1. **What is Java?**

Java is a platform-independent high-level programming language. It is platform-independent because its byte codes can run on any system regardless of its operating system.

2. **What are the features of Java?**

- Object-oriented programming (OOP) concepts
- Platform independent
- High performance
- Multi-threaded

3. What are the OOP concepts?

- Inheritance
- Encapsulation
- Polymorphism
- Abstraction
- Interface

4. **What is data encapsulation, and why is it useful?**

Encapsulation is a concept in Object-Oriented Programming for combining properties and methods in a single unit. Encapsulation helps developers follow a modular approach for software development because each object has its own set of methods and variables and serves its functions independent of other objects. In addition to that, encapsulation serves data hiding purposes.

5. **What is polymorphism?**

Polymorphism is one interface with many implementations. This characteristic allows you to assign a different meaning or usage to something in different contexts. For example, you can use polymorphisms to enable more than one form for entities, such as variables, functions, or objects.

6. **What are the types of polymorphism, and how do they differ?**

There are two types of polymorphism:

- *Compile-time polymorphism* is method overloading.
- *Run-time polymorphism* uses inheritance and interface.

7. **What does an interface in Java refer to?**

- An interface as it relates to Java is a blueprint of a class or a collection of abstract methods and static constants.
- Each method is public and abstract, but it does not contain any constructor.

8. **What are constructors in Java?**

In Java, a constructor refers to a block of code used to initialize an object. In addition:

- Constructors must have the same name as that of the class.
- Constructors have no return type.
- Creating an object will call a constructor automatically.

9. **Name and explain the types of constructors in Java.**

The two types of constructors in Java are the *Default Constructor* and the *Parameterized Constructor*.

- *Default Constructor*
  - Does not take any inputs
  - Main purpose is to initialize the instance variables with the default values
  - Widely used for object creation
- *Parameterized Constructor*
  - Capable of initializing the instance variables with the provided values.
  - These constructors take the arguments.

10. **What is JDK?**

- JDK stands for Java development kit.
- It can compile, document, and package Java programs.
- It contains both JRE and development tools.

11. **What is JVM?**

- JVM stands for Java virtual machine.
- It is an abstract machine that provides a run-time environment that allows programmers to execute Java bytecode.
- JVM follows specification, implementation, and runtime instance notations.

12. **What is JRE?**

- JRE stands for Java runtime environment.
- JRE refers to a runtime environment that allows programmers to execute Java bytecode.
- JRE is a physical implementation of the JVM.

13. **In Java, what are the differences between heap and stack memory?**

Memory

- Stack memory is used only by one thread of execution.
- All the parts of the application use heap memory.

Access

- Other threads can't access stack memory.
- Objects stored in the heap are globally accessible.

Memory Management

- Stack follows the LIFO manner to free memory.
- Memory management for heap stems from the generation associated with each object.

Lifetime

- Stack exists until the end of the execution of the thread.
- Heap memory lives from the start till the end of application execution.

Usage

- Stack memory only contains local primitive and reference variables to objects in heap space.
- Whenever you create an object, it is always stored away in the heap space.

14. **What is a JIT compiler?**

A JIT compiler runs after the program is executed and compiles the code into a faster form, hosting the CPU's native instructing set.

15. **How does a JIT compiler differ from a standard compiler?**

JIT can access dynamic runtime information, and a standard compiler does not. Therefore, JIT can better optimize frequently used inlining functions.

16. **What is an inner class?**

An inner class is a class that is nested within another class. An Inner class has access rights for the class that is nesting it, and it can access all variables and methods defined in the outer class.

17. **What is a subclass?**

A subclass is a class that inherits from another class called the superclass. Subclass can access all public and protected methods and fields of its superclass.

18. **What is a package in Java?**

In Java, packages are the collection of related classes and interfaces which bundle together.

19. **How can developers use packages in Java?**

Packages in Java allow developers to modularize the code and optimize its reuse easily. In addition, developers can use other classes to import and reuse the code within the packages.

20. **What are the advantages of packages in Java?**

- Packages help developers avoid name clashes.
- Packages provide easier access control.
- Packages can also contain hidden classes that are not visible to the outer classes and are only used within the package.
- Packages create a standardized hierarchical structure, making it easier to locate related classes.

21. **What is a class in Java?**

All Java codes are defined in a class. It has variables and methods.

22. **What is a variable within Java?**

Variables are attributes that define the state of a class.

23. **How do you use a method in Java?**

Methods are the place where the exact business logic has to be done. Methods contain a set of statements or instructions that satisfy specified requirements.

24. **What is a Java object?**

An object is an instance of a class. The object has a state and behaviour.

25. **What is a singleton class, and how can it be used?**

A singleton class in Java can have only one instance. Therefore, all its methods and variables belong to this instance. The singleton class concept is useful when the developer needs to limit the number of objects for a class.

26. **What is a constructor in Java?**

The sole purpose of using Constructors in Java is to create an instance of a class. Creating an object of a class will invoke them. Some key features of Java constructors include:

- Constructors can be public, private, or protected.
- If a class already defines a constructor with arguments, you can no longer use a default no-argument constructor — you have to write one.
- Instantiating a class will only call them once.
- They must have the same name as the class itself.
- They do not return a value, and you do not have to specify the keyword void.
- If you do not create a constructor for the class, Java helps you by using a so-called default no-argument constructor.

27. **What does the term constructor overloading mean?**

Constructor overloading indicates passing different numbers and types of variables as arguments, all of which are private variables of the class.

28. **How are non-primitive variables used in Java?**

Non-primitive variables always refer to objects in Java.

29. **In Java, what is a static variable?**

A static variable is associated with a class and not objects of that class.

30. **What are Java data types, and how are they grouped?**

In Java, a variable must be a specified data type such as an integer, floating-point number, character Boolean, or string. The two groups of data types are:

- Primitive data types, which include byte, short, int, long, float, double, Boolean, and char
- Non-primitive data types, which include string, arrays, and classes.

31. **How do you define primitive data types and describe each by size and description?**

- *byte* is 1 byte in size. It stores whole numbers from -128 to 127
- *short* is 2 bytes in size. It stores whole numbers from -32,768 to 32,767
- *int* is 4 bytes in size. It stores whole numbers from -2,147,483,648 to 2,147,483,647
- *long* is 8 bytes in size. It stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
- *float* is 4 bytes in size. It stores fractional numbers and is sufficient for storing 6 to 7 decimal digits.
- *double* is 8 bytes in size. It stores fractional numbers and is sufficient for storing 15 decimal digits.
- *Boolean* is 1 bit in size. It stores true or false values.
- *char* is 2 bytes in size. It stores a single character/letter or ASCII values.

32. **What do the terms autoboxing and unboxing mean in Java?**

- *Autoboxing* represents the Java compiler that automatically transforms primitive data types into object equivalents or wrapper types to ease compilation.
- *Unboxing* is the automatic transformation of wrapper types into their primitive equivalent.

33. **What are wrapper classes in Java?**

- Every primitive data type has a class dedicated to it, known as wrapper classes.
- We call them wrapper classes because they "wrap" the primitive data type into an object of that class.
- Wrapper classes convert the Java primitives into reference types (objects).

34. **In Java, what are the differences between methods and constructors?**

Methods	**Constructors**
Used to represent the behaviour of an object.	Used to initialize the state of an object.
Must have a return type.	Does not have a return type.
Needs to be invoked explicitly.	Invoked implicitly.
The compiler does not provide a default method.	The compiler provides a default constructor if the class has none.
Method name may or may not be the same as class name.	Constructor name must always be the same as the class name.

35. **Can you override a private method or static method in Java?**

You cannot override a private or static method in Java. You cannot override a private method in subclass because it's not accessible there.

36. **What is method hiding?**

Method hiding is an alternative to overriding a private or static method, which occurs when you hide the superclass method. You create a similar method with the same return type and same method arguments in child class. For example, you can create another private method with the same name in the child class.

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

- Equals() method
  - Is defined in object class in Java.
  - Used for checking the equality of two objects defined by business logic.
- "==" (equality operator)
  - A binary operator provided by Java programming language and used to compare primitives and objects.
  - public boolean equals (object o) is the method provided by the Object class.
  - Default uses == operator to compare two objects. For example, you can override a method like string class. equals() method is used to compare the values of two objects.

38. **Can you write multiple catch blocks under a single try block?**

Yes, you can have multiple catch blocks under a single try block. Your approach should be from specific to general.

39. **What is a local variable?**

Local variables are defined in the method and scope of the variables that exist inside the method itself.

40. **What is an instance variable?**

An instance variable is defined inside the class and outside the method. The scope of the variables exists throughout the class.

41. **How do you use final keywords and final variables in Java?**

- When Java programmers use final keywords with a variable of primitive data types, they cannot change the variable's value.
- When you use final with non-primitive variables, you cannot change the members of the referred object.

42. **What is inheritance in Java?**

Inheritance in Java is the concept where the properties of one class can be inherited by the other. It helps to reuse the code and establish a relationship between different classes.

43. **In Java, what types of classes perform inheritance?**

- Parent class
- Child class

44. **What types of inheritance does Java support?**

- Single inheritance
- Multilevel inheritance
- Hierarchical inheritance
- Hybrid inheritance

45. **What is Java exception handling?**

In Java, exceptions are objects. When you throw an exception, you throw an object. However, you can't throw just any object as an exception — only those objects whose classes descend from throwable. Throwable serves as the base class for an entire family of classes, declared in java.lang, that your program can instantiate and throw.

46. **What are the differences between unchecked exception, checked exception, and errors?**

- An **Unchecked exception** inherits from RuntimeException (which extends from exception). The JVM treats RuntimeException differently as there is no requirement for the application code to deal with them explicitly.
- A **checked exception** inherits from the exception class. The client code has to handle the checked exceptions either in a try-catch clause or has to be thrown for the super-class to catch the same. A checked exception thrown by a lower class (sub-class) enforces a contract on the invoking class (super-class) to catch or throw it.
- **Errors** (members of the error family) usually appear for more serious problems, such as OutOfMemoryError (OOM), that may not be so easy to handle.

47. **What are loops in Java?**

You would use a loop to execute a statement or a block of statements repeatedly.

48. **What are the types of loops in Java, and how are they used?**

- **For loops** are used in Java to execute statements repeatedly for a given number of times. For loops are used when the programmer knows the number of times to execute the statements.
- The **while loop** is useful when certain statements need to execute repeatedly until it fulfils a condition. In while loops, it checks the condition *before* the execution of statements.
- The **do while loop** is the same as the while loop, except that it checks the condition *after* the execution of block of statements. Also, do while loop statements execute at least once.

49. **What is an infinite loop?**

An infinite loop runs without any condition and runs infinitely. You can break an infinite loop by defining any breaking logic in the body of the statement blocks.

50. **How do you declare an infinite loop?**

```
for (Condition)

{

// Statements to execute

// Add any loop breaking logic

}
```

51. **What is the difference between the continue and break statement?**

Break and continue are two important keywords used in loops. When using a *break* keyword in a loop, the loop breaks instantly. The current iteration breaks when using the *continue* keyword, and the loop continues with the next iteration.

52. **What is the entry point in Java, and how is it written?**

- main() in Java is the entry point for any Java program.
- main() is always written as public static void main string args.

53. **In Java, what are public static void main string args?**

Public static void main string args, also known as public static void main(String[] args), means:

- *Public* is an access modifier used to specify who can access this method. Also, this method is accessible by any class.
- *Static* is a keyword in java that identifies when it is class-based. main() is made static in Java to access it without creating the instance of a class. If main is not made static, the compiler will throw an error as main() is called by the JVM before creating any objects. It can only invoke static methods directly via the class.
- *Void* is the return type of the method that defines the method. That method does not return a value.
- *Main* is the name of the method searched by JVM as a starting point for an application (with a particular signature only). It is also the method where the main execution occurs.
- *String args[]* is the parameter that passes to the main method.

54. **In Java, what's the purpose of static methods and static variables?**

Developers use a static keyword to make a method or variable shared for all objects when there is a requirement to share a method or a variable between multiple objects of a class. This is used instead of creating separate copies for each object.

55. **How do you use, call, and access a static method in Java?**

- You must use the static keyword before the method name.
- Call a static method using the class (className.methodName).
- Static methods cannot access any non-static instance variables or methods.

56. **How do you use, call, and access a non-static method in Java?**

- You do not need to use the static keyword before the method name.
- Call a non-Static method like any general method.
- Non-static methods can access any static method or static variables without creating an instance of the class.

57. **In Java, what are this() and super(), and where are you required to use them?**

In Java, super() and this() are special keywords used to call the constructor. When using this() and super(), they must be the first line of a block.

58. **What does this() represent, and how is it used in Java?**

- this() represents the current instance of a class
- Used to:
    - Call the default constructor of the same class
    - Access methods of the current class
    - Point to the current class instance

59. **What does super()represent, and how is it used in Java?**

- super() represents the current instance of a parent/base class
- Used to:
    - Call the default constructor of the parent/base class
    - Access methods of the base class
    - Point to the superclass instance

60. **What is a Java switch statement, and how can it be used?**

- As a standard programming logic, it can simply be achieved by using if…else conditions.
- In programs involving more complicated cases, complex scenarios require calling several methods for which switch solves this problem.
- Switch avoids several nested if…else statements.
- In Java scenarios that yield a high number of iterations, the switch is typically faster than using if….else statements.

61. **What is the default of the switch case?**

In a switch statement, the default case executes when no other switch condition matches. Because the default case is optional, you can only declare it after coding all other switch cases.