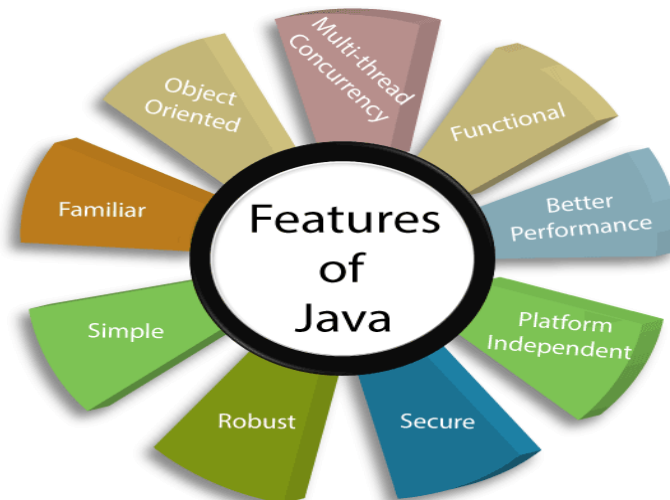


Java Questions with Answers

1. Why did you choose Java?

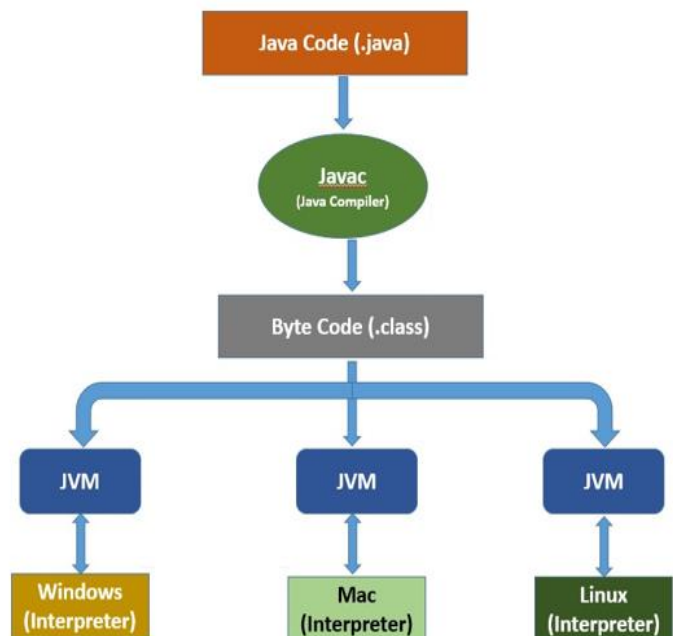
- Java is open source and easy to learn.
- Writing, Compiling and debugging is easy.
- We can reuse the code.

2. What is the use of Java?



3. Why Java is platform independent?

- Java compiler converts Source code to **Byte Code**.
- We can run it in any platform such as Windows, Mac, and Linux etc.



4. Data types and Its Default Value?

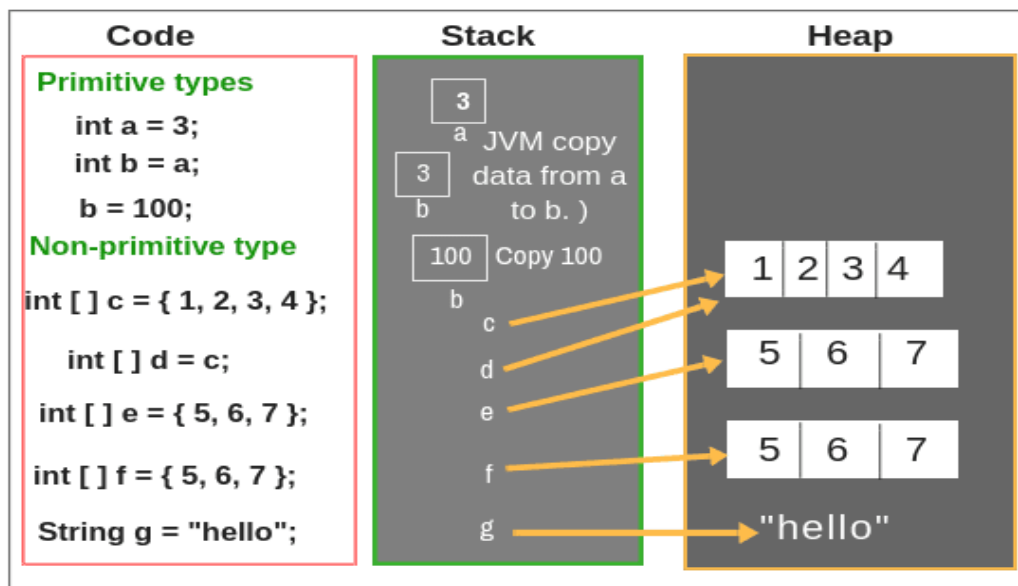
❖ Data type has two types namely,

✚ Primitive: (Store directly)

- Predefined
- Can store only one value
- There are no additional methods

✚ Non-primitive: (Store based on reference)

- Not-Predefined
- Can store more than one value
- It references a memory location which stores the Data. (Reference variables)



Default Values:

Data Type	Memory Size (Byte)	Default Value	Wrapper Class
byte	1	0	Byte
Short	2	0	Short
int	4	0	Integer
long	8	0	Long
float	4	0.0f	Float
double	8	0	Double
char	2	-	Character
boolean	1 bit	False	Boolean

5. Access modifiers?

Public:

- Global Level Access. (inside and outside the package)
- Example: public class student()

Private:

- Class Level Access. (Inside the class only)
- Example: private class student {}

Protected:

- Same like public but to be used with “Extends” keyword
- Example: protected class student {}

Default:

- Package Level Access. (only inside the package)
Example : class student {}

6. What is OOPS? And it's Concepts?

Ans: It is the method of implementation in which our program is oriented in the form of,

Class:

- Combination of method & object

Method:

- Set of actions to be performed

Object:

- **Instance of the class,**
- It allocates memory
- Using object, we can call the method

Concepts:

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

7. Encapsulation?

Ans: Binding code and data together as a single entity (Information).

8. Inheritance? And it's Types.

Ans: Accessing one class property into another class using “**extends**” keyword.

Types:

Single Inheritance

- Accessing one parent class property into one child class.

Multiple Inheritance

- It is impossible in Java
- It can be overcome by Interface
- More than two parent class properties accessed by one child class.

Multilevel Inheritance

- One class property is accessed by one child class which is being accessed by another child class.

Hierarchical Inheritance

- One parent class property accessed by two (or) more child classes.

Hybrid

- Combination of single and multiple Inheritances.

9. Why Multiple Inheritance is very important?

- More than two parent class properties accessed by one child class.
- It is impossible in Java
- But it can be overcome by Interface

10. Polymorphism? Its types and various names.

Ans: One task can be completed in many ways

Types:

 **Method Overloading** / Static Binding / Compile-time Polymorphism

 **Method Overriding** / Dynamic Binding / Run-time Polymorphism

Method Overloading

- In the same class method names will be same,
- Parameters will be different based on
 - Data Type will differ
 - Data count will differ
 - Data order will differ

Method Overriding

- Class names are different,
- Method names and parameters will be same.

11.Constructor? And it's Types.

- When we create an object, the default constructor will be executed automatically.



Types:

- Default/Non-parameterized
- Parameterized

12.Abstraction? And its types.

Ans: Hiding the implementation part.

Types:

-  Abstract class (or) Partial Abstraction
-  Interface (or) fully Abstraction.

13.Abstract class Vs. Interface?

Abstraction/Abstract Class	Interface
Partial abstraction	Fully abstraction
"Extends" keyword is used	"Implements" keyword is used
It supports both abstract method & non abstract method	It supports only abstract method

Cannot create a object for Abstract Class & Interface, It should be extends or implemented to child class to call those methods.

14. Variable and Its types.

- Variable is a piece of memory that can contain a data value.
- Variables has three types namely,

Local Variable

- The variable which is assigned a value inside the method.
- Local variable should be initialized by a value.
- Its value is only inside the method.

Class/Instance variable

- The variable which is assigned by a value outside the method, inside the class.
- Class variable need not be initialized by a value, if not initialized the default value "0" will be executed.
- It can be used anywhere inside the program.

Static variable

- We can call this static variable without using an object.
- It can be only given for the class variable.
- When a class variable is assigned as public static, the variable can be used inside and outside the package. (class.objectvariable)

15. Static?

- Static is a keyword which can be used in **class variable** and **method level**, to call directly **without using an object**.
- When a class variable is assigned as public static, the variable can be used inside and outside the package. (**class.objectvariable**)

16. This vs. Super?

This:

It is **current** class reference.

Super:

It is **parent** class reference.

17. Final-Finally-Finalize?

Final	Finally	Finalized
It is a keyword	It is a block	It is method
It can be used in 3 levels, <ul style="list-style-type: none">- Final class can't be inherited- Final method can't be overridden- Final variable values can't be changed	It will execute whether the exception occurs or not	This method is used to clear the memory

18. Throw vs. Throws?

Throw	Throws
Used inside the method	Used in the method level ,
It can handle only one Exception	it can handle more than one Exception
It will throw Exception	It will declare the exception

19. For loop? And Nested For loop?

For loop	Nested For loop
A For loop is an iterative structure (or) block that repeatedly executes a block of statements unless a specific condition is met or a break statement is encountered.	A Nested for loop is a for loop inside another for loop .

20. While vs. Do-while?

While	Do-while
The while loop in java executes one or more statements after testing the loop continuation condition at the start of each iteration.	The do-while loop , however, tests the loop continuation condition after the first iteration has completed.

21. Break vs. Continue?

Break	Continue
It will terminate the loop itself	It will skip the particular iteration

22. Array and Its Methods?

Ans: Storing multiple values in a single variable.

Declaration:

```
Int a [] = new int [5];
```

```
Int a [] = {1, 2, 3, 4, 5 };
```

Features:

- It supports only similar data types
- It is index based
- If the value is not assigned for the index the default value will be executed.

23. String and Its Methods?

✚ Collections of character or word enclosed with double quotes.

✚ String has many methods as follows,

<ul style="list-style-type: none">• CharAt()• IndexOf()• Equals()• Contains()	<ul style="list-style-type: none">• IsEmpty()• Split()• equalsIgnoreCase()• startswith()	<ul style="list-style-type: none">• endswith()• substring()• replace()• toUpperCase()
--	---	--

24. String Buffer vs. String Builder?

String Buffer	String Builder
Mutable	Mutable
Synchronized	Asynchronized
Thread safe	It is not Thread safe
Slower than string buffer heap	Faster than string buffer heap

25. Immutable vs. Mutable?

Immutable	Mutable
If we add a duplicate value it will share the memory.	If we add the duplicate value it will create a new memory.
If we concatenate the values it will create new memory.	If we append the values it will share the memory.

26. Collection vs. Collections?

Collection	Collections
It is an interface and it is a group of objects.	Collections is a class
Types namely, 1.List 2.Set 3. Queue 4.Map	1. It will support dissimilar data types. 2.It is dynamic memory allocation.

27. List vs. Set vs. Map?

List	Set	Map
It allows duplicates	Doesn't allow duplicates	Key will not allow duplicates, it will overwrite the value. But value will allow duplicates
It prints in Insertion order	In random order	Key + value = one entry
Index based	Value based	Key based

28. Array List vs. Linked List vs. Vector? `add()`, `retainAll()`, `removeAll()`; It will support dissimilar data types

Array List	Linked List	Vector
Best: Searching and retrieving	Best: Deletion and insertion	-
Worst: Deletion and insertion	Worst: Searching and retrieving	-
Asynchronized	Asynchronized	Synchronized

Not Thread Safe

Not Thread Safe

Thread Safe

29. HashSet vs. linkedHashSet vs. TreeSet?

It will support only similar Data Types

Hash Set	Linked Hash Set	Tree set
Random order	Insertion order	Ascending order
Not allow duplicate	Not allow duplicate	Not allow duplicate
Allows single null value (but not duplicate null)	Allows single null value (but not duplicate null)	Not allows even single null

30. HashMap vs. HashTable?

Hash Map	Hash Table
Non synchronized	Synchronized
It is not-thread safe and can't be shared between many threads without proper synchronization code	-
Hash map allows one null key and multiple null values	Hash table doesn't allow any null key or value

Random Order, It will accept one Null Key

Random Order, It will not accept one Null Key

31. Array vs. ArrayList?

Array	Array List
Storing multiple values in a single variable	Group of Objects
Allows similar data type	Allows dis-similar data types
Fixed memory	Memory is not fixed
High Memory wastage	No memory wastage

32. Iterator vs. List Iterator?

Iterator	List Iterator
Can be used List, Set, Map	Only in List
Iterator can traverse elements in a collection only in forward direction.	List Iterator can traverse in both forward and backward directions

33. Exception vs. Error?

Exception	Error
Exceptions are the problems which can occur at runtime and compile time	Errors mostly occur at runtime that's they belong to an unchecked type
Two categories 1. Checked exceptions, 2. Unchecked exceptions	Ex: Network issue, JVM Error, out of memory

34. Customized Exception or User Defined Exception?

- If you are **creating** your own **Exception** that is known as **custom exception** or **user-defined exception**.
- Java **custom exceptions** are used to **customize** the **exception** according to **user** need.
- By the help **of custom exception**, you can have your own **exception** and message.

35. Parent of all the Classes and Interfaces?

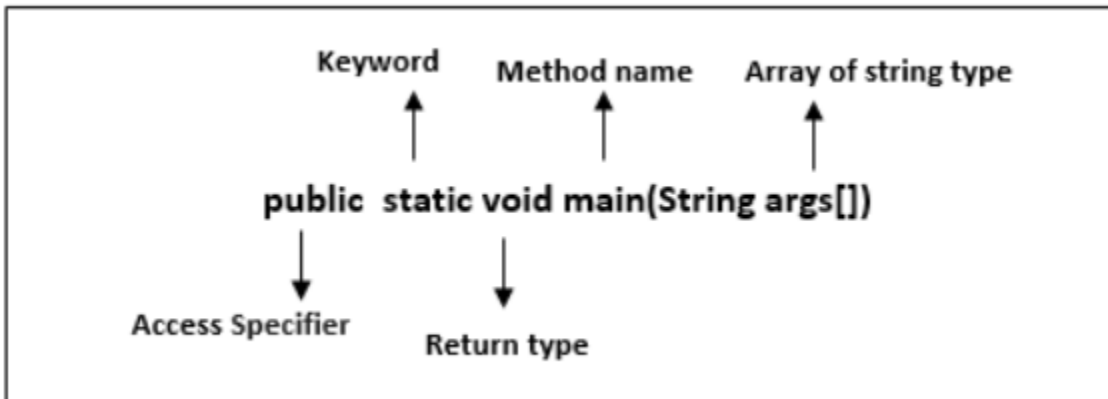
- Parent of all the Classes is "**Object**".
- Parent of all the Interfaces is "**SearchContext**"

36. Pre-increment vs. Post-increment?

Pre-Increment	Post-Increment
++i	i++
Before assigning the value to the variable, the value is incremented by one	After assigning the value to the variable, the value is incremented

37. Explain main method syntax? And, Can we change it?

Declaration: **public static void main (String [] args)**



public: It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected, and default before the main() method, it will not be visible to JVM.

static: You can make a method static by using the keyword static. We should call the main() method without creating an object. Static methods are the method which invokes without creating the objects, so we do not need any object to call the main() method.

void: In Java, every method has the return type. Void keyword acknowledges the compiler that main() method does not return any value.

main(): It is a default signature which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the main() method.

String args[]: The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array. It is used to hold the command line arguments in the form of string values.

38. Can static method be overridden?

Ans:

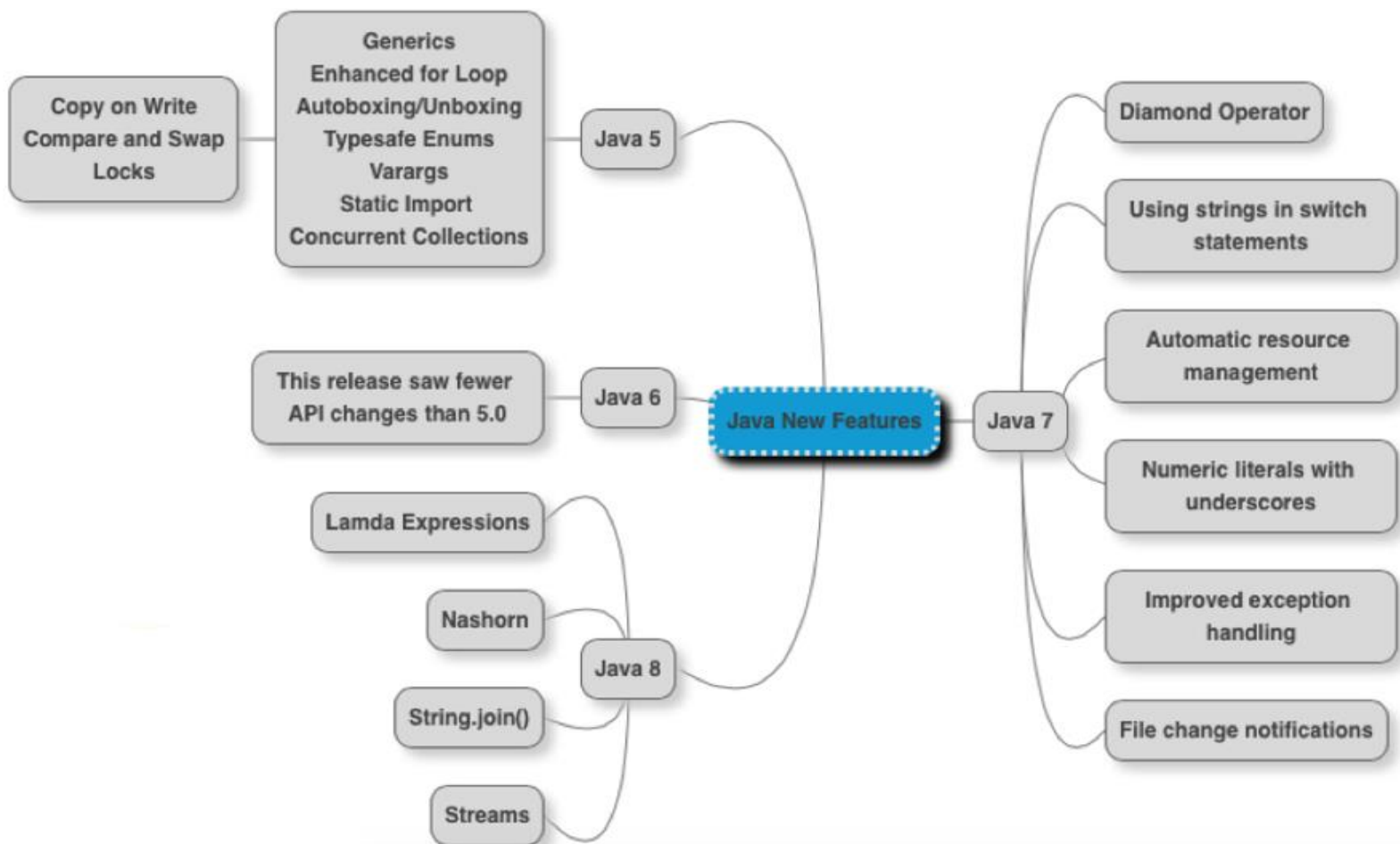
Static methods cannot be overridden because they are not dispatched on the object instance at runtime. The compiler decides which method gets called.

Static methods can be overloaded (meaning that you can have the same method name for several methods as long as they have different parameter types).

39. Enum or Enumerator?

Enum	Enumerator
Enum is a data type	Enumeration is legacy Iterator
Enums are instance controlled classes in java.	Enumeration was the old way to iterate through a collection. It has two methods nextElement and hasMoreElements which are more like next and hasNext methods of Iterator interface.

40. Features of Java 1.8?



*****All the best*****