JAVA INTERVIE QUESTIONS

Command used to compile java program:

javac filename.java

Command used to run java program:

java filename

A static method can access static methods and variables as follows:

* A static method can call only other static methods; it cannot call a non-static method
* A static method can be called directly from the class, without having to create an instance of the class
* A static method can only access static variables; it cannot access instance variables
* Since the static method refers to the class, the syntax to call or refer to a static method is: class name.method name
* To access a non-static method from a static method, create an instance of the class
* Static Method cannot be overridden
* You cannot use super and this inside the static method
* Example of Static method: Arrays.sort() and Collections.sort()

**Characteristics of Non-Static Methods**

A non-static method in Java can access static methods and variables as follows:

* A non-static method can access any static method without creating an instance of the class
* A non-static method can access any static variable without creating an instance of the class because the static variable belongs to the class
* Example of Non-Static method: put() and get() method of hashmap.

WHAT ARE KEYWORDS IN JAVA

Keywords are predefined, reserved words used in Java programming that have special meanings to the compiler.

Some of the keywords are,

package final

class finally

private throws

protected static

public implements

WHAT ARE IDENTIFIERS IN JAVA

Identifiers are the name given to variables, classes, methods, etc. Consider the above code;

**Rules for Naming an Identifier**

* Identifiers cannot be a keyword.
* Identifiers are case-sensitive.
* It can have a sequence of letters and digits. However, it must begin with a letter, $ or \_. The first letter of an identifier cannot be a digit.
* It's a convention to start an identifier with a letter rather and $ or \_.
* Whitespaces are not allowed.
* Similarly, you cannot use symbols such as @, #, and so on.

WHAT IS CONSTRUCTOR IN JAVA

A constructor in Java is similar to a method that is invoked when an object of the class is created.

NEED FOR CONTRUCTOR

constructors are used to assign values to the class variables at the time of object creation, either explicitly done by the programmer or by Java itself (default constructor)

RULES FOR CREATING CONSTRUCTORS

* Constructors are invoked implicitly when you instantiate objects.
* The two rules for creating a constructor are:  
  The name of the constructor should be the same as the class.  
  A Java constructor must not have a return type.
* If a class doesn't have a constructor, the Java compiler automatically creates a **default constructor** during run-time. The default constructor initializes instance variables with default values. For example, the int variable will be initialized to 0
* Constructor types:  
  **No-Arg Constructor** - a constructor that does not accept any arguments  
  **Parameterized constructor** - a constructor that accepts arguments  
  **Default Constructor** - a constructor that is automatically created by the Java compiler if it is not explicitly defined.
* A constructor cannot be abstract or static or final.
* A constructor can be overloaded but cannot be overridden.

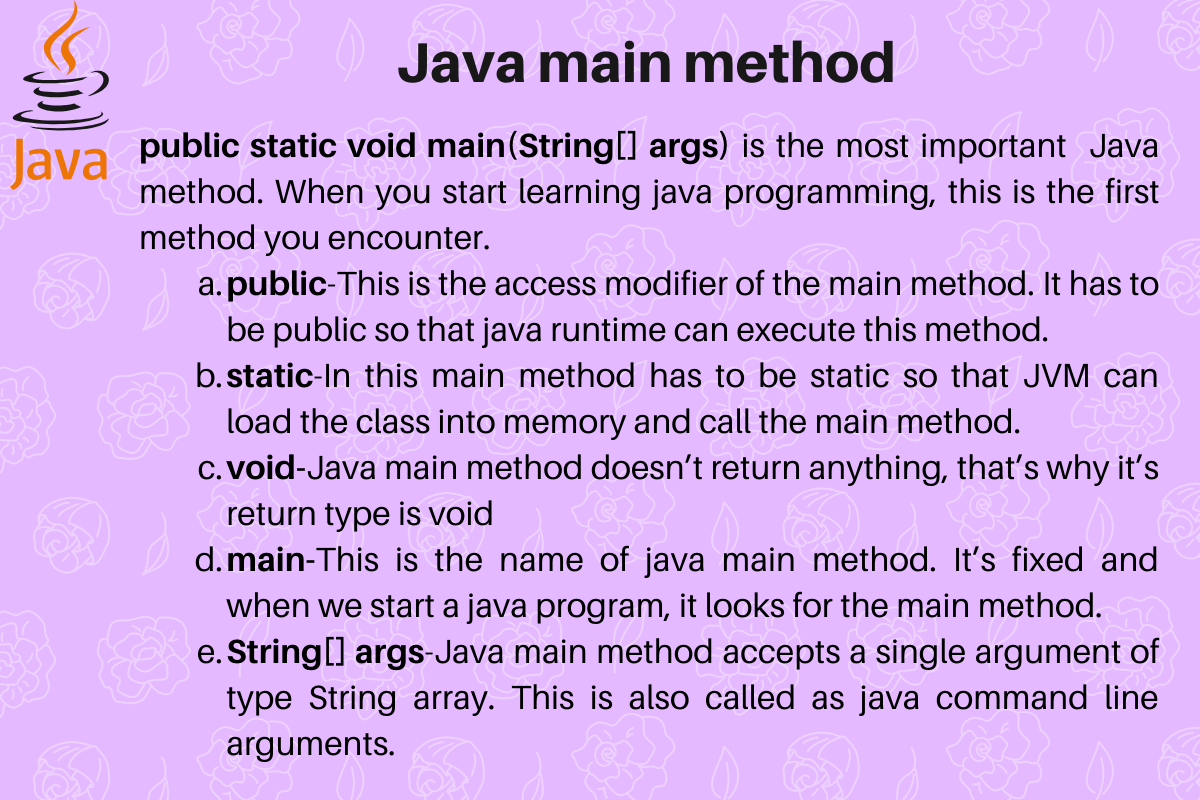
this Keyword:

**this keyword in Java** is a reference variable that refers to the current object of a method or a constructor.

The main purpose of using this keyword in Java is to remove the confusion between class attributes and parameters that have same names.

Following are various uses of ‘this’ keyword in Java:

* It can be used to refer instance variable of current class
* It can be used to invoke or initiate current class constructor
* It can be passed as an argument in the method call
* It can be passed as argument in the constructor call
* It can be used to return the current class instance



## **Declaration of Collections**

public class Collections extends Object

**Need for Collections**

* Developers often find Collections classes operations easy and extremely convenient to perform the basic operations on elements as there is no more need to get into the details of any basic operations and can focus on the more important tasks.
* Collections class uses a particular method to search for a particular element in a collection. Thus, it is able to perform sorting operations on the elements of the Collection interface.
* Collections.binarySearch() is used to search for the desired element in a collection with the help of the popular[Binary Search Algorithm.](https://www.simplilearn.com/binary-search-algorithm-article)
* Collections.sort() is used to perform sorting operations on the specified Collection.
* Collections.max() is used to return the maximum element from a specified Collection.
* Collections.min() is used to return the minimum element from a specified Collection.
* Collections.reverse() is used to reverse the order of the elements present in the specified Collection.
* Collections.copy() function is used to copy one element from one collection to another collection.

|  |  |  |
| --- | --- | --- |
| **HashMap** | **HashSet** | **Hashtable** |
| It allows null for both key and values | HashSet permits to have a single null value | It does not allow null for both key and value |
| HashMap does not maintain any order | HashSet does nto maintain any insertion order, cause insertion order is not constant overtime. But if we use LinkedHashSet it maintains an order | Hashtable does not maintain insertion order |
| HashMap uses put method to insert into hashmap | HashSet uses add method to insert into hashset | HashTable uses put method to insert into hashtable |
| HashMap is not Synchronized, better performance | HashSet is not Synchronized but can be synchronized externally | HashTable is Synchronized (thread safe) |

|  |  |  |
| --- | --- | --- |
| **HashSet** | **TreeSet** | **LinkedHashSet** |
| HashSet is fastest than LinkedHashSet and TreeSet | TreeSet is slow when compared with both Hashset and LinkedHashSet | LinkedHashSet is second fastest next to HashSet |
| HashSet does not maintain any order | TreeSet maintains Sorting Order | LinkedHashSet maintains insertion order |
| HashSet allows null | TreeSet does not allow null | LinkedHashSet allows null |
| HashSet uses equals() method | TreeSet uses compareTo() method | LinkedHashSet uses equals() method |
| HashSet backed by HashMap | TreeSet backed by NavigableMap | LinkedHashSet backed by HashSet |

SUPER KEYWORD:

The super keyword in Java is used in subclasses to access superclass members (attributes, constructors and methods).

USE OF SUPER KEYWORD:

1. To call methods of the superclass that is overridden in the subclass.
2. To access attributes (fields) of the superclass if both superclass and subclass have attributes with the same name.
3. To explicitly call superclass no-arg (default) or parameterized constructor from the subclass constructor.