JAVA

1. JDK (java development kit)

- **JDK** is a software development environment which is used to develop java app.
- It includes JRE, Interpreter, Compiler, achiever (jars), documentation (javadoc).

2. JRE (java runtime environment)

- JRE is a set of software tools which is used to developing java application.
- It is a part of **JDK**.
- We can download **JRE** separately also.
- It combines the **JVM**, platform classes and support libraries.

3. JVM (java virtual machine)

- JVM is a run time environment which is used to execute the java byte code.
- It is an abstract machine
- It is Platform dependent.
- It contains Virtual processor, interpreter, and JIT Compiler.
- It loads the Class loader subsystem and create the Runtime Data Area.
  https://dzone.com/articles/jvm-architecture-explained

4. JIT Compiler (just in time compiler)

- JIT is a component of java run time environment (JRE) which is used to improve the Performance of application.
- It compiles the java byte code into native machine code at run time.
  https://www.ibm.com/support/knowledgecenter/en/SSYKE2_8.0.0/com.ibm.java.vm.80.doc/docs/jit_overview.html

5. Difference between interpreter and compiler

https://www.programiz.com/article/difference-compiler-interpreter

6. Difference between Heap and Stack

https://www.edureka.co/blog/interview-questions/java-interview-questions/

7. Variables

- Local Variable------inside method or block
- Instance Variable-----inside class outside the block
- Static variable --- variable is declared with static key word

8. Transient Variables

- Transient Variable declared with transient key word
- By Default all object variables are converted into persistence state. In some cases we need to avoid persisting state. So at this time we can declare the variable as transient then it will not be persisted.
- Persistence State means object is stored in persistence storage (file).

9. Serialization

- It is the process of making the object state as persistent. That means the state of the object is converted into stream of bytes and stored in a file.
- It is mostly used in the networking programming. The objects that need to be transmitted through the network have to be converted in to bytes. For that purpose every class or interface must be implements serializable interface.
- It is a marker interface without any methods.

10. Volatile variable

- It is a Keyword
- It is used as an indicator to java compiler and thread that do not cache value of this variable and always read from main memory.
- It is Thread safe.
  http://tutorials.jenkov.com/java-concurrency/volatile.html

11. Object Class

- Object class is belongs to java.lang.package.
- It is the super class of all other classes in java
- Every class is must be extends with object class either directly or indirectly.

12. Non Static methods Object Class

- Clone() , toString() , equals() , finalize() , getClass() , wait() , notify() ,notifyAll()

13. Access Modifiers

- Private ---- access within the class
- Protected -- access within package but inherited the subclass of outer package.
- Public --- we can access any where
- default --- access with in the package
  http://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/

14. Non-Access Modifiers

- static – it is used to specify whether it is class member or instance member
- final – restrict the further modifications
- abstract --- it is used to enhance the member furthered.
  http://javaconceptoftheday.com/java-interview-questions-on-modifiers/

15. String

- String is a class. It represents the sequence of char values.
- It is immutable that means it cannot be changed once initialized. Whenever we are change the value it was creating the new instance.
- If you want to achieve mutable then we go for String Buffer and String Builder.
- **compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc. are the string methods**

16. String Buffer

- String buffer is a class which is used to create mutable strings
- Mutable means it can be modified based on our requirement
- It is faster than String
- It is thread-safe because multiple threads cannot access simultaneously.
- It is Synchronized
- **append(), insert(), replace(), delete(), reverse(), capacity(), ensure capacity() etc...**

17. String Builder

- String builder is a class which is used to create mutable strings
- It is also same like String buffer but it is non synchronized
- It is faster than string buffer
- It is not thread-safe
- **append(), insert(), replace(), delete(), reverse(), substring(), ensure capacity() etc..**

18. Thread

- In multi task program contains two or more parts can execute concurrently. Each part of such executed program is called thread
- Thread is defined in two ways

  1) By extend the Thread class

  2) To implements the runnable interface

19. Thread Life-Cycle

- **New** – Thread is instantiated but start() is not called on thread instance
- **Runnable –** when start() is called .But not pick thread scheduler for execution
- **Running –** When it is executing. The thread scheduler picks the thread from the thread pool and executes it.
- **Waiting, blocked, sleeping –** it is not in the running pool. May be in sleep mode or waiting for another thread to send notification.
- **Dead –** when run() is executed

  http://www.interviewgrid.com/interview_questions/java/java_threads

20. Liveness problems

- **Deadlock** – where 2 or more threads blocked forever, waiting for each other
- **Starvation** –- unable to get access to shared resources
- **Livelock** – thread acts in response to another thread and other one is response to another thread response.

  http://www.learncertification.com/study-material/liveness-in-java-291

  http://www.interviewgrid.com/interview_questions/java/java_threads

21. Synchronization

- Synchronization is a capability to control the access of multiple threads to any shared resources
- It is better option when we allow only one thread at a time
- Performance is low.
- 2 types of synchronizations are there 1) process synchronization , 2) Thread Synchronization

  https://www.javatpoint.com/synchronization-in-java

22. Thread Pool

- It represents a group of working threads that are waiting for the job and reuse many times.
- In this case group of fixed size threads are created
- A Thread from the thread pool is pulled out and assigned a job by the service provider. After completion of the job thread is contained in the thread pool again.
- It gives the good performance .because there is no need to create new thread.

23. Thread scheduler

- It is the part of **JVM** that decides the which thread should run
- There is no guarantee that which runnable thread will be chooses to run by the thread scheduler.
- Only one thread can run in a single process.
- It uses the preemptive scheduling or time slicing scheduling to schedule threads.

24. Preemptive scheduling

- It ensures the highest priority thread is executed until thread is waiting or dead state.
- If a certain task is running and the scheduling method is preemptive, and if there is another task with high priority than the executing task, then executing task is preempted by high priority task.
- It follows the Round-robin fixed priority preemptive scheduling algorithm.

25. Time slicing scheduling

- Here thread is executed for a predefined slice of time and then re-enters in to the pool of ready threads
- At this time the scheduler determines the executable task based on the priority and various other tasks.
- Here time slice scheduling method, a task executes for predefined slice of time. After the execution of that task, if there is another task with higher priority, then the scheduler executes the task otherwise executes the task depending on the priority and other factors.

26. Daemon Thread

- It is a thread that executes at low priority **EX: GARBAGE COLLECTION**

27.OOPS

- It is a methodology based on the concept of objects which contains the data in the form of fields.
- Java is class based object oriented programming language. That means objects in java are instances of classes.
- It is based on mainly 4 principles those are :

  Encapsulation

  Abstraction

  Inheritance

  Polymorphism

28.Abstraction

- Hiding the internal implementation details and just highlight the set of services what we are offering is abstraction
- We can achieve this by using **abstract class (0 to 100%)**, and **interface (100%)**.
- Abstract key word is applicable for classes and methods only
- Constructor, static initialization block, instance initialization block and variables are cannot be abstract

29.Abstract Class

- The class which is declared with abstract key word is abstract class.
- It may or may not be contains abstract class
- Abstract classes cannot be instantiated. But they can be sub classed.
- Abstract class may have a static fields and static methods.
- When we inherit the abstract class we must provide the implementation for abstract methods which are present in parent class otherwise we need provide this class as abstract.
- Abstract class constructor will be executed at the time of creating child class object

  https://javaconceptoftheday.com/java-interview-questions-on-abstract-class

30.Abstract Method

- The method which is declared with abstract key word is abstract method.
- Abstract method don't have a implementation, it cannot be static
- **Ex: abstract void sum(int a , int b);**

- It cannot be static

31. Interface

- Interface is a contract between client an service provider
- Every Service requirement is considered as interface
- From the client point of view interface is set of services what we are expecting and service provider point of view interface is set of services what is offering
- We can achieve the security using interface because we are not highlighting the internal implementation.
- It supports the multiple inheritance functionality.
- It contains only abstract methods and by default every method in interface is public and abstract
- We can achieve loosely coupling by using interface
- A class can extend only one class at a time but an interface can extend any number of interface at a time
- Variables in interface are by default public, static and final whether we are declaring or not.
- Variables should be initialized.
- We don’t know anything about implementation, you just have a requirement specification then we should go for Interface

32. Marker Interface

- Marker Interface is also like an interface but there are no fields or methods.
- It is used to convey the JVM that the class implemented this interface have some special behaviors

Ex: Clonable Interface
Serializable interface
Remote interface
Thread safe interface

33. Serializable Interface

- By implementing this Interface we can send the objects across the network and we can save the state of object as file.

34. Clonable Interface

- It is used to provide exactly duplicate object for our own object position

35. Interface vs. abstract class

https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface- in-java/

36. Encapsulation

- Wrapping up of data under a single unit is called Encapsulation.
- It is a mechanism of binding data and code in to single module
- It increases the length of the code.

37. Tightly Encapsulated class

- If every data member in a class is declared with private then it is called Tightly Encapsulated Class
- It improves the modularity of the class

38. Inheritance (Is-A relation)

- When one object acquires all the properties and behaviors of parent object is known as inheritance.
- Code reusability is main advantage

39. Has-A relation

- The instance of one class has a reference to an instance of another class or other instance of same class.
- Also referred as Composition , or aggregation
- Mostly we are using new key word to implement Has-A relation
- It increases the dependency between classes and creates complications.

40. Association

- It is used to establish relationship between two separate classes through their objects. It can be

  One-One

  One–many

  Many-many, many-one

41. Composition

- If A & B are two classes related to each other whenever object A is destroyed then automatically B object will be destroyed .This relationship is called strong association also called as Composition.

42. Aggregation

- If A & B are two classes related to each other whenever object A is destroyed there is no guarantee of destruction of B object. That means without existing object of A there is a chance of existing the B class Object. This association is called Aggregation.

43. Polymorphism

- An object behaves differently in different situation that means one thing is performed by different ways.
- We can achieve polymorphism by using

  Overloading (compile time polymorphism)
  Overriding (Run time polymorphism)

44. Method signature

- Method Name with parameters is called as method signature
  Method Signature = method Name + Parameters

45. Overloading

- Same method name with different Method Signature, where signature can be different by number of parameter or type of parameter or both

46. Overriding

- If a sub class has the same method as declared in parent class is defined as Overriding.
- Overriding means to override the functionality of existing method.
  Rules:
  1. Method must have same name and parameters in parent and child
  2. Must be follow inheritance.
  3. Not accept the private, static and final.

47. Static block

- If you want to perform any operations at the time of loading the class then we have to define that functionality inside the static block. Because if you declare the block as static it will be executed at the time of class initializing only.

48. Constructor

- Whenever we are creating object some peace of code will be executed automatically to perform initialization. That peace of code execution part is called Constructor.

  Rules:

  1. Constructor name and class name must be same.
  2. Must have no explicit return type

  Types:

  1. Default Constructor
  2. Parameterized Constructor

49. Coupling

- Coupling is relation between two classes. That is if one class has know about the class then it said to be those are in coupling.

  Types:

  1. **Loose Coupling ---** class A has know about what class B is exposed only
  2. **Tightly Coupling –** class A knows the internal working of class B also

  http://javaeasy.weebly.com/types-of-coupling.html

50. Cohesion

- Cohesion refers to all about how a single class is designed. It must be associated with making sure that a class is designed with a single well-focused purpose.

51. Garbage collection

- Garbage collector is a program which runs on the **JVM** which gets rid of the objects which are not being used by a java application anymore. It is form automatic memory management.

https://stackoverflow.com/questions/3798424/what-is-the-garbage-collector-in-java

52. Exceptions

http://www.interviewgrid.com/interview_questions/java/java_exceptions

https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception

- Exception is an error event that can happen during the execution of program. It was disturbs the flow of original execution.

  Types:

  1. **Checked Exceptions** –- these exceptions are raised at the time of compiling. So we can handle it before executing the program
  2. **Un Checked exceptions ---** these are raised at the time of executing.

  Exception Handling Keywords:

  1. throw –--- used to throw the specific exception
  2. throws --- used to handle multiple exceptions
  3. try-catch
  4. finally

  Exception Methods:

  1. getMessage() --- return the exception message while creating exception
  2. getLocalizedmessage() – return exception message
  3. getCause() --- returns the exception cause
  4. toString() --- return string contains the name of throwable class and localized message
  5. printStackTrace()

53. Concurrency

- It is a java API which provides the high-level threading facility that is easier to use compared to low level threading facility using Thread and Runnable
- It is a high level abstraction for creating and executing the multi thread programs instead of low-level multi threading via explicit thread creation and execution. http://www.interviewgrid.com/interview_questions/java/java_concurrency

54. Arrays

- An array is indexed collection of fixed number of homogeneous data elements.
- Arrays are fixed in size that means once we can create array there is no chance to increase or decrease the array size based on our requirement.
- Array could hold only homogeneous data elements (same type).

  Student s = new Student [1000]

  S[0] = new Student[];
- Array hold only homogeneous type but we can resolve this issue using Object[].

  Object[] o = new Student[1000];

  o[0] = new Student[];

  o[1] = new Admin[];

- Array is not recommended when memory point of view.
- Array is highly recommended when performance point of view

55. Arrays default initialization values

- Int array ------ 0
- Boolean array----- false
- Object array --- null

56. Multi Dimensional Arrays

```
• // Declare, instantiate and initialize a multi-dimensional array separately
  //Declare a multi-dimensional array
  String[][] stringArray;
  //Instantiate multi-dimensional array
  stringArray = new String[2][5];
  //Initialize multi-dimensional array
  stringArray[1][2]='Test String';

  //Declare, instantiate and initialize a multi-dimensional array
  String[][] stringArray = {'Str1','Str2','Str3','Str4','Str5'},
  {'abc','efg'}
```

https://study.com/academy/lesson/java-multidimensional-arrays.html

57. Wrapper Classes that are acts on collections

- **Synchronized Wrapper**
- **UnModifiable Wrapper**
- **Checked Interface Wrapper**

58. Legacy Class

- Early version of java did not include the Collections framework. It only defined several classes and interfaces that provide methods for storing objects. When Collections framework was added in J2SE 1.2, the original classes were reengineered to support the collection interface. These classes are also known as Legacy classes
- The following are legacy classes.
  1. Dictionary
  2. Hash table
  3. Properties
  4. Stack
  5. Vector

https://www.studytonight.com/java/legacy-classes-and-interface.php