Nov 16, 2023 12:00 PM

1. is main method is mandatory for compile or for execution ?
Ans : You can compile a Java program without a main method, but you cannot execute it. If your Java program doesn't contain the main method, then you will get an error.

2. when java file name and class name should be same ?
ans : In Java, the file name and class name should be the same when the class is public. This is because the compiler needs to know which class is the main class to run the program. If the file name and class name are not the same, the compiler will not be able to find the main class and the program will not run.

3. in single java file how many class can be created ?
ans : There can be any number of classes in a single Java file, but only one of them can be public. The public class is the one that is executed when the program is run. The other classes can be private or protected, and they can only be accessed by methods in the public class.
4. can we compile multiple class at a time ?

ans : Yes, you can compile multiple Java classes at a time using the javac command. When you have multiple Java source files that are part of the same program and have dependencies on each other, you can compile them together.
Ex :- javac MainClass.java HelperClass.java

5. can we executed multiple class at a time ?
ans : YES, we can execute multiple class using TestNG ,
TestNG is a testing framework for Java that allows you to run tests in a structured and flexible way, often used for unit testing, integration testing, and functional testing in Java applications. When you want to execute multiple classes at once using TestNG, you typically create a test suite XML file that specifies which classes should be included in the test run.

```
// TestClass1.java
import org.testng.annotations.Test;
public class TestClass1 {
    public void testMethod1() {   //code }
        }
public class TestClass2
{

// TestClass2.java
    public void testMethod2()
        {
         // Your test code here
         }
}
<!-- testng.xml -->
```

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="MyTestSuite">
  <test name="MyTest">
    <classes>
      <class name="path.to.TestClass1"/>
      <class name="path.to.TestClass2"/>
      <!-- Add more classes if needed -->
    </classes>
  </test>
</suite>
```

6. will jvm executes user defined methods automatically

ans : No, the JVM does not execute user-defined methods automatically. The JVM only executes the main() method. If you want to execute a user-defined method, you must call it from the main() method.

7. what are the two keyword we have to provide memory to class member ?

ans : In Java, there are two keywords used to allocate memory to class members (fields or variables): new and static

new Keyword:

The new keyword is used to dynamically allocate memory for an object. It is commonly used when creating instances of a class (objects) at runtime.

When you create an object using new, memory is allocated on the heap, and the constructor of the class is invoked to initialize the object.

static Keyword:

The static keyword is used to create class-level variables and methods. Memory for static members is allocated once per class, rather than per instance of the class.

static members belong to the class itself, not to any specific instance of the class.

8. what is the difference between static and new keyword ?

ans : In Java, the new keyword creates a new object. The static keyword is used to perform operations that are not dependent on instance creation.

Here's some more information about the static and new keywords:

Static:

Static members are initialized when the class is loaded into memory.

Static methods can be used to create utility or helper classes.

Static methods do not require an instance to be invoked upon.

new:

Allocates memory for the object.

Used to create new objects.

9. if class does not have main method how can we execute the class methods ?

ans :

have some douvt

{

Yes, we can execute a java program without a main method by using a static block. Static block in Java is a group of statements that gets executed only once when the class is loaded into the memory by Java ClassLoader, It is also known as a static initialization block.

}

10. can we call main method explicitly , what is the syntax ?

ans : Yes, you can call the main method explicitly in Java. The syntax is

```
class MyClass {
public static void main(String[] args) {
        System.out.println("Hello, World!");
        }
        public static void callMain() {
         main(null);
                }
}
```

11. what will happen if we call main method in its own body ?

ans : If you call the main method in its own body, it will create an infinite loop. The reason for this is that the main method is the entry point for the program, and when it is called, it will start executing the code in its body. This code will then call the main method again, which will then start executing the code in its body again, and so on. This will continue until the program runs out of memory or is manually terminated.

12. cannwe overload the main() method ?

ans : Yes, we can overload the main() method in Java. A Java class can have any number of overloaded main() methods. But it should have one main() method with signature as “public static void main(String[] args)” to run. If not class will compile but not run.

13. what is meaning of auto compilation ? in java

ans : Auto compilation in Java is a feature that allows the Java compiler to automatically c ompile Java source code when it is changed. This can be a useful feature for developers who are working on large projects, as it can save them time from having to manually compile their code each time they make a change.

There are a few different ways to enable auto compilation in Java. One way is to use the javac compiler with the -auto option. Another way is to use the Eclipse IDE, which has a built-in compiler that supports auto compilation.

14. what is the deffrent between System.out.println and system.out.println and system.out.prntf(); ?

ans : in Java, System.out.println() prints a line of text. System.out.printf() formats a line of text.

Here's some more information about these functions:

System.out.println()

Creates a new line after printing the string. It can be used for simple text or text containing concatenation. For example, println("a: " + a + " b: " + b).

System.out.printf()

Does not create a new line after printing. It can be used to format strings. For example, `printf("a: %d b: %d", a, b)`.

15. what is the java source file structure ?

ans : A Java source file has the following structure:

Package statement: The package statement is used to declare the package to which the class belongs.

Import statements: The import statements are used to import classes from other packages.

Class definition: The class definition is the main part of the Java source file. It contains the declaration of the class, its members, and its methods.

16. can we place package statement anywhere in the java file ?

ans : No, the package statement in Java must be the very first statement in a Java file. It is used to organize classes into logical groups. The package statement is followed by import statements, which are used to import classes from other packages. After the import statements, the rest of the Java file can be defined.

17 what is the use of main method parameter in java ?

ans : The main() method can also accept parameters from the command line. These parameters are passed to the main() method as an array of strings. The name of the array is args. You can use the args array to get information from the user, such as the name of a file to read or the port number to listen on.

18 which part of main method can be changable ?

ans : Name of the String array argument: For example, you can change args to myStringArgs

Position of the static and public keywords: You can interchange these keywordsOrder of the main method: You can change the order of public static void main() to static public void main().

19 hat is the syntax to call main method explicitly ?

ans :
```
public class MainMethodExample {
  public static void main(String[] args) {
```

```
        System.out.println("Inside the main method");
    }
    public static void anotherMethod() {
        System.out.println("Inside another method");
        // Calling the main method explicitly
        main(new String[]{"arg1", "arg2"});
    }
    public static void main(String arg) {
        System.out.println("Inside the overloaded main method");
    }
}
```

20. why main method is called mediator method between programmer and jvm ?

ans : The main method is called the mediator method between the programmer and the JVM because it is the entry point for the program. This means that the JVM will start executing the program at the main method. The main method is also responsible for initializing the program and calling the other methods in the program.

21. what is coding standards and naming conventions in java ?

ans : ava coding standards and naming conventions are a set of rules that define how Java code should be written. These standards are important because they help to make code more readable, consistent, and maintainable.

22. what is the token in java ?

ans : a token refers to the smallest individual units or elements in the source code of a program. Java source code is made up of a series of tokens, and these tokens are the building blocks that form the structure of the code. The Java compiler recognizes and processes these tokens to understand the syntax and semantics of the program.

Common types of tokens in Java include:

Keywords: Reserved words that have a special meaning in Java (e.g., public, class, if, else, etc.).

Identifiers: Names given to variables, methods, classes, etc. by the programmer. Identifiers must follow certain naming conventions.

Literals: Represent constant values in the code, such as numeric literals (e.g., 5, 3.14), string literals (e.g., "Hello, World!"), etc.

Operators: Symbols that perform operations on variables and values (e.g., +, -, *, /, ==, !=, etc.).

Separators: Characters that separate various elements of the code, such as commas ,, semicolons ;, parentheses (), etc.

Comments: Used for documentation and are ignored by the compiler. They can be single-line (//) or multi-line (/* */).

23. can we use predefine class name as identifier ?

ans : Yes, you technically can use a predefined class name as an identifier in Java, but it is s trongly discouraged. Using the names of predefined classes for your own identifiers can lead to confusion and make your code less readable and maintainable. It is best practice to choose meaningful, unique, and descriptive names for your variables, methods, and classes.

24. if we create a class with name String , how can we differentiate predefined class String ?

ans : It is not recommended to create a class with the same name as a predefined class in Java. This can lead to confusion for other developers who are working on your code, as well as the compiler, which may not be able to differentiate between your class and the predefined class.

If you need to create a class with the same name as a predefined class, you can use the import statement to import the predefined class. This will ensure that the compiler can differentiate between the two classes.

25. access modifier is signature or not in java?

ans : No, access modifier is not part of the method signature in Java.

The method signature is the combination of the method name and the parameter types. The access modifier is a keyword that specifies the visibility of the method. It is not part of the method name or the parameter types, so it is not part of the method signature.

26. null is an keyword or not in java ?

ans : No, null is not a keyword in Java. It is a literal.

A keyword is a reserved word that has a special meaning to the compiler. Literals are values that can be assigned to variables.

The null literal is used to represent the absence of a value. It can be assigned to any reference variable, but it cannot be used as an operand in an expression.

27. annotation is an keyword or not in java ?

ans : No, annotation is not a keyword in Java. It is a non-access modifier that provides additional information about a program element. Annotations are used to provide metadata about classes, methods, fields, and other program elements. They can be used for a variety of purposes, such as documentation, configuration, and performance tuning.

Annotations are preceded by the @ symbol. For example, the following code shows an annotation that is used to document a method:

----------DATA TYPE----------

28. how many types of data types java supports ?

ans : byte: an 8-bit signed integer

short: a 16-bit signed integer
int: a 32-bit signed integer
long: a 64-bit signed integer
float: a 32-bit floating-point number
double: a 64-bit floating-point number
char: a single character
boolean: a true or false value

29. what is difference between primitive type and reference type in java ?

ans : The main difference between primitive and reference type in java is that primitive type always has a value, it can never be null but reference type can be null, which denotes the absence of value.

It's true that primitive types are immutable in the sense that their values cannot be changed once they are assigned. If you assign a new value to a variable, it creates a new memory location for that value.

```
int x = 10;
x = 20;  // x now refers to a new memory location with the value 20
```

30. how can we create CUI & GUI for reading runtime values ?

ans : There are two ways to create a CUI (Command-line User Interface) and GUI (Graphical User Interface) for reading runtime values in Java:

1. CUI:

To create a CUI, you can use the Scanner class to read input from the console. Once you have the input, you can use the System.getProperty() method to read the runtime value.

2. GUI:

To create a GUI, you can use the Swing or JavaFX toolkit. Swing is a lightweight GUI toolkit that is included with the Java SE platform. JavaFX is a newer GUI toolkit that is built on top of JavaFX Scene Graph.

31. what is the meaning of command line arguments in java ?

ans : The command line argument in java is the information passed to the program at the time of running the program. It is the argument passed through the console when the program is run. The command line argument is the data that is written right after the program's name at the command line while executing the program.

32. why main method has String [] parameter ?

ans : The main method in Java has a String array parameter to receive any command-line arguments passed to the program when it is executed.

33. how can we read command line arguments in java?

ans : To read command-line arguments in Java, you can use the args parameter of the main() method. The args parameter is an array of String objects, which contains the command-line arguments that were passed to the program.

For example, the following code prints all of the command-line arguments to the

ex:-

```
public class Main {
  public static void main(String[] args) {
    for (String arg : args) {
     System.out.println(arg);
    }
 }
}
```

----------OPERATOR----------

34. WHAT IS Operator and operand ?

ans : An operator is a symbol that represents an operation. An operand is a data item that the operator acts on. In Java, there are many different types of operators, including arithmetic operators, relational operators, logical operators, and assignment operators.

35. what is the meaning of operator precedence & operator associativity ?

ans : Operator precedence is a hierarchy of operators, with operators of higher precedence being evaluated before operators of lower precedence. For example, in the expression a + b * c, the multiplication operation (*) has higher precedence than the addition operation (+), so the multiplication operation will be evaluated first.

36. what are the operator executed from Right to Left in java ?

ans : In Java, the assignment operator (=) is executed from right to left. This means that in the expression a = b = c, the assignment is executed from right to left, and the expression is equivalent to a = (b = c).

The *= operator is executed from right to left, so in the expression a *= b *= c, the multiplication is executed from right to left, and the expression is equivalent to

a = (b *= c) *= a.

37. when will we get ArithmeticException,infinity or NaN from div operator ?

ans : you will get an ArithmeticException when you try to divide an integer by zero. This is because there is no number that can be divided by zero to produce a finite result.

You will get Infinity when you divide a floating-point number by zero. This is because Infinity is a special number that represents the concept of infinity.

You will get NaN (Not a Number) when you divide zero by zero. This is because there is no defined mathematical operation that can be performed on zero divided by zero.

38. how can we compare two primitive value & two objects in java ?

ans : Using the == operator. This operator compares the values of two primitive data types like int, long, short, byte, float, and double. For example, if you have two int variables, a and b, and you want to compare their values, you can use the ==

To compare two objects in Java, you can use the equals() method. This method compares the values of two objects. For example, if you have two String objects, s1 and s2, and you want to compare their values, you can use the equals() method .

39. can you identify when will you get below Is ?

a. CE: bad operand types for the binary operators

b. incomparable types ?

ans:

a: It occurs when the data types of the operands used in the binary operator are incompatible.

```
public class BadOperatorError {
public static void main(String args[]) {
int a = 26;
if( a & 32 == 1){
  System.out.println("inside if block");
}
else{
  System.out.println("inside else block");
}
 }
}
```

// badoperatorerror.java:4: error: bad operand types for binary operator

b: The error "incomparable types" in Java typically occurs when you are trying to compare two values or expressions using the equality (== or !=) or relational (<, <=, >, >=) operators, and the types of those values are not compatible for comparison.

```
int number = 5;
String text = "Hello";
if (number == text) {
  // Error: incomparable types: int and String
}
```

40. what is deffrence between & , | ?

ans: Bitwise AND (&):

Performs a bitwise AND operation between the corresponding bits of two integer operands.

Bitwise OR (|):

Performs a bitwise OR operation between the corresponding bits of two integer operands.

```
int a = 5;    // binary: 0101
```

```
int b = 3;    // binary: 0011
int resultAnd = a & b;  // Result: 0001 (1 in decimal)
int resultOr = a | b;   // Result: 0111 (7 in decimal)
```

41. do we have any inbuilt data type TO STORE different type value ?

Ans: Yes, in Java, you can use Java Collections to store different types of values. Here are some of the most common Java Collections:

42. What is a user define datatype ?

Ans : A user-defined data type (UDT) is a data type that is created by the user. It is a way to extend the built-in data types that are already available in Java. UDTs can be used to create more complex and customized data types that are specific to a particular application.

There are several different types of UDTs that can be created in Java. These include:

Classes:

Classes are the most common type of UDT. They are used to create objects that have properties and methods. Classes can be used to model real-world objects, such as customers, products, and orders.

Enums:

Enums are used to create a list of named constants. They can be used to represent things like days of the week, months of the year, and suits in a deck of cards.

Interfaces:

Interfaces are used to define a set of methods that a class must implement. They can be used to create contracts between different classes.

Annotations:

Annotations are used to provide additional information about a class, method, or field. They can be used for things like logging, debugging, and performance tuning.

UDTs can be a very powerful tool for creating complex and customized data types. They can be used to improve the readability and maintainability of code.

43. create a class to store different types of values ?

Ans Here is a simple Java class that can be used to store different types of values:

```
public class DataStore {
  private Object value;

  public DataStore(Object value) {
    this.value = value;
  }

  public Object getValue() {
    return value;
  }

  public void setValue(Object value) {
    this.value = value;
  }
}
```

This class can be used to store any type of value, including primitive types, objects, and arrays. To use the class, you would first create a new instance of it, passing in the value that you want to store. You can then access the value using the `getValue()` method. To set the value, you can use the `setValue()` method.

Here is an example of how to use the `DataStore` class:

```
DataStore dataStore = new DataStore(10);
int value = (int) dataStore.getValue();

dataStore.setValue("Hello, world!");
String stringValue = (String) dataStore.getValue();
```

44. draw class object memory diagram that is created internally in JVM ?

Ans In Java, the constant pool is indeed a part of the class file structure, and it stores constant values, such as string literals, numeric constants, and references to other classes, methods, and fields. However, it is not stored in the Method Area of the JVM heap.

Here's a more accurate explanation:

Constant Pool in Class Files:

- The constant pool is an integral part of the class file format, which is the binary representation of a compiled Java class.
- It is a table that contains various constants used by the class, including strings, numeric constants, class names, method names, and field names.

Location in JVM Memory:

- When a Java program is executed, the Java Virtual Machine (JVM) loads classes into memory. The memory is divided into several areas, including the Method Area.
- The constant pool is indeed loaded into memory as part of the class loading process. However, it is typically stored in a specific section within the Method Area, not the entire Method Area itself.

Method Area:

- The Method Area is a part of the JVM memory that stores class-level information, including the class bytecode, static variables, and the constant pool.
- It is separate from the Heap, which is where objects (instances of classes) are allocated.

Role of Constant Pool:

- The constant pool plays a crucial role in supporting features like dynamic linking, where classes and methods are resolved at runtime.

- It allows the JVM to efficiently manage and access constant values used by the class during execution.