structure?

ArrayList : dynamic array,index based, get operation fast ,remove and add operation is slower than linked list.

LinkedList : doubly linked list , remove and add operation is fast

196.What is the difference between a Set and a Map in Java?

197.What is the use of a Dictionary class?

Dictionary is an abstract class that represents a key/value storage repository and operates much like Map

198.What is the default size of load factor in a HashMap collection in Java?

0.75f

199.What is the significance of load factor in a HashMap in Java?

Load factor used to check dynamically increments of size.

200.What are the major differences between a HashSet and a HashMap?

201.What are the similarities between a HashSet and a HashMap in Java?

202.What is the reason for overriding equals() method?

For equality of the object.

203.How can we synchronize the elements of a List, a Set or a Map?

By using Collections.synchronizedList() for the list case.

204.What is Hash Collision? How Java handles hash-collision in HashMap?

205.What are the Hash Collision resolution techniques?

206.What is the difference between Queue and Stack data structures?

207.What is an Iterator in Java?

208.What is the difference between Iterator and Enumeration in Java?

209.What is the design pattern used in the implementation of Enumeration in Java?

Iterator design pattern

Strategy Design Pattern wrong

210.Which methods do we need to override to use an object as key in a HashMap?

Equal and hashCode method.

211. How will you reverse a List in Java?

212.How will you convert an array of String objects into a List?

List<String> list = Arrays.asList(new String[]{"a,b,c,d,d"});

213.What is the difference between peek(), poll() and remove() methods of Queue interface in java?

Peek = return head element

Poll or remove = delete element but if queue is empty then poll return null and remove throw NoSuchElementFound exception

214.What is the difference between Array and ArrayList in Java?

Array = fixed length,form object for both class and primitive data type;

ArrayList = not fixed length it increase size dynamically,form object of Class type doesnot suppport primitive datatype.

215.How will you insert, delete and retrieve elements from a HashMap collection in Java?

Insert = put,delete=remove,retrieve = get

216.What are the main differences between HashMap and ConcurrentHashMap in Java?

217.What is the increasing order of performance for following collection classes in Java?

218.Why does Map interface not extend Collection interface in Java?

Map represent key value pair concept and collection interface support one element concept to add.

219.What are the different ways to iterate elements of a list in Java?

Iterator and listiterator

220.What is CopyOnWriteArrayList? How it is different from ArrayList in Java?

Follow 245

221.How remove() method is implemented in a HashMap?

222.What is BlockingQueue in Java Collections?

Interface,doesnot except null,synchronized

223.How is TreeMap class implemented in Java?

Using red Black Tree concept,due to which TreeMap is in sorted order.

Root node - black,

Left = Less item than root node,Right =
greater item than root node

224.What is the difference between Fail-fast and Fail-safe iterator in Java?

Fail Fast = ConcuurrentModification Exception throw,worked on clone object ex : HashMap

225.How does ConcurrentHashMap work in Java?

Internally support hashtable

Provide 16 segment lock to make thread safe
object.

226.What is the importance of hashCode() and equals() methods?

227.What is the contract of hashCode() and equals() methods in Java?

228.What is an EnumSet in Java?

enum gfg{A,B,C}

EnumSet<gfg> es = new
EnumSet<>();

229.What are the main Concurrent Collection classes

in Java?

ConcurrentHashMap,Blocking Queue,CopyOnWriteArrayList,Deque,ConncurrentSkipList Map

230.How will you convert a Collection to SynchronizedCollection in Java?

Collection.SynchronizedCollection(collectionObject)

231.How IdentityHashMap is different from a regular Map in Java?

Check Address equality and fail fast.

232.What is the main use of IdentityHashMap?

To store entry reference like in HashMap

To store proxy data

233.How can we improve the performance of IdentityHashMap?

234.Is IdentityHashMap thread-safe?

No

235.What is a WeakHashMap in Java?

Automatically remove the entry

when key is no longer used.

236.How can you make a Collection class read Only in Java?

Collections. unModifiableollection()

237.When is UnsupportedOperationException thrown in Java?

An UnsupportedOperationException is a subclass of RuntimException in Java and it can be thrown to indicate that the requested operation is not supported. The UnsupportedOperationException class is a member of the Java Collections Framework. This exception is thrown by almost all of the concrete collections like List, Queue, Set and Map.

238.Let say there is a Customer class. We add objects of Customer class to an ArrayList. How can we sort the Customer objects in ArrayList by using customer firstName attribute of Customer class?

239.What is the difference between Synchronized Collection and Concurrent Collection?

The main reason for this slowness is locking; **synchronized collections** locks the whole **collection** e.g. whole Map or List while **concurrent collection** never locks the whole Map or List.

240.What is the scenario to use ConcurrentHashMap in Java?

241.How will you create an empty Map in Java?

The emptyMap() method is used to get the empty map (immutable). This map is serializable.

```
import java.util.*;
```

```
public class CollectionsDemo {

   public static void main(String args[]) {



      // create an empty map

      Map<String, String> emptymap =
Collections.emptyMap();



      System.out.println("Created empty
immutable map: "+emptymap);



      // try to add elements

      emptymap.put("1","value");

   }

}
```

242.What is the difference between remove() method of Collection and remove() method of Iterator?

Use of collections remove method throw concurrent modification exception.

243.Between an Array and ArrayList, which one is the preferred collection for storing objects?

244.Is it possible to replace Hashtable with ConcurrentHashMap

in Java?

it can be safely used in the concurrent multi-threaded environment but also provides better performance over **Hashtable** and synchronizedMap. **ConcurrentHashMap** is best suited when you have multiple readers and few writers.

245.How CopyOnWriteArrayList class is different from ArrayList and Vector classes?

- Using CopyOnWriteArrayList is costly for update operations, because each mutation creates a cloned copy of underlying array and add/update element to it.
- It is thread-safe version of ArrayList. Each thread accessing the list sees its own version of snapshot of backing array created while initializing the iterator for this list.
- Because it gets snapshot of underlying array while creating iterator, it does not throw ConcurrentModificationException.
- Mutation operations on iterators (remove, set, and add) are not supported. These methods throw `UnsupportedOperationException`.
- CopyOnWriteArrayList is a concurrent replacement for a synchronized List and offers better concurrency when iterations outnumber mutations.
- It allows duplicate elements and heterogeneous Objects (use generics to get compile time errors).
- Because it creates a new copy of array everytime iterator is created, performance is slower than ArrayList.

CopyOnWriteArrayList = fail safe,create clone copy of array after iterator (update) that's why it is not throw concurrent modification exception,worked on clone object,increase memory heads,

Vector = fail fast,throw concurrent modification exception

246.Why ListIterator has add() method but Iterator does not have?

ListIterator can iterate in the both directions of a Collection. It maintains two

pointer for previous and next element. In ListIterator we can use add() method to add an element into the list immediately before the element returned by next() method.

So a subsequent call to next() method will not be affected. And the call to the previous() method will return the newly added element.In Iterator we can only traverse in one direction. So there is no purpose of add() method there.

247.Why do we sometime get ConcurrentModificationException during iteration?

248.How will you convert a Map to a List in Java?

```
Map<Integer, String> myMap = new HashMap<>();

	myMap.put(1, "Java");

	myMap.put(2, "JavaFX");

	myMap.put(3, "CoffeeScript");

	myMap.put(4, "TypeScript");



	ArrayList<Integer> keyList = new
ArrayList<Integer>(myMap.keySet());

	ArrayList<String> valueList = new
ArrayList<String>(myMap.values());
```

249.How can we create a Map with reverse view and lookup in Java?

250.How will you create a shallow copy of a Map?

clone () method

251.Why we cannot create a generic array in Java?

https://stackoverflow.com/questions/2927391/whats-the-reason-i-cant-create-generic-array-types-in-java#:~:text=Arrays%20of%20generic%20types%20are,sound%20and%20not%20dynamically%20checked.

252.What is a PriorityQueue in Java?

Not null,duplicates,FIFO principle

253.What are the important points to remember while using Java Collections Framework?

All class and interface

254.How can we pass a Collection as an argument to a method and ensure that method will not be able to modify it?

We can create a read-only collection using Collections.unmodifiableCollection(Collection c) method before passing it as argument, this will make sure that any operation to change the collection will throw UnsupportedOperationException.

255.Can you explain how HashMap works in Java?

256.Can you explain how HashSet is implemented

in Java?

257.What is a NavigableMap in Java?

Abtract class

258.What is the difference between descendingKeySet() and descendingMap() methods of NavigableMap?

259.What is the advantage of NavigableMap over Map?

260.What is the difference between headMap(), tailMap() and subMap() methods of NavigableMap?

261.How will you sort objects by Natural order in a Java List?

262.How can we get a Stream from a List in Java?

List. **stream**()

263.Can we get a Map from a Stream in Java?

```
List<Employee> employeeList = new
ArrayList<>(Arrays.asList(

                              new Employee(1,
"A", 100),

                              new Employee(2,
"A", 200),
```

```
                                new Employee(3,
"B", 300),

                                new Employee(4,
"B", 400),

                                new Employee(5,
"C", 500),

                                new Employee(6,
"C", 600)));



        Map<String, List<Employee>>
employeesMap = employeeList.stream()


.collect(Collectors.groupingBy(Employee::ge
tName));



        System.out.println(employeesMap);
```

264.What are the popular implementations of Deque in Java?

Queue

Advanced Multi-threading

265.What is a Thread in Java?

266.What is the priority of a Thread and how it is used in scheduling?

267.What is the default priority of a thread in Java?

268.What are the three different priorities that can be set on a Thread in Java?

269.What is the purpose of join() method in Thread class?

270.What is the fundamental difference between wait() and sleep() methods?

271.Is it possible to call run() method instead of start() on a thread in Java?

272.What is a daemon thread in Java?

273.How can we make a regular thread Daemon thread in Java?

274.How will you make a user thread into daemon thread if it has already started?

275.Can we start a thread two times in Java?

276.What is a Shutdown hook in

Java?

277.What is synchronization in Java?

278.What is the purpose of Synchronized block in Java?

279.What is static synchronization?

280.What is a Deadlock situation?

281.What is the meaning of concurrency?

282.What is the main difference between process and thread?

283.What is a process and thread in the context of Java?

284.What is a Scheduler?

285.What is the minimum number of Threads in a Java program?

286.What are the properties of a Java thread?

287.What are the different states of a Thread in Java?

288.How will you set the priority of a thread in Java?

289.What is the purpose of Thread Groups in Java?

290.Why we should not stop a thread by calling its stop() method?

291.How will you create a Thread in Java?

292.How can we stop a thread in the middle of execution in Java?

293.How do you access the current thread in a Java program?

294.What is Busy waiting in Multi-threading?

295.How can we prevent busy waiting in Java?

296.Can we use Thread.sleep() method for real-time processing in Java?

297.Can we wake up a thread that has been put to sleep by using Thread.sleep() method?

298.What are the two ways to check if a Thread has been interrupted?

299.How can we make sure that Parent thread waits for

termination of Child thread?

300.How will you handle InterruptedException in Java?

301.Which intrinsic lock is acquired by a synchronized method in Java?

302.Can we mark a constructor as synchronized in Java?

303.Can we use primitive values for intrinsic locks?

304.Do we have re-entrant property in intrinsic locks?

305.What is an atomic operation?

306.Can we consider the statement i++ as an atomic operation in Java?

307.What are the Atomic operations in Java?

308.Can you check if following code is thread-safe?

309.What are the minimum requirements for a Deadlock situation in a program?

310.How can we prevent a Deadlock?

311. How can we detect a Deadlock situation?

312.What is a Livelock?

313.What is Thread starvation?

314.How can a synchronized block cause Thread starvation in Java?

315.What is a Race condition?

316.What is a Fair lock in multi-threading?

317.Which two methods of Object class can be used to implement a Producer Consumer scenario?

318.How JVM determines which thread should wake up on notify()?

319.Check if following code is thread-safe for retrieving an integer value from a Queue?

320.How can we check if a thread has a monitor lock on a given object?

321.What is the use of yield() method in Thread class?

322.What is an important point to consider while passing an

object from one thread to another thread?

323.What are the rules for creating Immutable Objects?

324.What is the use of ThreadLocal class?

325.What are the scenarios suitable for using ThreadLocal class?

326.How will you improve the performance of an application by multi- threading?

327.What is scalability in a Software program?

328.How will you calculate the maximum speed up of an application by using multiple processors?

329.What is Lock contention in multi-threading?

330.What are the techniques to reduce Lock contention?

331.What technique can be used in following code to reduce Lock contention?

332.What is Lock splitting technique?

333.Which technique is used in ReadWriteLock class for reducing Lock contention?

334.What is Lock striping?

335.What is a CAS operation?

336.Which Java classes use CAS operation?

337.Is it always possible to improve performance by object pooling in a multi-threading application?

338.How can techniques used for performance improvement in a single thread application may degrade the performance in a multi-threading application?

339.What is the relation between Executor and ExecutorService interface?

340.What will happen on calling submit() method of an ExecutorService instance whose queue is already full?

341.What is a ScheduledExecutorService?

342.How will you create a Thread pool in Java?

343.What is the main difference between Runnable and Callable interface?

344.What are the uses of Future interface in Java?

345.What is the difference in concurrency in HashMap and in Hashtable?

346.How will you create synchronized instance of List or Map Collection?

347.What is a Semaphore in Java?

348.What is a CountDownLatch in Java?

349.What is the difference between CountDownLatch and CyclicBarrier?

350.What are the scenarios suitable for using Fork/Join framework?

351.What is the difference between RecursiveTask and RecursiveAction class?

352.In Java 8, can we process stream operations with a Thread pool?

353.What are the scenarios to use parallel stream in Java 8?

354.How Stack and Heap work in Java multi-threading environment?

355.How can we take Thread dump in Java?

356.Which parameter can be used to control stack size of a

thread in Java?

357.There are two threads T1 and T2? How will you ensure that these threads run in sequence T1, T2 in Java?

Java 8

358.What are the new features released in Java 8?

\359.What are the main benefits of new features introduced in Java 8?

360.What is a Lambda expression in Java 8?

361.What are the three main parts of a Lambda expression in Java?

362.What is the data type of a Lambda expression?

363.What is the meaning of following lambda expression?

364.Why did Oracle release a new version of Java like Java 8?

365.What are the advantages of a lambda expression?

366.What is a Functional interface in Java 8?

367.What is a Single Abstract Method (SAM) interface in Java 8?

368.How can we define a Functional interface in Java 8?

369.Why do we need Functional interface in Java?

370.Is it mandatory to use @FunctionalInterface annotation to define a Functional interface in Java 8?

371.What are the differences between Collection and Stream API in Java 8?

372.What are the main uses of Stream API in Java 8?

373.What are the differences between Intermediate and Terminal Operations in Java 8 Streams?

374.What is a Spliterator in Java 8?

375.What are the differences between Iterator and Spliterator in Java 8?

Iterator	Spliterator
Introduced in Java 1.2.	Introduced in Java 1.8.

It is an Iterator for whole Collection API.	It is an Iterator for both Collection and Stream API, except Map implemented classes.
It is an Universal Iterator.	It is NOT an Universal Iterator.
It does NOT support Parallel Programming.	It supports Parallel Programming.

376.What is Type Inference in Java 8?

```
List<Integer> list2 = new ArrayList<>();
```

377.Does Java 7 support Type Inference?

No

Ex : List<Integer> list1 = new ArrayList<Integer>();

378.How does Internal Iteration work in Java 8?

379.What are the main differences between Internal and External Iterator?

https://howtodoinjava.com/java8/java-8-tutorial-internal-vs-external-iteration/

380.What are the main advantages of Internal Iterator over

External Iterator in Java 8?

381.What are the applications in which we should use Internal Iteration?

382.What is the main disadvantage of Internal Iteration over External Iteration?

383.Can we provide implementation of a method in a Java Interface?

Yes static and default method need method implementation

384.What is a Default Method in an Interface?

385.Why do we need Default method in a Java 8 Interface?

The reason we have default methods in interfaces is to allow the developers to add new methods to the interfaces without affecting the classes that implements these interfaces.

386.What is the purpose of a Static method in an Interface in Java 8?

Java interface static method helps us in providing security by not allowing implementation classes to override them.

387.What are the core ideas behind the Date/Time API of Java 8?

388.What are the advantages of new Date and Time API in Java 8 over old Date API?

389.What are the main differences between legacy Date/Time API in Java and Date/Time API of Java 8?

390.How can we get duration between two dates or time in Java 8?

391.What is the new method family introduced in Java 8 for processing of Arrays on multi core machines?

392.How does Java 8 solve Diamond problem of Multiple Inheritance?

393.What are the differences between Predicate, Supplier and Consumer in Java 8?

394.Is it possible to have default method definition in an interface without marking it with default keyword?

no

395.Can we create a class that implements two Interfaces with default methods of same name and signature?

Yes one method invocation.

Can a java class implement Two interfaces with same methods having the same signature but **different return types**??

No, its an error

If two interfaces contain a method with the same signature but different return types, then it is impossible to implement both the interface simultaneously.

According to JLS (§8.4.2) methods with same signature is not allowed in this case.

396.How Java 8 supports Multiple Inheritance?

```java
interface Clickable{
	default void click(){
		System.out.println("click");
	}

	default void print(){
		System.out.println("Clickable");
	}
}

interface Accessible{
	default void access(){
		System.out.println("access");
	}

	default void print(){
		System.out.println("Accessible");
	}
}

public class Button implements Clickable, Accessible {

	public void print(){
		Clickable.super.print();
		Accessible.super.print();
	}

	public static void main(String[] args) {
		Button button = new Button();
		button.click();
		button.access();
		button.print();
	}
}
```

397.In case we create a class that extends a base class and implements an interface. If both base class and interface have a default method with same name and arguments, then which definition will be picked by JVM?

Interface

398.If we create same method and define it in a class , in its parent class and in an interface implemented by the class, then definition will be invoked if we access it using the reference of Interface and the object of class?

By reference of object because when we call by using reference of interface it shows compile time error "`non-static method stop() cannot be referenced from a static context`

"

**399.Can we access a static method of an interface by using reference of
the
interface?**

400.How can you get the name of Parameter in Java by using reflection?

**401.What is Optional in
Java 8?**

**402.What are the uses of
Optional?**

403.Which method in Optional provides the fallback mechanism

in case of null value?

404.How can we get current time by using Date/Time API of Java 8?

405.Is it possible to define a static method in an Interface?

406.How can we analyze the dependencies in Java classes and packages?

407.What are the new JVM arguments introduced by Java 8?

408.What are the popular annotations introduced in Java 8?

409.What is a StringJoiner in Java 8?

410.What is the type of a Lambda expression in Java 8?

411. What is the target type of a lambda expression ?

412.What are the main differences between an interface with default method and an abstract class in Java 8?

Abstract class have constructor

when any class extends an abstract class, the constructor of sub class will invoke the constructor of super class either implicitly or explicitly. This chaining of constructors is one of the reasons abstract class can have constructors in Java.

public class Main

```
{
public static void main (String args[])
  {
Server server = new Tomcat ("Apache Tomcat");
server.start ();
}
}
abstract class Server
{
protected final String name;
public Server (String name)
  {
this.name = name;
}
public abstract boolean start ();
}
class Tomcat extends Server
{
public Tomcat (String name)
  {
super (name);
}
```

```
@Override public boolean start ()

 {

System.out.println (this.name + " started successfully");

return true;

}

}
```

Java Tricky Questions

413.Is there any difference between a = a + b and a += b expressions?

```
byte a = 1; int b = 2;

a += b; // compiles

a = a + b; // doesn't compile as a byte + int = int
a = (byte) (a + b);

 // compiles as this is the same as +=
```

414.What does the expression 1.0 / 0.0 return? Will there be any compilation error?

Infinity

415.Can we use multiple main methods in multiple classes?

Yes,by overloading

416.Does Java allow you to override a private or static method?

NO

417.What happens when you put a key object in a HashMap that is already present?

Replace it value

418.How can you make sure that N threads can access N resources without deadlock?

419.How can you determine if JVM is 32-bit or 64-bit from Java Program?

System.getProperties()

420.What is the right data type to represent Money (like Dollar/Pound) in Java?

Big Decimal

421.How can you do multiple inheritances in Java?

422.Is ++ operation thread-safe in Java?

423.How can you access a non-static variable from the static context?

By creating object of class and call it with that object

424.Let say there is a method that throws NullPointerException in the superclass. Can we override it with a method that throws RuntimeException?

no

425.How can you mark an array volatile in Java?

Yes, you can make an array (both primitive and reference type array e.g. an int array and String array) volatile in Java but only changes to reference pointing to an array will be visible to all threads, not the whole array.

426.What is a thread local variable in Java?

427.What is the difference between sleep() and wait() methods in Java?

428.Can you create an Immutable object that contains a mutable object?

429.How can you convert an Array of bytes to String?

430.What is difference between CyclicBarrier and CountDownLatch class?

431.What is the difference between StringBuffer and StringBuilder?

StringBuffer = synchronized,less efficient

StringBuilder = not synchronized , more efficient

432.Which class contains clone method? Cloneable or Object class?

Object

433.How will you take thread dump in Java?

434.Can you cast an int variable into a byte variable? What happens if the value of int is larger than byte?

435.In Java, can we store a double value in a long variable without explicit casting?

No

436.What will this return 5*0.1 == 0.5? true or false?

True

437.Out of an int and Integer, which one takes more memory?

438.Can we use String in the switch case statement in Java?

439.Can we use multiple main methods in same class?

440.When creating an abstract class, is it a good idea to call abstract methods inside its constructor?

441.How can you do constructor chaining in Java?

442.How can we find the memory usage of JVM from Java code?

443.What is the difference between x == y and x.equals(y) expressions in Java?

444. How can you guarantee that the garbage collection takes place?

445.What is the relation between x.hashCode() method and x.equals(y) method of Object class?

446.What is a compile time constant in Java?

A variable of primitive type or type String, that is final and initialized with a compile-time constant expression, is called a constant variable.

```
public final int A_CONSTANT = 5;
```

All compile-time constants are inlined during compile time i.e. their references will be replaced with their values by the compiler both in the current class, and in other classes where they are used.

If we change the constant value in the defining class then we have to recompile all classes which are using it otherwise old value continue to be used.

447.Explain the difference between fail-fast and fail-safe iterators?

Fail Fast Vs Fail Safe Iterators In Java :

Fail-Fast Iterators	Fail-Safe Iterators
Fail-Fast iterators doesn't allow modifications of a collection while iterating over it.	Fail-Safe iterators allow modifications of a collection while iterating over it.
These iterators throw *ConcurrentModificationException* if a collection is modified while iterating over it.	These iterators don't throw any exceptions if a collection is modified while iterating over it.
They use original collection to traverse over the elements of the collection.	They use copy of the original collection to traverse over the elements of the collection.
These iterators don't require extra memory.	These iterators require extra memory to clone the collection.
Ex : Iterators returned by *ArrayList*, *Vector*, *HashMap*.	Ex : Iterator returned by *ConcurrentHashMap.*

448. You have a character array and a String. Which one is more secure to store sensitive data (like password, date of birth, etc.)?

character array it store address of the object.

449.Why do you use volatile keyword in Java?

To make variable global for the application in multithread environment.

450.What is the difference between poll() and remove() methods of Queue in Java?

Poll - return null when queue is empty

Remove - throw objectNotFoundexception when queue is empty.

451.Can you catch an exception thrown by another thread in Java?

452.How do you decide which type of Inner Class – Static or Non-Static to use in Java?

453.What are the different types of Classloaders in Java?

454.What are the situations in which you choose HashSet or TreeSet?

455.What is the use of method references in Java?

456.Do you think Java Enums are more powerful than integer constants?

457.Why do we use static initializers in Java?

458.Your client is complaining that your code is throwing NoClassDefFoundError or NoSuchMethodError, even though you are able to compile your code without error and method exists in your code. What could be the reason behind this?

It means class in not available on runtime.

459.How can you check if a String is a number by using regular expression?

460.What is the difference between the expressions String s = "Temporary" and String s = new String("Temporary ")? Which one is better and more efficient?

461.In Java, can two equal objects have the different hash code?

462.How can we print an Array in Java?

463.Is it ok to use random numbers in the implementation of hashcode() method in Java?

464.Between two types of dependency injections, constructor injection and setter dependency injection, which one is better?

constructor injection because it initialize all value

465.What is the difference between DOM and SAX parser in Java?

466.Between Enumeration and Iterator, which one has better

performance in Java?

Enumerator

467.What is the difference between pass by reference and pass by value?

468.What are the different ways to sort a collection in Java?

469.Why Collection interface doesn’t extend Cloneable and Serializable interfaces?

470.What is the difference between a process and a thread in Java?

471.What are the benefits of using an unordered array over an ordered array?

472.Between HashSet and TreeSet collections in Java, which one is better?

473.When does JVM call the finalize() method?

474.When would you use Serial Garabage collector or Throughput Garbage collector in Java?

475.In Java, if you set an object reference to null, will the Garbage Collector immediately free the memory held by that object?

476.How can you make an Object eligible for Garbage collection in Java?

477.When do you use Exception or Error in Java? What is the difference between these two?

478.What is the advantage of PreparedStatement over Statement class in Java?

479.In Java, what is the difference between throw and throws keywords?

480.What happens to the Exception object after the exception handling is done?

481.How do you find which client machine is sending request to your servlet in Java?

482.What is the difference between a Cookie and a Session object in Java?

483.Which protocol does Browser and Servlet use to communicate with each other?

484. What is HTTP Tunneling?

485.Why do we use JSP instead of Servlet in Java?

486.Is empty '.java' file name a valid source file name in Java?

487.How do you implement Servlet Chaining in Java?

488.Can you instantiate this

class?

489.Why Java does not support operator overloading?

490.Why String class is Immutable or Final in Java?

491.What is the difference between sendRedirect and forward methods?

492.How do you fix your Serializable class, if it contains a member that is not serializable?

493.What is the use of run time polymorphism in Java?
494.What are the rules of method overloading and method overriding in Java?

495.What is the difference between a class and an object in Java?

496.Can we create an abstract class that extends another abstract class?

497.Why do you use Upcasting or Downcasting in Java ?

498.What is the reason to organize classes and interfaces in a package in Java?

499.What is information hiding in Java?

Encapsulation

500.Why does Java provide default constructor?

501.What is the difference between super and this keywords in Java?

502.What is the advantage of using Unicode characters in Java?

503.Can you override an overloaded method in Java?

504.How can we change the heap size of a JVM?

505.Why should you define a default constructor in Java?

506.How will you make an Object Immutable in Java?

507.How can you prevent SQL Injection in Java Code?

508.Which two methods should be always implemented by HashMap key Object?

509.Why an Object used as Key in HashMap should be Immutable?

510.How can we share an object between multiple threads?

511. How can you determine if your program has a

deadlock?

JSP

512.What are the implicit objects in JSP?

513.How will you extend JSP code?

514.How will you handle runtime exceptions in JSP?

515.How will you prevent multiple submits of a page that come by clicking refresh button multiple times?

516.How will you implement a thread safe JSP page?

517.How will you include a static file in a JSP page?

518.What are the lifecycle methods of a JSP?

519.What are the advantages of using JSP in web architecture?

520.What is the advantage of JSP over Javascript?

521.What is the Lifecycle of JSP?

522.What is a JSP expression?

523.What are the different types of directive tags in JSP?

524.What is session attribute in JSP?

525.What are the different scopes of a JSP object?

526.What is pageContext in JSP?

527.What is the use of jsp:useBean in JSP?

528.What is difference between include Directive and include Action of JSP?

529.How will you use other Java files of your application in JSP code?

530.How will you use an existing class and extend it to use in the JSP?

531.Why _jspService method starts with _ symbol in JSP?

532.Why do we use tag library in JSP?

533.What is the different type of tag library groups in

JSTL?

534.How will you pass information from one JSP to another JSP?

535.How will you call a stored procedure from JSP?

536.Can we override _jspService() method in JSP?

537.What is a directive in JSP?

538.How will you implement Session tracking in JSP?

539.How do you debug code in JSP?

540.How will you implement error page in JSP?

541.How will you send XML data from a JSP?

542.What happens when we request for a JSP page from web browser?

543.How will you implement Auto Refresh of page in JSP?

544.What are the important status codes in HTTP?

545.What is the meaning of Accept attribute in HTTP header?

546.What is the difference between Expression and Scriptlet in JSP?

547.How will you delete a Cookie in JSP?

548.How will you use a Cookie in JSP?

549.What is the main difference between a Session and Cookie in JSP?

550.How will you prevent creation of session in JSP?

551.What is an output comment in JSP?

552.How will you prevent caching of HTML output by web browser in JSP?

553.How will you redirect request to another page in browser in JSP code?

554.What is the difference between sendRedirect and forward in a JSP?

555.What is the use of config implicit object in JSP?

556.What is the difference between init-param and

context-param?

557.What is the purpose of RequestDispatcher?

558.How can be read data from a Form in a JSP?

559.What is a filter in JSP?

560.How can you upload a large file in JSP?

561.In which scenario, Container initializes multiple JSP/Servlet objects?

Java Design Patterns

562.When will you use Strategy Design Pattern in Java?

563.What is Observer design pattern?

564.What are the examples of Observer design pattern in JDK?

565.How Strategy design pattern is different from State design pattern in Java?

566.Can you explain Decorator design pattern with an example in Java?

567.What is a good scenario for using Composite design Pattern in Java?

568.Have you used Singleton design pattern in your Java project?

569.What are the main uses of Singleton design pattern in Java project?

570.Why java.lang.Runtime is a Singleton in Java?

571.What is the way to implement a thread-safe Singleton design pattern in Java?

572.What are the examples of Singleton design pattern in JDK?

573.What is Template Method design pattern in Java?

574.What are the examples of Template method design pattern in JDK?

575.Can you tell some examples of Factory Method design pattern implementation in Java?

576.What is the benefit we get by using static factory method to create object?

577.What are the examples of Builder design pattern in JDK?

578.What are the examples of Abstract Factory design pattern

in JDK?

579.What are the examples of Decorator design pattern in JDK?

580.What are the examples of Proxy design pattern in JDK?

581.What are the examples of Chain of Responsibility design pattern in JDK?

582.What are the main uses of Command design pattern?

583.What are the examples of Command design pattern in JDK?

584.What are the examples of Interpreter design pattern in JDK?

585.What are the examples of Mediator design pattern in JDK?

586.What are the examples of Strategy design pattern in JDK?

587.What are the examples of Visitor design pattern in JDK?

588.How Decorator design pattern is different from Proxy pattern?

589.What are the different scenarios to use Setter and Constructor based injection in Dependency Injection (DI) design pattern?

590.What are the different scenarios for using Proxy design pattern?

591.What is the main difference between Adapter and Proxy design pattern?

592.When will you use Adapter design pattern in Java?

593.What are the examples of Adapter design pattern in JDK?

594.What is the difference between Factory and Abstract Factory design pattern?

595.What is Open/closed design principle in Software engineering?

596.What is SOLID design principle?

597.What is Builder design pattern?

598.What are the different categories of Design Patterns used in Object Oriented Design?

I km-599.What is the design pattern suitable to access elements of a Collection?

600.How can we implement Producer Consumer design pattern in Java?

601.What design pattern is suitable to add new features to an

existing object?

602.Which design pattern can be used when to decouple abstraction from the implementation?

603.Which is the design pattern used in Android applications?

604.How can we prevent users from creating more than one instance of singleton object by using clone() method?

605.What is the use of Interceptor design pattern?

606.What are the Architectural patterns that you have used?

607.What are the popular uses of Façade design pattern?

608.What is the difference between Builder design pattern and Factory design pattern?

609.What is Memento design pattern?

610.What is an AntiPattern?

611. What is a Data Access Object (DAO) design pattern?

Spring Questions

612.What is Spring framework?

613.What are the benefits of Spring framework in software development?

614.What are the modules in Core Container of Spring framework?

The Spring Core container contains core, beans, context and expression language (EL) modules.

Core and beans =provides features of IOC and DI.

Context = This module supports internationalization (I18N), EJB, JMS, Basic Remoting.

SpEl = It is an extension to the EL defined in JSP. It provides support to setting and getting property values, method invocation, accessing collections and indexers, named variables, logical and arithmetic operators, retrieval of objects by name etc.

615.What are the modules in Data Access/Integration layer of Spring framework?

This group comprises of JDBC, ORM, OXM, JMS and Transaction modules. These modules basically provide support to interact with the database.

616.What are the modules in Web layer of Spring framework?

This group comprises of Web, Web-Servlet, Web-Struts and Web-Portlet.
These modules provide support to create web application.

617.What is the main use of Core Container module in Spring framework?

1.provide implementation of Bean Factory which important part of Bean.

2.uses of ioc and di.

3.context uses the base of core and context . Application Context is the feature of Context.

4.SpEl defines in Jsp and it manipulate the object graph at runtime.

618.What kind of testing can be done in Spring Test Module?

619.What is the use of BeanFactory in Spring framework?

creating and initializing all the objects, i.e. beans mentioned in the configuration file.

Using XMLBeanfactory,**BeanFactory** instantiate bean when you call **getBean().**

620.Which is the most popular implementation of BeanFactory in Spring?

BeanFactory interface is **XMLBeanFactory.**

621.What is XMLBeanFactory in Spring framework?

To load bean from configuration file.

BeanFactory : Bean factory is a container. It configures, instantiates and manages a set of beans. These beans are collaborated with one another and have dependencies among themselves. The reflection of these dependencies are used in configuring data that is used by BeanFactory.

XMLBeanFactory : XMLBeanFactory is a bean factory that is loaded its beans from an XML file.

What is a BeanFactory and XMLBeanFactory?

- The BeanFactory provides an advanced configuration mechanism capable of managing beans (objects) of any nature, using potentially any kind of storage facility. The BeanFactory is the actual container which instantiates, configures, and manages a number of beans. These beans typically collaborate with one another, and thus have dependencies between themselves. These dependencies are reflected in the configuration data used by the BeanFactory.A BeanFactory is represented by the interface org.springframework.beans.factory.BeanFactory, for which there are multiple implementations.

- Although for most scenarios, almost all user code managed by the BeanFactory does not have to be aware of the BeanFactory, the BeanFactory does have to be instantiated somehow. This can happen via explicit user code such as:

```
Resource res = new FileSystemResource("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory(res);
```

or

```
ClassPathResource res = new ClassPathResource("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory(res);
```

or

```
ClassPathXmlApplicationContext appContext = new ClassPathXmlApplicationContext(
new String[] {"applicationContext.xml", "applicationContext-part2.xml"});
// of course, an ApplicationContext is just a BeanFactory
BeanFactory factory = (BeanFactory) appContext;
```

XMLBeanFactory :

- BeanFactory has many implementations in Spring. But one of the most useful one is org.springframework.beans.factory.xml.XmlBeanFactory, which loads its beans based on the definitions contained in an XML file. To create an XmlBeanFactory, pass a java.io.InputStream to the constructor. The InputStream will provide the XML to the factory.

- For example, the following code snippet uses a java.io.FileInputStream to provide a bean definition XML file to XmlBeanFactory.

```
BeanFactory factory = new XmlBeanFactory(new FileInputStream("beans.xml"));
```

- To retrieve the bean from a BeanFactory, call the getBean() method by passing the name of the bean you want to retrieve.

```
MyBean myBean = (MyBean) factory.getBean("myBean");
```

622.What are the uses of AOP module in Spring framework?

623.What are the benefits of JDBC abstraction layer module in Spring framework?

624.How does Spring support Object Relational Mapping (ORM) integration?

Spring supports Object Relational Mapping (ORM) by providing ORM Module. This module helps in integrating with popular ORM framework like Hibernate, JDO, and iBATIS SQL Maps etc.

Transaction Management module of Spring framework supports all of these ORM frameworks as well as JDBC.

625.How does Web module work in Spring framework?

The Web layer consists of the Web, Web-Servlet, Web-Struts, and Web-Portlet modules.

Spring's *Web* module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using **servlet listeners** and a web-oriented application context. It also contains the web-related parts of Spring's remoting support.

626.What are the main uses of Spring MVC module?

627.What is the purpose of Spring configuration file?

For the declaration of beans to create in container.

628.What is the purpose of Spring IoC container?

It initialize,create and retreive bean from container.

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans.

Two ways

1.Bean Factory :- Lazy loading,Important implementation is **XMLBeanFactory,**need of getBean() method to initialize bean.

2.Application context:-Eager loading,Important

implentation,**ClassPathXmlApplicationContext,**it initialize singleton bean object when start up of container.

629.What is the main benefit of Inversion of Control (IOC) principle?

Loosely coupled

630.Does IOC containers support Eager Instantiation or Lazy loading of beans?

Eager loading

631.What are the benefits of ApplicationContext in Spring?

1.Use to create bean object in container.

2.No need of getBean() method to initialized the singleton scoped bean its created at the time of start up container.

3.ClassPathXmlApplicationContext is used to load the bean from container.

632.How will you implement ApplicationContext in Spring framework?

By using ClassPathXmlApplicationContext

633.Explain the difference between ApplicationContext and BeanFactory in Spring?

Same as 634.

634.Between ApplicationContext and BeanFactory which one is preferable to use in Spring?

1. BeanFactory instantiate bean when you call **getBean()** method while **ApplicationContext** instantiate Singleton bean when container is started, It doesn't wait for getBean() to be called.

2. BeanFactory doesn't provide support for internationalization but **ApplicationContext** provides support for it.

3. Another difference between **BeanFactory** vs **ApplicationContext** is ability to publish event to beans that are registered as listener.

4. One of the popular implementation of **BeanFactory** interface is **XMLBeanFactory** while one of the popular implementation of **ApplicationContext** interface is **ClassPathXmlApplicationContext**.

5. If you are using auto wiring and using **BeanFactory** than you need to register **AutoWiredBeanPostProcessor** using API which you can configure in XML if you are using **ApplicationContext**. In summary **BeanFactory** is OK for testing and non production use but **ApplicationContext** is more feature rich container implementation and should be favored over **BeanFactory.**

635.What are the main components of a typical Spring based application?

Ioc and di

636.Explain Dependency Injection (DI) concept in Spring framework?

- Dependency Injection (DI) is a software design pattern that implements inversion of control for resolving dependencies.
- An injection is the passing of a dependency to a dependent object that would use it.
- DI is a process whereby objects define their dependencies. The other

objects they work with—only through constructor arguments or arguments to a factory method or property—are set on the object instance after it is constructed or returned from a factory method.

- The container then injects those dependencies, and it creates the bean. This process is named Inversion of Control (IoC) (the bean itself controls the instantiation or location of its dependencies by using direct construction classes or a Service Locator).
- DI refers to the process of supplying an external dependency to a software component.

637.What are the different roles in Dependency Injection (DI)?

1. The **service** you want to use.
2. The **client** that uses the service.
3. An **interface** that's used by the client and implemented by the service.
4. The **injector** which creates a service instance and injects it into the client.

Benifits:

That principle improves the reusability of your code and limits the ripple effect if you need to change lower level classes. But even if you implement it perfectly, you still keep a dependency on the lower level class. The interface only decouples the usage of the lower level class but not its instantiation. At some place in your code, you need to instantiate the implementation of the interface. That prevents you from replacing the implementation of the interface with a different one.

The goal of the dependency injection technique is to remove this dependency by separating the usage from the creation of the object. This reduces the amount of required boilerplate code and

improves flexibility.

638.Spring framework provides what kinds of Dependency Injection mechanism?

The basic principle behind Dependency Injection (DI) is that objects define their dependencies only through constructor arguments, arguments to a factory method, or properties which are set on the object instance after it has been constructed or returned from a factory method. Then, it is the job of the container to actually inject those dependencies when it creates the bean. This is fundamentally the inverse, hence the name Inversion of Control (IoC).

6.1. Setter Injection

Setter-based DI is realized by calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

```
public class TestSetterDI {



DemoBean demoBean = null;



public void setDemoBean(DemoBean demoBean) {

   this.demoBean = demoBean;

}

}
```

6.2. Constructor Injection

Constructor-based DI is realized by invoking a constructor with a number of arguments, each representing a collaborator. Additionally, calling a static factory method with specific arguments to construct the bean, can be considered almost equivalent, and the rest of this text will consider arguments to a constructor and arguments to a static factory method similarly.

```
public class ConstructorDI {


DemoBean demoBean = null;


public TestSetterDI (DemoBean demoBean) {

   this.demoBean = demoBean;

}

}
```

6.3. Interface Injection

In this methodology we implement an interface from the IOC framework. IOC framework will use the interface method to inject the object in the main class. It is much more appropriate to use this approach when you need to have some logic that is not applicable to place in a property. Such as logging support.

```
public void SetLogger(ILogger logger)

{

  _notificationService.SetLogger(logger);

  _productService.SetLogger(logger);

}
```

639.In Spring framework, which Dependency Injection is better? Constructor-based DI or Setter-based DI?

Constructor Based

640.What are the advantages of Dependency Injection (DI)?

Makes application loosely coupled

641.What are the disadvantages of Dependency Injection (DI)?

- DI creates clients that demand configure details supplied by construction code.
- DI can make code difficult to trace because it separates behavior from construction; this means developers refer to more files to follow how a system performs.
- DI can cause an explosion of types, especially in languages that have explicit interface types like C# and Java.
- DI can encourage dependence on DI framework.
- Tight coupling :
    - A change in only one module usually forces a ripple effect of changes in other modules.

642.What is a Spring Bean?

Object of java class.IOC maintain life cycle of spring bean.

643.What does the definition of a Spring Bean contain?

Java class and ioc

644.What are the different ways to provide configuration metadata to a Spring Container?

645.What are the different scopes of a Bean supported by Spring?

singleton,prototype,request,session,global

646.How will you define the scope of a bean in Spring?

It means object creation restriction.

647.Is it safe to assume that a Singleton bean is thread safe in Spring Framework?

No,singleton is about creation of instance not about execution.

648.What are the design-patterns used in Spring framework?

Factory pattern,Singleton,MVC pattern,Proxy etc

649.What is the lifecycle of a Bean in Spring framework?

Init , service , destroy

650.What are the two main groups of methods in a Bean's lifecycle?

A Bean in Spring has two main groups of lifecycle methods.

Initialization Callbacks: Once all the necessary properties of a Bean are set by the container, Initialization Callback methods are used for performing initialization work. A developer can implement method afterPropertiesSet() for this work.

Destruction Callbacks: When the Container of a Bean is destroyed, it calls the methods in DisposableBean to do any cleanup work.

There is a method called destroy() that can be used for this purpose to make

Destruction Callbacks.

Recent recommendation from Spring is to not use these methods, since it can strongly couple your code to Spring code.

651.Can we override main lifecycle methods of a Bean in Spring?

Yes, Spring framework allows developers to override the lifecycle methods of a Bean. This is used for writing any custom behavior for Bean.

652.What are Inner beans in Spring?

https://www.java2novice.com/spring/inject-inner-bean/

653.How can we inject a Java Collection in Spring framework?

https://www.concretepage.com/spring/spring-collection-injection-example-with-list-set-map-and-properties

654.What is Bean wiring in Spring?

https://www.careerride.com/Spring-bean-wiring.aspx

- The *default autowire mode in XML configuration* is `no`.
- The *default autowire mode in java configuration* is `byType`.

655.What is Autowiring in Spring?

To inject dependency into another class by using autowire attribute .for example :- autowire-”no,byType,byName etc”

656.What are the different modes of Autowiring supported by Spring?

no , **byName** , **byType** , constructor , and autodetect

657.What are the cases in which Autowiring may not work in Spring framework?

- Explicit dependencies in constructor-argument and property settings always override autowiring. You cannot autowire simple properties such as primitives, Strings, and Classes.
- Overriding possibilities: We can define dependencies using property or constructor-args tag which will always override autowiring.
- Primitive data type: We have to define primitive data types String or Integer using property or constructor-args tag. You cannot autowire these tags.
- Confusing Nature: If you have lot of dependency in a program, then it’s hard to find using autowire attribute of bean.

658.Is it allowed to inject null or empty String values in Spring?

yes

659.What is a Java-based Configuration in

Spring?

660.What is the purpose of @Configuration annotation?

Spring @Configuration annotation allows us to use annotations for dependency injection.

661.What is the difference between Full @Configuration and 'lite' @Beans mode?

https://www.logicbig.com/tutorials/spring-framework/spring-core/bean-definition-in-components.html

662.In Spring framework, what is Annotation-based container configuration?

663.How will you switch on Annotation based wiring in Spring?

664.What is @Autowired annotation?

It is use to inject the dependency.

665.What is @Required annotation?

To make setter method mandatory by using annotation driven or RequiredAnnotationBeanPostProcessor

https://examples.javacodegeeks.com/enterprise-java/spring/spring-required-annotation-example/

666.What are the two ways to enable RequiredAnnotationBeanPostProcessor in

Spring?

667.What is @Qualifier annotation in Spring?

To solve ambiguity problem.

668.How Spring framework makes JDBC coding easier for developers?

669.What is the purpose of JdbcTemplate?

670.What are the benefits of using Spring DAO?

671.What are the different ways to use Hibernate in Spring?

672.What types of Object Relational Mapping (ORM) are supported by Spring?

673.How will you integrate Spring and Hibernate by using HibernateDaoSupport?

674.What are the different types of the Transaction Management supported by Spring framework?

675.What are the benefits provided by Spring Framework's Transaction Management?

676.Given a choice between declarative and programmatic Transaction Management, which method will you choose?

677.What is Aspect Oriented Programming

(AOP)

678.What is an Aspect in Spring?

679.In Spring AOP, what is the main difference between a Concern and a Cross cutting concern?

680.What is a Joinpoint in Spring AOP?

681.What is an Advice in Spring AOP?

682.What are the different types of Advice in Spring AOP?

683.What is a Pointcut in Spring AOP?

684.What is an Introduction in Spring AOP?

685.What is a Target object in Spring AOP?

686.What is a Proxy in Spring AOP?

687.What are the different types of AutoProxy creators in Spring?

688.What is Weaving in Spring AOP?

689.In Spring AOP, Weaving is done at compile time or run time?

690.What is XML Schema-based Aspect implementation?

691.What is Annotation-based aspect implementation in Spring AOP?

692.How does Spring MVC framework work?

693.What is DispatcherServlet?

694.Can we have more than one DispatcherServlet in Spring MVC?

Yes, a **Spring MVC** web application **can have more than one DispatcherServlets**. Each **DispatcherServlet** has to operate in its own namespace. It has to load its own ApplicationContext with mappings, handlers, etc. Only the root application context **will** be shared among these Servlets.

695.What is WebApplicationContext in Spring MVC?

696.What is Controller in Spring MVC framework?

It is an annotation which contains bussiness logic for application.

697.What is @RequestMapping annotation in Spring?

Map URL.

https://www.journaldev.com/3358/spring-requestmapping-requestparam-pathvariable-example

698.What are the main features of Spring MVC?

i. Lightweight

The Spring Framework is very lightweight with respect to its size and functionality. It is due to its POJO implementation which doesn't force to inherit any class or implement any interfaces.

ii. Aspect Oriented Programming(AOP)

It is an important part of Spring Framework. Aspect Oriented Programming is used for separating cross-cutting concerns (for example logging, security etc.) from the business logic of the application.

iii. Transaction Management

Transaction Management use for unify several transaction management APIs and is used to coordinate transactions for Java object. Also, not tie to the J2EE environment and use with containerless environments.

iv. Container

The Spring Framework designs and manages the lifecycle and configurations of application objects.

v. Dependency Injection

Dependency Injection is a feature of Spring Framework allows you to develop loosely coupled applications. Therefore, the unit testing of these loosely coupled applications becomes easier. This also allows the developer to swap

out some of the modules according to its need.

699.What is the difference between a Singleton and Prototype bean in Spring?

Singleton = one instance

Prototype =new object is created each time

700.How will you decide which scope- Prototype or Singleton to use for a bean in Spring?

Prototype scope: A new object is created each time it is injected. **Singleton scope**: The same object is returned each time it is injected. **Prototype scope is used** for all **beans** that are stateful, while the **singleton scope** should **be used** for stateless **beans.**

https://java2blog.com/injecting-prototype-bean-singleton-bean-spring/

701.What is the difference between Setter and Constructor based Dependency Injection (DI) in Spring framework?

1.Setter injection in Spring uses setter methods like `setDependency()` to inject dependency on any bean managed by Spring's IOC container. On the other hand constructor injection uses constructor to inject dependency on any Spring-managed bean.

2.Because of using setter method, setter Injection in more readable than constructor injection in Spring configuration file usually `applicationContext.xml` . Since setter method has name e.g. `setReporotService()` by reading Spring XML config file you know which dependency you are setting. While in constructor injection, since it uses an index to inject the dependency, it's not as readable as setter injection and you need to refer either Java documentation or code to find which index corresponds to which property.

3.Another difference between setter vs constructor injection in Spring and one of the

drawback of setter injection is that it does not ensures dependency Injection. You can not guarantee that certain dependency is injected or not, which means you may have an object with incomplete dependency. On other hand constructor Injection does not allow you to construct object, until your dependencies are ready.

4.By using setter injection, you can override certain dependency which is not possible which is not possible with constructor injection because every time you call the constructor, a new object is gets created.

702.What are the drawbacks of Setter based Dependency Injection (DI) in Spring?

- **No Guarantee**: In Setter based DI, there is no guarantee that a certain dependency is injected or not. We may have an object with partial or no dependency. Whereas in Constructor based DI, an object in not created till the time all the dependencies are ready.
- **Security**: One can use Setter based DI to override another dependency. This can cause Security breach in a Spring application.
- **Circular Dependency**: Setter based DI can cause circular dependency between objects. Where as Constructor based DI will throw ObjectCurrentlyInCreationException if there is a circular dependency during the creation of an object.

703.What are the differences between Dependency Injection (DI) and Factory Pattern?

FACTORY DESIGN EX:

```
public class CashRegister

{

 private PriceCalculator calculator = PriceCalculatorFactory.getInstance();

 public void add(Transaction tx)

{
```

```
 int price  calcualtor.getPrice(tx);
 add(price);
 }
 }
```

DI EX:

```
public class CashRegister {
private PriceCalculator calculator;
public CashRegister(PriceCalculator calculator)
{
 this.calculator = calculator;
}
 public void add(Transaction tx)
{
 int price = calcualtor.getPrice(tx);
 add(price);
 }
 public void setCalcuator(PriceCalculator calc)
{
 this.calculator = calc;
 }
 }
```

1) Factory pattern adds **coupling** between object, factory, and dependency. Object not only needs a dependent object to work properly but also a Factory object. While in case of dependency injection, Object just knows the dependency, it doesn't know anything about container or factory

2) As compared to the Factory pattern, Dependency injection makes unit testing easier. If you use the factory pattern, you need to create the object you want to test, the factory and the dependent object, of course, you factor can return a mock object, but you need all this just to start with unit testing. On the other hand, if you use dependency injection, you just need to mock the dependency and inject into an object you want to test, no clutter or boilerplate is required.

3) Dependency injection is more flexible than the factory pattern. You can even switch to different DI frameworks like Spring IOC or Google Guice.

4) One of the drawbacks of Dependency injection, as compared to Factory pattern, is that you need a container and configuration to inject the dependency, which is not required if you use a factory design pattern.

In a real sense, it's not such a bad thing because you have one place to see the dependency of your class and you can control them, but yes when you compare DI to a factory method, this is the additional step you need to do.

5) Due to low coupling, DI results in much cleaner co than factory pattern. Your object looks like POJO, and you also come to know what is mandatory and what is an option by looking at which type of dependency injection your class is using.

If an object is injected using Setter injection, which means it's optional and can be injected at any time, while dependencies which are injected using constructor injection means they are mandatory and must be supplied in the order they are declared.

6) Another tricky scenario with using DI is creating an object with too many dependencies and worse if those are injected using constructor injection.

That code becomes difficult to read. One solution to that problem is to use the Facade pattern and inject dependencies by encapsulating in another object. For example, you can introduce an object say `ApplicationSettings`, which can contain `DatabaseSetting`, `FileSetting`, and other configuration settings required by an

object. You can read more about Facade pattern in the **Design Pattern Library** course on Pluralsight, which is one of the best collection fo design patterns in one place.

7) You should use Dependency Injection Patterns to introduce loose coupling. Use Factory Patterns if you need to delegate the creation of objects. In short, dependency injection frees your application from factory pattern boilerplate code.

704.In Spring framework, what is the difference between FileSystemResource and ClassPathResource?

Build path concept

705.Name some popular Spring framework annotations that you use in your project?

706.How can you upload a file in Spring MVC Application?

707.What are the different types of events provided by Spring framework?

708.What is the difference between DispatcherServlet and ContextLoaderListener in Spring?

709.How will you handle exceptions in Spring MVC Framework?

Exception handler or controller advice annotation

710.What are the best practices of Spring Framework?

711. What is Spring Boot?

Hibernate

712.What is Hibernate framework?

713.What is an Object Relational Mapping (ORM)?

714.What is the purpose of Configuration Interface in Hibernate?

715.What is Object Relational Impedance Mismatch?

716.What are the main problems of Object Relational Impedance Mismatch?

717.What are the key characteristics of Hibernate?

718.Can you tell us about the core interfaces of Hibernate framework?

719.How will you map the columns of a DB table to the properties of a Java class in Hibernate?

720.Does Hibernate make it mandatory for a mapping file to have .hbm.xml extension?

721.What are the steps for creating a SessionFactory in Hibernate?

722.Why do we use POJO in Hibernate?

723.What is Hibernate Query Language (HQL)?

724.How will you call a stored procedure in Hibernate?

725.What is Criteria API in Hibernate?

726.Why do we use HibernateTemplate?

727.How can you see SQL code generated by Hibernate on console?

728.What are the different types of collections supported by Hibernate?

729.What is the difference between session.save() and session.saveOrUpdate() methods in Hibernate?

730.What are the advantages of Hibernate framework over JDBC?

731.How can we get statistics of a SessionFactory in Hibernate?

732.What is the Transient state of an object in Hibernate?

733.What is the Detached state of an object in Hibernate?

734.What is the use of Dirty Checking in Hibernate?

735.What is the purpose of Callback interface in Hibernate?

736.What are the different ORM levels in Hibernate?

737.What are the different ways to configure a Hibernate application?

738.What is Query Cache in Hibernate?

739.What are the different types of Association mappings supported by Hibernate?

740.What are the different types of Unidirectional Association mappings in Hibernate?

741.What is Unit of Work design pattern?

742.In Hibernate, how can an object go in Detached state?

743.How will you order the results returned by a Criteria in Hibernate?

744.How does Example criterion work in Hibernate?

745.How does Transaction management work in

Hibernate?

746.How can we mark an entity/collection as immutable in Hibernate?

747.What are the different options to retrieve an object from database in Hibernate?

748.How can we auto-generate primary key in Hibernate?

749.How will you re-attach an object in Detached state in Hibernate?

750.What is the first level of cache in Hibernate?

751.What are the different second level caches available in Hibernate?

752.Which is the default transaction factory in Hibernate?

753.What are the options to disable second level cache in Hibernate?

754.What are the different fetching strategies in Hibernate?

755.What is the difference between Immediate fetching and Lazy collection fetching?

756.What is 'Extra lazy fetching' in Hibernate?

757.How can we check is a collection is initialized or not under Lazy Initialization strategy?

758.What are the different strategies for cache mapping in Hibernate?

759.What is the difference between a Set and a Bag in Hibernate?

760.How can we monitor the performance of Hibernate in an application?

761.How can we check if an Object is in Persistent, Detached or Transient state in Hibernate?

762.What is 'the inverse side of association' in a mapping?

763.What is ORM metadata?

764.What is the difference between load() and get() method in Hibernate?

765.When should we use get() method or load() method in Hibernate?

766.What is a derived property in Hibernate?

767.How can we use Named Query in Hibernate?

768.What are the two locking strategies in

Hibernate?

769.What is the use of version number in Hibernate?

770.What is the use of session.lock() method in Hibernate?

771.What inheritance mapping strategies are supported by Hibernate?

Mavn

772.What is Maven?

773.What are the main features of Maven?

774.What areas of a Project can you manage by using Maven?

775.What are the main advantages of Maven?

776.Why do we say “Maven uses convention over configuration”?

777.What are the responsibilities of a Build tool like Maven?

778.What are the differences between Ant and Maven?

779.What is MOJO in Maven?

780.What is a Repository in Maven?

781.What are the different types of repositories in Maven?

782.What is a local repository in Maven?

783.What is a central repository in Maven?

784.What is a Remote repository in Maven?

785.Why we should not store jars in CVS or any other version control system instead of Maven repository?

786.Can anyone upload JARS or artifacts to Central Repository?

787.What is a POM?

788.What is Super POM?

789.What are the main required elements in POM file?

790.What are the phases in Build lifecycle in

Maven?

791.What command will you use to package your Maven project?

792.What is the format of fully qualified artifact name of a Maven project?

793.What is an Archetype in Maven?

794.What is the command in Maven to generate an Archetype?

795.What are the three main build lifecycles of Maven?

796.What are the main uses of a Maven plugin?

797.How will you find the version of a plugin being used?

798.What are the different types of profile in Maven? Where will you define these profiles?

799.What are the different setting files in Maven? Where will you find these files?

800.What are the main elements we can find in settings.xml?

801.How will you check the version of Maven in your system?

802.How will you verify if Maven is installed on Windows?

803.What is a Maven artifact?

804.What are the different dependency scopes in Maven?

805.How can we exclude a dependency in Maven?

806.How Maven searches for JAR corresponding to a dependency?

807.What is a transitive dependency in Maven?

808.What are Excluded dependencies in Maven?

809.What are Optional dependencies in Maven?

810.Where will you find the class files after compiling a Maven project successfully?

811. What are the default locations for source, test and build directories in Maven?

812.What is the result of jar:jar goal in Maven?

813.How can we get the debug or error messages from the

execution of Maven?

814.What is the difference between a Release version and SNAPSHOT version in Maven?

815.How will you run test classes in Maven?

816.Sometimes Maven compiles the test classes but doesn't run them? What could be the reason for it?

817.How can we skip the running of tests in Maven?

818.Can we create our own directory structure for a project in Maven?

819.What are the differences between Gradle and Maven?

820.What is the difference between Inheritance and Multi-module in Maven?

821.What is Build portability in Maven?

GIT

822.How can we see n most recent commits in GIT?

823.How can we know if a branch is already merged into master in GIT?

824.What is the purpose of git stash

drop?

825.What is the HEAD in GIT?

826.What is the most popular branching strategy in GIT?

827.What is SubGit?

828.What is the use of git instaweb?

829.What are git hooks?

830.What is GIT?

831.What is a repository in GIT?

832.What are the main benefits of GIT?

833.What are the disadvantages of GIT?

834.What are the main differences between GIT and SVN?

835.How will you start GIT for your project?

836.What is git clone in GIT?

837.How will you create a repository in GIT?

838.What are the different ways to start work in GIT?

839.GIT is written in which language?

840.What does ‘git pull’ command in GIT do internally?

841.What does ‘git push’ command in GIT do internally?

842.What is git stash?

843.What is the meaning of ‘stage’ in GIT?

844. What is the purpose of git config command?

845.How can we see the configuration settings of GIT installation?

846.How will you write a message with commit command in GIT?

847.What is stored inside a commit object

in GIT?

848.How many heads can you create in a GIT repository?

849.Why do we create branches in GIT?

850.What are the different kinds of branches that can be created in GIT?

851.How will you create a new branch in GIT?

852.How will you add a new feature to the main branch?

853.What is a pull request in GIT?

854.What is merge conflict in GIT?

855.How can we resolve a merge conflict in GIT?

856.What command will you use to delete a branch?

857.What command will you use to delete a branch that has unmerged changes?

858.What is the alternative command to merging in GIT?

859.What is Rebasing in GIT?

860.What is the ‘Golden Rule of Rebasing’ in GIT?

861.Why do we use Interactive Rebasing in place of Auto Rebasing?

862.What is the command for Rebasing in Git?

863.What is the main difference between git clone and git remote?

864.What is GIT version control?

865.What GUI do you use for working on GIT?

866.What is the use of git diff command in GIT?

867.What is git rerere?

868.What are the three most popular version of git diff command?

869.What is the use of git status command?

870.What is the main difference between git diff and git

status?

871.What is the use of git rm command in GIT?

872.What is the command to apply a stash?

873.Why do we use git log command?

874.Why do we need git add command in GIT?

875.Why do we use git reset command?

876.What does a commit object contain?

877.How can we convert git log messages to a different format?

878.What are the programming languages in which git hooks can be written?

879.What is a commit message in GIT?

880.How GIT protects the code in a repository?

881.How GIT provides flexibility in version control?

882.How can we change a commit message in GIT?

883.Why is it advisable to create an additional commit instead of amending an existing commit?

884.What is a bare repository in GIT?

885.How do we put a local repository on GitHub server?

886.How will you delete a branch in GIT?

887.How can we set up a Git repository to run code sanity checks and UAT tests just before a commit?

888.How can we revert a commit that was pushed earlier and is public now?

889.In GIT, how will you compress last n commits into a single commit?

890.How will you switch from one branch to a new branch in GIT?

891.How can we clean unwanted files from our working directory in GIT?

892.What is the purpose of git tag command?

893.What is cherry-pick in

GIT?

894.What is shortlog in GIT?

895.How can you find the names of files that were changed in a specific commit?

896.How can we attach an automated script to run on the event of a new commit by push command?

897.What is the difference between pre-receive, update and post-receive hooks in GIT?

898.Do we have to store Scripts for GIT hooks within same repository?

899.How can we determine the commit that is the source of a bug in GIT?

900.How can we see differences between two commits in GIT?

901.What are the different ways to identify a commit in GIT?

902.When we run git branch <branchname>, how does GIT know the SHA-1 of the last commit?

903.What are the different types of Tags you can create in GIT?

904.How can we rename a remote repository?

905.Some people use git checkout and some use git co for checkout. How is that possible?

906.How can we see the last commit on each of our branch in GIT?

907.Is origin a special branch in GIT?

908.How can we configure GIT to not ask for password every time?

909.What are the four major protocols used by GIT for data transfer?

910.What is GIT protocol?

911. How can we work on a project where we do not have push access?

912.What is git grep?

913.How can your reorder commits in GIT?

914.How will you split a commit into multiple commits?

915.What is filter-branch in GIT?

916.What are the three main trees maintained by

GIT?

917.What are the three main steps of working GIT?

918.What are ours and theirs merge options in GIT?

919.How can we ignore merge conflicts due to Whitespace?

920.What is git blame?

921.What is a submodule in GIT?

AWS

922.What do you know about AWS Region?

923.What are the important components of IAM?

924.What are the important points about AWS IAM?

925.What are the important features of Amazon S3?

926.What is the scale of durability in Amazon S3?

927.What are the Consistency levels supported by Amazon S3?

928.What are the different tiers in Amazon S3 storage?

929.How will you upload a file greater than 100 megabytes in Amazon S3?

930.What happens to an Object when we delete it from Amazon S3?

931.What is the use of Amazon Glacier?

932.Can we disable versioning on a version-enabled bucket in Amazon S3?

933.What are the use cases of Cross Region Replication Amazon S3?

934.Can we do Cross Region replication in Amazon S3 without enabling versioning on a bucket?

935.What are the different types of actions in Object Lifecycle Management in Amazon S3?

936.How do we get higher performance in our application by using Amazon CloudFront?

937.What is the mechanism behind Regional Edge Cache in Amazon CloudFront?

938.What are the benefits of Streaming

content?

939.What is Lambda@Edge in AWS?

940.What are the different types of events triggered by Amazon CloudFront?

941.What is Geo Targeting in Amazon CloudFront?

942.What are the main features of Amazon CloudFront?

943.What are the security mechanisms available in Amazon S3? Cloud Computing 944.What are the benefits of Cloud Computing?

945.What is On-demand computing in Cloud Computing?

946.What are the different layers of Cloud computing?

947.What resources are provided by Infrastructure as a Service (IAAS) provider?

948.What is the benefit of Platform as a Service?

949.What are the main advantages of PaaS?

950.What is the main disadvantage of PaaS?

951.What are the different deployment models in Cloud computing?

952.What is the difference between Scalability and Elasticity?

953.What is Software as a Service?

954.What are the different types of Datacenters in Cloud computing?

955.Explain the various modes of Software as a Service (SaaS) cloud environment?

956.What are the important things to care about in Security in a cloud environment?

957.Why do we use API in cloud computing environment?

958.What are the different areas of Security Management in cloud?

959.What are the main cost factors of cloud based data center?

960.How can we measure the cloud-based services?

961.How a traditional datacenter is different from a cloud environment?

962.How will you optimize availability of your application in

a Cloud environment?

963.What are the requirements for implementing IaaS strategy in Cloud?

DOCKER

964.What is Docker?

965.What is the difference between Docker image and Docker container?

966.How will you remove an image from Docker?

967.How is a Docker container different from a hypervisor?

968.Can we write compose file in json file instead of yaml?

969.Can we run multiple apps on one server with Docker?

970.What are the common use cases of Docker?

971.What are the main features of Docker-compose?

972.What is the most popular use of Docker?

973.What is the role of open source development in the

popularity of Docker?

UNIX Shell

974.How will you remove all files in current directory? Including the files that are two levels down in a sub-directory.

975.What is the difference between the –v and –x options in Bash shell scripts?

976.What is a Filter in Unix command?

977.What is Kernel in Unix operating system?

978.What is a Shell in Unix OS?

979.What are the different shells in Unix that you know about?

980.What is the first character of the output in ls –l command ?

981.What is the difference between Multi-tasking and Multi-user environment?

982.What is Command Substitution in Unix?

983.What is an Inode in Unix?

984.What is the difference between absolute path and

relative path in Unix file system?

985.What are the main responsibilities of a Unix Shell?

986.What is a Shell variable?

Microservices

987.What is a Microservice?

988.What are the benefits of Microservices architecture?

989.What is the role of architect in Microservices architecture?

990.What is the advantage of Microservices architecture over Service Oriented Architecture (SOA)?

991.Is it a good idea to provide a Tailored Service Template for Microservices development in an organization?

992.What are the disadvantages of using Shared libraries approach to decompose a monolith application?

993.What are the characteristics of a Good Microservice?

994.What is Bounded Context?

995.What are the points to remember during integration of Microservices?

996.Is it a good idea for Microservices to share a common database?

997.What is the preferred type of communication between Microservices? Synchronous or Asynchronous?

998.What is the difference between Orchestration and Choreography in Microservices architecture?

999.What are the issues in using REST over HTTP for Microservices?

1000. Can we create Microservices as State Machines?

ACKNOWLEDGMENT
S

We thank our readers who constantly send feedback and reviews to motivate us in creating these useful books with the latest information!

INTRODUCTIO

N

Java is one of the most popular programming language. There is a growing demand for Java Developer jobs in technology companies.

This book contains technical interview questions that an interviewer asks for Java technology and related topics like Spring, Hibernate, Maven, Git, Microservices, AWS etc.

Each question is accompanied with an answer so that you can prepare for job interview in short time.

We have compiled this list after attending dozens of technical interviews in top-notch companies like- Facebook, Oracle, Netflix, Amazon etc.

Once you go through them in the first pass, mark the questions that you could not answer by yourself. Then, in second pass go through only the difficult questions.

After going through this book 2-3 times, you will be well prepared to face a technical interview for a Java Developer position from Software Engineer level to Principal Engineer level.

All the
best!!

Java Interview Questions

Java Basics

1. What is the difference between JDK and JRE?

JDK stands for Java Development Kit. It contains the tools and libraries for development of Java programs. It also contains compilers and debuggers needed to compile Java program,

JRE stands for Java Runtime Environment. This is included in JDK. JRE provides libraries and JVM that is required to run a Java program.

2. What is Java Virtual Machine (JVM)?

Java Virtual Machine (JVM) is an abstract machine that

executes Java Bytecode. There are different JVM for different hardware and software platforms. So JVM is platform dependent. JVM is responsible for loading, verifying and executing the Bytecode on a platform.

3. What are the different types of memory areas allocated by JVM?

In java, JVM allocates memory to different processes, methods and objects. Some of the memory areas allocated by JVM are:

1. ClassLoader: It is a component of JVM used to load class

files. 2. Class (Method) Area: It stores per-class structures such as the runtime constant pool, field and method data, and the code for methods. 3. Heap: Heap is created a runtime and it contains the runtime

data area in which objects are allocated. 4. Stack: Stack stores local variables and partial results at runtime. It also helps in method invocation and return value. Each thread creates a private JVM stack at the time of thread creation. 5. Program Counter Register: This memory area contains the address of the Java virtual machine instruction that is currently being executed. 6. Native Method Stack: This area is reserved for all the

native methods used in the

application.

4. What is JIT compiler?

Just In Time compiler also known as JIT compiler is used for performance improvement in Java. It is enabled by default. It is compilation done at execution time rather earlier. Java has popularized the use of JIT compiler by including it in JVM.

5. How Java platform is different from other platforms?

Java is a platform independent language. Java compiler converts Java code in to byte code that can be interpreted by JVM. There are JVM written for almost all the popular platforms in the world.

Java byte code can run on any supported platform in same way. Where as other languages require libraries compiled for a specific platform to run.

6. Why people say that Java is 'write once and run anywhere' language?

You can write Java code on Windows and compile it in Windows platform. The class and jar files that you get from Windows platform can run as it is on Unix environment. So it is a truly platform independent language.

Behind all this portability is Java byte code. Byte code generated by Java compiler can be interpreted by any JVM. So it becomes much easier to write programs in Java and expect those to run on any platform.

Java compiler javac compiles java code and JVM java runs that code.

7. How does ClassLoader work in Java?

In Java, ClassLoader is a class that is used to load files in JVM. ClassLoader loads files from their physical file locations e.g. Filesystem, Network location etc.

There are three main types of ClassLoaders in Java.

1. Bootstrap ClassLoader: This is the first ClassLoader. It loads classes from rt.jar file. 2. Extension ClassLoader: It loads class files from jre/lib/ext location. 3. Application ClassLoader: This ClassLoader depends

on CLASSPATH to find the location of class files. If you specify your jars in CLASSPATH, then this ClassLoader will load them.

8. Do you think ‘main’ used for main method is a keyword in Java?

No, main is just a name of method. There can be multiple methods with same name main in a class file. It is not a keyword in Java.

9. Can we write main method as public void static instead of public static void?

No, you cannot write it like this. Any method has to first specify the modifiers and then the return value. The order of modifiers can change.

We can write static public void main() instead of public static void main().

10. In Java, if we do not specify any value for local variables,

then what will be the default value of the local variables?

Java does not initialize local variables with any default value. So these variables will be just null by default.

11. Let say, we run a java class without passing any arguments. What will be the value of String array of arguments in Main method?

By default, the value of String array of arguments is empty in Java. It is not null.

12. What is the difference between byte and char data types in Java?

Both byte and char are numeric data types in Java. They are used to represent numbers in a specific range.

Major difference between them is that a byte can store raw

binary data where as a char stores characters or text data.

Usage of char is E.g. char ch
= ‘x’;

Byte values range from -128 to
127.

A byte is made of 8 bits. But a char is made of 16 bits. So it is equivalent to 2 bytes.

OOPS

13. What are the main principles of Object Oriented Programming?

Main principles of Object Oriented Programming (OOPS) are:

1. Abstraction 2. Encapsulation 3. Inheritance 4. Polymorphism

14. What is the difference

between Object Oriented Programming language and Object Based Programming language?

Object Oriented Programming languages like Java and C++ follow concepts of OOPS like- Encapsulation, Abstraction, Polymorphism and Inheritance etc.

Object Based Programming languages follow some features of OOPS but they do not provide support for Polymorphism and Inheritance. Egg. JavaScript, VBScript etc.

Object Based Programming languages provide support for Objects and you can build objects from constructor. They languages also support Encapsulation. These are also known as Prototype-oriented languages.

15. In Java what is the default value of an object reference defined as an instance variable in an Object?

All the instance variable object references in Java are null.

16. Why do we need constructor in Java?

Java is an object-oriented language, in which we create and use objects. A constructor is a piece of code similar to a method. It is used to create an object and set the initial state of the object.

A constructor is a special function that has same name as class name.

Without a constructor, there is no other way to create an object.

By default, Java provides a default constructor for every object. If we overload a constructor then we have to implement default constructor.

17. Why do we need default constructor in Java classes?

Default constructor is the no-argument constructor that is automatically generated by Java if no other constructor is defined.

Java specification says that it will provide a default

constructor if there is no overloaded constructor in a class. But it does not say anything about the scenario in which we write an overloaded constructor in a class.

We need at least one constructor to create an object, that's why Java provides a default constructor.

When we have overloaded constructor, then Java assumes that we want some custom treatment in our code. Due to which it does not provide default constructor. But it needs default constructor as per the specification. So it gives error.

18. What is the value returned by Constructor in Java?

When we call a constructor in Java, it returns the object created by it. That is how we create new objects in Java.

19. Can we inherit a Constructor?

No, Java does not support inheritance of constructor.

20. Why constructors cannot be final, static, or abstract in Java?

If we set a method as final it means we do not want any class to override it. But the constructor (as per Java Language Specification) cannot be overridden. So there is no use of marking it final.

If we set a method as abstract it means that it has no body and it should be implemented in a child class. But the constructor is called implicitly when the new keyword is used. Therefore it needs a body.

If we set a method as static it means that it belongs to the class, but not a particular object. The constructor is always called to initialize an object. Therefore, there is no use of marking constructor static.

Inheritanc
e