1

1. What is Java?

1

Java is a programming language and computing platform first released in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacentres, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

2

2. Mention some features of Java?

2

Object-Oriented: Java is a object oriented programming language. Everything in Java is an Object. Portable: Java run time environment uses a bytecode verification process to make sure that code loaded over the network doesn ' t violate Java security constraints.
• Platform independent: Java is platform independent. Java is a write once, run anywhere language. Without any modifications, we can use a program in different platforms.
• Secured: Java is well known for its security. It delivers virus free systems.
• High Performance: Java enables high performance with the use of JIT (JustIn-Time) compilers • Multithreaded: Java Multithreaded features allows us to write programs that can perform many tasks simultaneously. Multithreading concept of Java shares a common memory area. It doesn ' t occupy memory for each thread.

3

3. What is the difference between Declaration and Definition in Java?

3

Declaration: If you just declare a class or method/function or variable without mentioning anything about what that class or method/function or variable looks like is called as declaration in Java.
Definition: If you define how a class or method/function or variable is implemented then it is called definition in Java. When we create an interface or abstract class, we simply declare a method/function but not define it.

4

4. What is an Object in Java?

4

An object is an instance of a class. Objects have state (variables) and behavior (methods). Example: A dog is an object of Animal class. The dog has its states such as color, name, breed, and behaviors such as barking, eating, wagging her tail.

5

5. What is a Class in Java?

5

A class can be defined as a collection of objects. It is the blueprint or template that describes the state and behavior of an object.

6

6. What is Constructor in Java?

6

Constructor in Java is used in the creation of an Object that is an instance of a Class. Constructor name should be same as class name. It looks like a method but it ' s not a method. It won ' t return any value. We have seen that methods may return a value. If there is no constructor in a class, then compiler automatically creates a default constructor.

7

7. What is Local Variable and Instance Variable?

7

Local Variable: Local variable is a variable which we declare inside a Method. A method will often store its temporary state in local variables. Instance Variable (Non-static): Instance variable is a variable which is declared inside a Class but outside a Method. We don ' t declare this variable as Static because these variables are non-static variables.

8

8. What are the OOPs concepts?

8

OOPS Stands for Object Oriented Programming System. It includes Abstraction, Encapsulation, Inheritance, Polymorphism, Interface etc.,

9

9. What is Inheritance in Java?

9

Inheritance is a process where one class inherits the properties of another class.

10

10. What is Polymorphism?

10

Polymorphism allows us to perform a task in multiple ways. Let ' s break the word Polymorphism and see it, ' Poly ' means ' Many ' and ' Morphos ' means ' Shapes ' . Assume we have four students and we asked them to draw a shape. All the four may draw different shapes like Circle, Triangle, and Rectangle.

11

11. What are the types of Polymorphism?

11

There are two types of Polymorphism in Java Page 5 of 18 1. Compile time polymorphism (Static binding) – Method overloading 2. Runtime polymorphism (Dynamic binding) – Method overriding We can perform polymorphism by ' Method Overloading ' and ' Method Overriding '

12

12. What is Method Overloading?

12

A class having multiple methods with same name but different parameters is called Method Overloading There are three ways to overload a method.

- Parameters with different data types
- Parameters with different sequence of a data types
- Different number of parameters

13

13. What is Method Overriding?

13

Declaring a method in child class which is already present in the parent class is called Method Overriding. In simple words, overriding means to override the functionality of an existing method. In this case, if we call the method with child class object, then the child class method is called. To call the parent class method we have to use super keyword.

14

14. What is Abstraction in Java?

14

Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users. Example: Mobile Phone. A layman who is using mobile phone doesn ' t know how it works internally but he can make phone calls.

15

15. What is Abstract Class in Java?

15

We can easily identify whether a class is an abstract class or not. A class which contains abstract keyword in its declaration then it is an Abstract Class. Points to remember:

- Abstract classes may or may not include abstract methods
- If a class is declared abstract then it cannot be instantiated.
- If a class has abstract method then we have to declare the class as abstract class
- When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class.

However, if it does not, then the subclass must also be declared abstract.

16

16. What is Abstract Method?

16

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this: In order to use an abstract method, you need to override that method in sub class.

17

17. What is Interface in Java?

17

An interface in Java looks similar to a class but both the interface and class are two different concepts. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract. We can achieve 100% abstraction and multiple inheritance in Java with Interface.

18

18. What is Encapsulation in Java?

18

Encapsulation is a mechanism of binding code and data together in a single unit. Let ' s take an example of Capsule. encapsulated inside a capsule. Likewise in encapsulation, all the methods and variables are wrapped together in a single class.

19

21. What are the different access modifiers available in Java?

19

Access modifiers are subdivided into four types such as Default, Public, Private, Protected default: The scope of default access modifier is limited to the package only. If we do not mention any access modifier, then it acts like a default access modifier. private: The scope of private access modifier is only within the classes. Note: Class or Interface cannot be declared as private protected: The scope of protected access modifier is within a package and also outside the package through inheritance only. public: The scope of public access modifier is everywhere. It has no restrictions. Data members, methods and classes that declared public can be accessed from anywhere.

20

22. Difference between static binding and dynamic binding?

20

1. Static binding is also known as early binding whereas dynamic binding is also known as late binding.
2. Determining the type of an object at compile time is Static binding whereas determining the type of an object at run time is dynamic binding
3. Java uses static binding for overloaded methods and dynamic binding for overridden methods.

21

24. What is Multiple Inheritance?

21

If a class implements multiple interfaces, or an interface extends multiple interfaces then it is known as multiple inheritance. We will update this post “ Java Interview Questions For Selenium Testers ” ASAP. Keep visiting. If you like this post, share it with your friends.

22

25.What are the Java IDE ’ s?

22

Ans: Eclipse and NetBeans are the IDE's of JAVA.

23

26.What is mean by Exception?

23

An Exception is a problem that can occur during the normal flow of an execution. A method can throw an exception when something wails at runtime. If that exception couldn ’ t be handled, then the execution gets terminated before it completes the task. If we handled the exception, then the normal flow gets continued. Exceptions are a subclass of java.lang.Exception.

24

27. What are the types of Exceptions?

24

Checked Exception: These exceptions are checked by the compiler at the time of compilation. Classes that extend Throwable class except Runtime exception and Error are called checked Exception. Checked Exceptions must either declare the exception using throes keyword (or) surrounded by appropriate try/catch. E.g. ClassNotFound Exception Unchecked Exception: These exceptions are not checked during the compile time by the compiler. The compiler doesn ’ t force to handle these exceptions. It includes: • Arithmetic Exception • ArrayIndexOutOfBounds Exception

25

28. What are the different ways to handle exceptions?

25

Two different ways to handle exception are explained below:
#1) Using try/catch: A risky code is surrounded by try block. If an exception occurs, then it is caught by the catch block which is followed by the try block.
#2) By declaring throws keyword: At the end of the method, we can declare the exception using throws keyword.

26

29.What are the Advantages of Exception handling?

26

Given below are the advantages: • The normal flow of the execution won ' t be terminated if exception got handled • We can identify the problem by using catch declaration

27

30. What are Exception handling keywords in Java?

27

Given below are the two Exception Handling Keywords: try: When a risky code is surrounded by a try block. An exception occurring in the try block is caught by a catch block. Try can be followed either by catch (or) finally (or) both. But any one of the blocks is mandatory. catch: This is followed by try block. Exceptions are caught here. finally: This is followed either by try block (or) catch block. This block gets executed regardless of an exception. So generally clean up codes are provided here.

28

31.What is the final keyword in Java?

28

Final variable: Once a variable is declared as final, then the value of the variable could not be changed. It is like a constant. Example: final int = 12; Final method: A final keyword in a method that couldn ' t be overridden. If a method is marked as a final, then it can ' t be overridden by the subclass. Final class: If a class is declared as final, then the class couldn ' t be subclassed. No class can extend the final class.

29

32. What is the use of static keyword?

29

static is a non-access modifier in Java which is applicable for the following:

- blocks
- variables
- methods
- nested classes

30

32.What is a Thread?

30

In Java, the flow of a execution is called Thread. Every java program has at least one thread called main thread, the Main thread is created by JVM. The user can define their own threads by extending Thread class (or) by implementing Runnable interface. Threads are executed concurrently.

31

33. How do you make a thread in Java?

31

There are two ways available in order to make a thread. #1) Extend Thread class: Extending a Thread class and override the run method. The thread is available in java.lang.thread. The disadvantage of using a thread class is that we cannot extend any other classes because we have already extend the thread class. We can overload the run () method in our class. #2) Implement Runnable interface: Another way is implementing the runnable interface. For that we should provide the implementation for run () method which is defined in the interface.

32

34.What does yield method of the Thread class do?

32

A yield () method moves the currently running thread to a runnable state and allows the other threads for execution. So that equal priority threads have a chance to run. It is a static method. It doesn ' t release any lock. Yield () method moves the thread back to the Runnable state only, and not the thread to sleep (), wait () (or) block.

33

35.Explain about wait () method.

33

wait () method is used to make the thread to wait in the waiting pool. When a wait () method is executed during a thread execution then immediately the thread gives up the lock on the object and goes to the waiting pool. Wait () method tells the thread to wait for a given amount of time. Then the thread will wake up after notify () (or) notify all () method is called. Wait() and the other above-mentioned methods do not give the lock on the object immediately until the currently executing thread completes the synchronized code. It is mostly used in synchronization.

34

37.How to stop a thread in java?

34

We can stop a thread by using the following thread methods. • Sleeping • Waiting • Blocked Sleep: Sleep () method is used to sleep the currently executing thread for the given amount of time. Once the thread is wake up it can move to the runnable state. So sleep () method is used to delay the execution for some period. It is a static method. Example: Thread. Sleep (2000) So it delays the thread to sleep 2 milliseconds. Sleep () method throws an uninterrupted

35

43.What is meant by Serialization?

35

Converting a file into a byte stream is known as Serialization. The objects in the file is converted to the bytes for security purposes. For this, we need to implement java.io.Serializable interface. It has no method to define. Variables that are marked as transient will not be a part of the serialization. So we can skip the serialization for the variables in the file by using a transient keyword.

36

49. What is the difference between String and StringBuffer?
What is mutable and immutable?

36

The most important difference between String and StringBuffer in java is that String object is immutable whereas StringBuffer object is mutable. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed. By immutable, we mean that the value stored in the String object cannot be changed. For example we cannot reverse string directly, only through using StringBuffer class. mutability String is immutability class it means once we are creating String objects it is not possible to perform modifications on existing object. (String object is fixed object) StringBuffer is a mutability class it means once we are creating StringBuffer objects on that existing object it is possible to perform modification.

37

50.What are the collections you have used?

37

Mostly in my current project we use ArrayList (There are other collections Set and Maps - just for your awareness and you can mention that you don't use them in your project). If you know it please explain to the interviewer.

38

51.What are arrays and list? Difference between them?

38

1. Arrays are fixed in size but ArrayLists are dynamic in size.
2. Arrays can store homogeneous elements whereas collections can store both.
Example: in Array we can store either int or String or boolean whereas in Array list we can store all of them together 3. To find the size on an Array we use ArrayName.length and for arrayList we use ArrayListName.size()
Example: in Array we can store either int or String or boolean whereas in Array list we can store all of them together 3. To find the size on an Array we use ArrayName.length and for arrayList we use ArrayListName.size()

39

52.How can we access variable without creating an object instance of it? variables and how you use it?

39

By declaring variable as a static we can access it from different class - those variables called class variables and also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block. Whereas, Instance variables are declared in a class, but outside a method, constructor or any block.

40

53. What is the difference between throw and throws?

40

Throws : • is used to declare an exception, which means it works similar to the try-catch block.
• is used in method declaration.
• is followed by exception class names.
• you can declare multiple exception with throws
• throws declare at method it might throws Exception
• used to handover the responsibility of handling the exception occurred in the method to the caller method.

41

54.What is the difference between interface and a class? Example from your framework?

41

Class : • Class will contain concrete methods
• Class is extended
• We can create an Object of the class
• Class can inherit only one Class and can implement many interfaces
Interface : • Interface will have Interface keyword.
• Interface will contain only abstract methods
• We cannot create object of interface
• Interface needs to be implemented
• Class can extends many interfaces
• We need to provide implementation to all methods when we implement interface to the class

42

55.What is singleton and have used singleton concept in your project ?

42

I know what is singleton class, however in my current project I do not use this concept. (A singleton class is a class that can have only one object (an instance of the class) at a time.)

43

56.Can we achieve 100% abstraction in JAVA? Can we achieve 100% abstraction in JAVA with use of the interfaces?

43

We cannot achieve 100% abstraction in JAVA unless we use Interfaces

44

57.What is the Difference between final, finally?

44

Final keyword: • Used to declare constant values.
The variable declared as final should be initialized only once and cannot be changed.
• Used to prevent inheritance. Java classes declared as final cannot be extended.
• Used to prevent method overriding. Methods declared as final cannot be overridden.

58. In an arraylist which has the value {1,2,3} how to change the value 3 to 30?

45

Use the set method to replace the old value with a new one.
list.set(3, 30);

59. What is Static class -give examples?

46

You cannot use static with outer class.
Static classes are basically a way of grouping classes together in Java. Java doesn't allow you to create top-level static classes; only nested (inner) static classes.

```
public class CarParts {
public static class Wheel {
public Wheel() {
System.out.println("Wheel created!");
}
}
public CarParts() {
System.out.println("Car Parts object created!");
}
}
```


60. How to get value from an arraylist?

47

ArrayList get(int index) method is used for fetching an element from the list. We need to specify the index while calling get method and it returns the value present at the specified index.
public Element get(int index);
System.out.println("First element of the ArrayList: "+al.get(0));

61. What is root in Java

48

The Object class of the java.lang package is the root class in Java i.e. It is the super class of every user-defined/predefined class n Java. All objects, including arrays, implement the methods of this class.
The reason for this is to have common functionalities such as synchronization, garbage collection, collection support, object cloning for all objects in Java.

49

62. Explain Flow of execution constructor, main method, static method?

49

First main method, second static method, third constructer will execute.

50

63. Is it possible to overload a final method?

50

Yes. But override is not possible.

51

64. Can we create object for an abstract class?

51

No, not possible to instantiate abstract class.

52

65. Why multiple inheritance with the help of classes is not supported in java/ the reason behind?

52

The problem with multiple inheritance is that two classes may define different ways of doing the same thing, and the subclass can't choose which one to pick.

53

66. A is an abstract class, B is concrete class, B extends A. how we could keep B as concrete class?

53

If B extends A abstract class, you must complete the overridden methods.

54

67. Can we create object for an abstract class?

54

No. not possible.

55

68. What gives Java its 'write once and run anywhere' nature?

55

The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer.

56

69. IS DELETE, NEXT, MAIN, EXIT or NULL keyword in java?

56

No

57

70. What if I write static public void instead of public static void?

57

The program compiles and runs correctly because the order of specifiers doesn't matter in Java.

58

71. What is the default value of the local variables?

58

The local variables are not initialized to any default value, neither primitives nor object references.

59

72. What will be the initial value of an object reference which is defined as an instance variable?

59

All object references are initialized to null in Java.

60

73. Is constructor inherited?

60

No, The constructor is not inherited.

61

74. Can you make a constructor final?

61

No, the constructor can't be final.

62

75. What are the restrictions that are applied to the Java static methods?

62

Two main restrictions are applied to the static methods.

- The static method can not use non-static data member or call the non-static method directly.
- this and super cannot be used in static context as they are non-static.

63

76. Can we override the static methods?

63

No, we can't override static methods.

64

78. Can we execute a program without main() method?

64

Yes, one of the ways to execute the program without the main method is using static block.

65

79. What if the static modifier is removed from the signature of the main method?

65

Program compiles. However, at runtime, It throws an error "NoSuchMethodError."

66

80. Can we make the abstract methods static in Java?

66

In Java, if we make the abstract methods static, It will become the part of the class, and we can directly call it which is unnecessary. Calling an undefined method is completely useless therefore it is not allowed.

67

81. Can we make constructors static?

67

As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.

68

82. Can we declare the static variables and methods in an abstract class?

68

Yes, we can declare static variables and methods in an abstract method. As we know that there is no requirement to make the object to access the static context, therefore, we can access the static context declared inside the abstract class by using the name of the abstract class.

69

83. What are the main uses of keyword -THIS ?

69

There are the following uses of this keyword.

- this can be used to refer to the current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke the current class constructor.
- this can be passed as an argument in the method call.

70

84. Can we assign the reference to this variable?

70

No, this cannot be assigned to any value because it always points to the current class object and this is the final reference in Java.

71

85. Can this keyword be used to refer static members?

71

Yes, It is possible to use this keyword to refer static members because this is just a reference variable which refers to the current class object. However, as we know that, it is unnecessary to access static variables through objects, therefore, it is not the best practice to use this to refer static members.

72

86. How can constructor chaining be done using this keyword?

72

Constructor chaining enables us to call one constructor from another constructor of the class with respect to the current class object. We can use this keyword to perform constructor chaining within the same class. Consider the following example which illustrates how can we use this keyword to achieve constructor chaining.

73

87. Which class is the superclass for all the classes?

73

The object class is the superclass of all other classes in Java.

74

88. Why is multiple inheritance not supported in java?

74

To reduce the complexity and simplify the language, multiple inheritance is not supported in java. Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

75

89. Why does Java not support pointers?

75

The pointer is a variable that refers to the memory address. They are not used in Java because they are unsafe(unsecured) and complex to understand.

76

90. What are the main uses of the super keyword?

76

There are the following uses of super keyword.

- super can be used to refer to the immediate parent class instance variable.
- super can be used to invoke the immediate parent class method.

77

91. Can you use this() and super() both in a constructor?

77

No, because this() and super() must be the first statement in the class constructor.

78

92. Can we overload the main() method?

78

Yes, we can have any number of main methods in a Java program by using method overloading.

79

93. Can we override the static method?

79

No, you can't override the static method because they are the part of the class, not the object.

80

94. Why can we not override static method?

80

It is because the static method is the part of the class, and it is bound with class whereas instance method is bound with the object, and static gets memory in class area, and instance gets memory in a heap.

81

95. Can we override the overloaded method?

81

Yes.

82

96. Can we override the private methods?

82

No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.

83

97. Can you have virtual functions in Java?

83

Yes, all functions in Java are virtual by default.

84

98. What is the final class?

84

If we make any class final, we can't inherit it into any of the subclasses.

85

99. Can we initialize the final blank variable?

85

Yes, if it is not static, we can initialize it in the constructor. If it is static blank final variable, it can be initialized only in the static block.

86

100. Can you declare the main method as final?

86

Yes, We can declare the main method as public static final void main(String[] args){}.

87

101. Can we declare a constructor as final?

87

The constructor can never be declared as final because it is never inherited. Constructors are not ordinary methods; therefore, there is no sense to declare constructors as final. However, if you try to do so, The compiler will throw an error.

88

102. Can we declare an interface as final?

88

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.

89

103. What is the difference between the final method and abstract method?

89

The main difference between the final method and abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition.

90

104. Can you achieve Runtime Polymorphism by data members?

90

No, because method overriding is used to achieve runtime polymorphism and data members cannot be overridden. We can override the member functions but not the data members.

91

105. What is Java instanceOf operator?

91

The instanceof in Java is also known as type comparison operator because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has a null value, it returns false.

92

106. Can there be an abstract method without an abstract class?

92

No, if there is an abstract method in a class, that class must be abstract.

93

106. Can there be an abstract method without an abstract class?

93

No, if there is an abstract method in a class, that class must be abstract.

94

107. Can you use abstract and final both with a method?

94

No, because we need to override the abstract method to provide its implementation, whereas we can't override the final method.

95

108. Is it possible to instantiate the abstract class?

95

No, the abstract class can never be instantiated even if it contains a constructor and all of its methods are implemented.

96

109. Can you declare an interface method static?

96

No, because methods of an interface are abstract by default, and we can not use static and abstract together.

97

110. Can the Interface be final?

97

No, because an interface needs to be implemented by the other class and if it is final, it can't be implemented by any class.

98

111. What is a marker interface?

98

A Marker interface can be defined as the interface which has no data member and member functions. For example, Serializable, Cloneable are marker interfaces.

99

112. Can we define private and protected modifiers for the members in interfaces?

99

No, they are implicitly public.

100

113. When can an object reference be cast to an interface reference?

100

An object reference can be cast to an interface reference when the object implements the referenced interface.

101

114. What is the static import?

101

By static import, we can access the static members of a class directly, and there is no to qualify it with the class name.

102

115. What is the base class for Error and Exception?

102

The Throwable class is the base class for Error and Exception.

103

116. Can finally block be used without a catch?

103

Yes, According to the definition of finally block, it must be followed by a try or catch block, therefore, we can use try block instead of catch. More details.

104

117. Is there any case when finally will not be executed?

104

Finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).More details.

105

118. Can subclass overriding method declare an exception if parent class method doesn't throw an exception?

105

Yes but only unchecked exception not checked.

106

120. Why are the objects immutable in java?

106

Because Java uses the concept of the string literal. Suppose there are five reference variables, all refer to one object "sachin". If one reference variable changes the value of the object, it will be affected by all the reference variables. That is why string objects are immutable in java.

107

121. How many ways can we create the string object?

107

1) String Literal
Java String literal is created by using double quotes. For Example:
String s="welcome";
Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. String objects are stored in a special memory area known as the string constant pool For example:
String s1="Welcome";
String s2="Welcome";//It doesn't create a new instance
2) By new keyword
String s=new String("Welcome");//creates two objects and one reference variable

108

122. How many objects will be created in the following code?

108

String s1="Welcome";
String s2="Welcome";
String s3="Welcome";
Only one object will be created using the above code because strings in Java are immutable.

109

123. Why java uses the concept of the string literal?

109

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

110

125. What is a nested class?

110

The nested class can be defined as the class which is defined inside another class or interface. We use the nested class to logically group classes and interfaces in one place so that it can be more readable and maintainable. A nested class can access all the data members of the outer class including private data members and methods. The syntax of the nested class is defined below.

```
class Java_Outer_class{
//code
class Java_Nested_class{
//code
}
}
```

111

126. What are the disadvantages of using inner classes?

111

There are the following main disadvantages of using inner classes.

Inner classes increase the total number of classes used by the developer and therefore increases the workload of JVM since it has to perform some routine operations for those extra classes which result in slower performance.

IDEs provide less support to the inner classes as compare to the top level classes and therefore it annoys the developers while working with inner classes.

112

127. Is there any difference between nested classes and inner classes?

112

Yes, inner classes are non-static nested classes. In other words, we can say that inner classes are the part of nested classes.

113

128. Can we access the non-final local variable, inside the local inner class?

113

No, the local variable must be constant if you want to access it in the local inner class.

114

129. Can a class have an interface?

114

Yes, an interface can be defined within the class. It is called a nested interface.

115

130. Can an Interface have a class?

115

Yes, they are static implicitly.

116

131. How is garbage collection controlled?

116

Garbage collection is managed by JVM. It is performed when there is not enough space in the memory and memory is running low. We can externally call the System.gc() for the garbage collection. However, it depends upon the JVM whether to perform it or not.

133. What kind of thread is the Garbage collector thread?

117

Daemon thread.

134. What is serialization?

118

Serialization in Java is a mechanism of writing the state of an object into a byte stream. It is used primarily in Hibernate, RMI, JPA, EJB and JMS technologies. It is mainly used to travel object's state on the network (which is known as marshaling). Serializable interface is used to perform serialization. It is helpful when you require to save the state of a program to storage such as the file. At a later point of time, the content of this file can be restored using deserialization. It is also required to implement RMI(Remote Method Invocation). With the help of RMI, it is possible to invoke the method of a Java object on one machine to another machine.

141. What is the difference between String and StringBuffer? What is mutable and immutable?

119

The most important difference between String and StringBuffer in java is that String object is
immutable whereas StringBuffer object is mutable.
The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.
By immutable, we mean that the value stored in the String object cannot be changed.
For example we cannot reverse string directly, only through using StringBuffer class.
immutability vs. mutability
String is immutability class it means once we are creating String objects it is not possible to
perform modifications on existing object. (String object is fixed object)
StringBuffer is a mutability class it means once we are creating StringBuffer objects on that
existing object it is possible to perform modification.

38. When to use Runnable interface Vs Thread class in Java?

120

Ans: If we need our class to extend some other classes other than the thread then we can go with the runnable interface because in java we can extend only one class. If we are not going to extend any class then we can extend the thread class.

121

39.Difference between start() and run() method of thread class.

121

Ans: Start() method creates new thread and the code inside the run () method is executed in the new thread. If we directly called the run() method then a new thread is not created and the currently executing thread will continue to execute the run() method.

122

40.What is Multi-threading?

122

Multiple threads are executed simultaneously. Each thread starts their own stack based on the flow (or) priority of the threads. On the 1st line execution, JVM calls the main method and the main thread stack looks as shown below.

123

41.What is Synchronization?

123

Synchronization makes only one thread to access a block of code at a time. If multiple thread accesses the block of code, then there is a chance for inaccurate results at the end. To avoid this issue, we can provide synchronization for the sensitive block of codes. The synchronized keyword means that a thread needs a key in order to access the synchronized code. Locks are per objects. Every Java object has a lock. A lock has only one key. A thread can access a synchronized method only if the thread can get the key to the objects lock. For this, we use " Synchronized " keyword.

124

42. What is the disadvantage of Synchronization?

124

Synchronization is not recommended to implement all the methods. Because if one thread accesses the synchronized code then the next thread should have to wait. So it makes slow performance on the other end.