1. Why do we need static keyword in Java Explain with an Example?
Ans:
It is mainly used for memory management in Java.
i. First - for static variables, we need static keyword
For variables that have a common value for all instances(objects) of the class, we need a static keyword to represent them. For example, in a banking system, account ID is unique to each user(here object in terms of Object-Oriented Programming) so we don’t need static keyword to represent the accountID variable, but interest rate(on deposits) is common for all users irrespective of their account ID so we should represent interestRate with a static keyword, in other words, it is common for all objects of the class or it is a class variable. When a variable is declared as static, then a single copy of that variable is created and shared among all of the objects at the class level.
It is needed because it makes our program more efficient, as every object doesn't allocate separate memory to a static variable.

Example:

```
class Test{
    int accountID; // Instance variable, unique to an object of class Test
    static int interestRate; // Static variable, common for all objects of
class Test
    // Here, we used a static block to initialize the static variable
interestRate
    static{
        interestRate = 2;
    }
}
```

ii. Second - For execution before the main method, we need static keyword
If we want to execute a piece of code before the main execution starts then we can include that particular piece of code inside a static block by using the static keyword. This is because the execution of the static block precedes the execution of the main method in Java. As soon as the class is loaded, the first block to execute are static blocks and only after that the execution of the main method starts. We can write any no of static blocks, those static blocks will be executed from top to bottom. Normally a static block is used to perform the initialization of the static variables.
Example:

```
public class Test{
    static {
        System.out.println("in static block");
    }
    public static void main(String args[]){
        System.out.println("in main method");
    }
}
```

When the above code is run, once the class Test is loaded, the static block will be executed first and then execution starts from the main method.
iii. Third - We want to call a method of a class without creating an object of the class, we require a static keyword for the method
A static method is a method that belongs to a class rather than an instance of a class. This means you can call a static method without creating an object of the class. Static methods are sometimes called class methods. We can access static methods from outside of the class in which they are defined. This is not possible with non-static methods. Subclasses can override static methods, but non-static methods cannot. Static methods can be used to create utility classes that contain general-purpose methods.
Example:

```
public class Test{
    static func(){
        System.out.println("in this static method");
    }
}
// Here, the static method func gets executed without the creation of an
object of class Test
```

2. What is class loading and how does the Java program actually executes?
Ans: In Java, classloading is the process of loading class files into the JVM (Java Virtual Machine) at runtime. It is responsible for loading classes from various sources, such as the file system, network, and databases, and making them available to the JVM for execution.

The class loading process in Java is divided into three phases: loading, linking, and initialization.

I. Loading: In the loading phase, the classloader locates the class file using the fully qualified class name, reads the class file, and converts it into a Class object. The Class object contains the metadata of the class, such as the fields, methods, and constructors.

II. Linking: In the linking phase, the JVM performs several operations on the Class object, such as verifying the class file's integrity, resolving symbolic references, and allocating memory for the class variables.

III. Initialization: In the initialization phase, the JVM initializes the class variables with their default values, and runs the class's static initialization block (if any).

Java Program Execution Steps:
i. Firstly, the Java Program is compiled and converted to a .class file.
ii. Then, the .class file is loaded into the system, i.e, Class is loaded. (The 3 steps mentioned

above)
iii. Once loaded, all memory space in Runtime data areas is utilized accordingly. For example, all methods are allocated under the Method Area, all objects are allocated in the Heap area, and all local variables in the Java Stack Area, in other words, all objects, methods, variables, and classes are allocated memory.
iv. After this, execution passes to the execution engine which comprises Interpreter and JIT Compiler. Both are inside the JVM. JIT Compiler helps the system to run faster.
v. After all the above steps, execution passes on to the Operating System(OS) which takes care of further processing.

3. Can we make a local variable as static?
Ans: No local variables cannot be made static.
Static keyword for Static variables is used as variables for the whole class and they are declared inside a class outside any method and are initialized inside a static block. And the scope of static variables is the whole class. This is the reason static variables can be accessed using the class name.
On the other hand, Local variables are method variables in the sense that their scope is limited to the method in which they are declared. They are declared inside a method. We cannot use these Local variables outside the method in which they are declared.
So if we have a static local variable (a variable whose scope is limited to the method), it violates the purpose of static.
Declaring a local variable as static means that we try to expand its scope and this is not possible. If we will try to attempt this the compiler will give an error.
When we declare a static variable inside a method it means its scope is limited to that area and its value varies from object to object which is against the definition of static variables. So Java won’t allow those declarations.

4. Why is the static block executed before the main method in Java?
Ans: The static block is executed before the main method because the Java Language Specification directed that static blocks must be executed first.

When a particular class is executed, The java virtual machine (JVM) performs
two actions during the runtime. They are as

1. At first, the class is loaded into the memory, that is, JVM loads the .class
file(in byte code) into the memory.
2. During the .class file loading into memory, the static block gets executed.
After the loading of the .class file, the main method will be called for execution.
Therefore, the static block is executed before the main method.

In other words, we can also say that the static block is stored in the memory during
the time of class loading and before the main method is executed, so the static block
is executed before the main method.

5. Why is a static method also called a class method?
Ans: A static method is a method that belongs to a class rather than an instance of a class. This means we can call a static method without creating an object of the class. Every instance of a class has access to the method. Static methods have access to class variables (static variables) without using the class's object (instance).In both static and non-static methods, static methods can be accessed directly. Since static methods can be accessed by any method of a class and also we can call it without creating an object of the class, they can also be referred to as class methods.

6. What is the use of static blocks in Java?
Ans: Static blocks are used to initialize static variables in Java. In fact, it is good practice to initialize static variables using static blocks. Also, we can use static blocks to execute some piece of code as soon as the Class is loaded.
Example:
static block used to initialize a static variable:

```
class Test{
    int age;
    static String college;
    static {
        college = "ABC";
    }
}
```

Static block used to execute before main method execution starts:

```
public class Test{
    static {
        System.out.println("in static block");
    }
    public static void main(String args[]){
        System.out.println("in main method");
    }
}
```

When the above code is run, once the class Test is loaded, the static block will be executed first and then execution starts from the main method.

7. Difference between Static and Instance variables?
Ans:
Static Variables:

- They are also known as class variables as they are properties of a class.
- These can also be called as Non-Instance variables.
- They are declared using the static keyword.
- These variables can be accessed without creating any objects of the class.

- Only one copy of a static variable exists per class no matter how many objects are created.
- Since these variables are related to the class, they are created when the class execution starts and destroyed automatically when the execution ends, i.e, without the creation of an object of the class.
- Initialization is not mandatory for such variables and their default value is 0.
- They can be accessed using the class name.
- They are stored in the Method Area of the JVM Data Area.
- To access static variables we have to use the Class' name as a reference variable.
- Only static variables can be accessed inside a static method.

Instance Variables:

- These variables are also known as Non-Static variables as they are different for each object of the class.
- They are declared inside a class but outside any method, constructor, or block.
- Since these variables are related to an instance of a class, they are created when an object of the class is created and destroyed when the object is destroyed.
- These variables can be accessed only by creating objects.
- They are stored in the Heap Area of the JVM Data Area as part of an object.
- To access instance variables we have to use the object's name as a reference variable.
- Instance variables cannot be accessed in a static method.

8. Difference between static and non-static members?

Ans:

Static Members:

Static members are those who belong to the class and you can access these members without instantiating the class. The static keyword can be used with methods, fields, classes (inner/nested), and blocks.

i. Static variables are called "class variables".
ii. Static members will get memory in the method area.
iii. For variables, if the value does not change from object to object then we need to use static variables.
iv. Inside a static area, we can access static variables only.
v. Static variables are created using static keywords.

Non-static Members:

An instance(which is a part of Non-static) member is essentially anything within a class that is not marked as static. That is, it can only be used after an instance of the class has been made (with the new keyword). This is because instance members belong to the object, whereas static members belong to the class.

i. Non-static variables are called "instance variables".
ii. Non-static members will get memory in the heap area.

iii. If the value changes from object to object then we need to use “non-static” variables
iv. Inside a nonstatic area we can access both static and non-static variables.
v. Non-static variables are created without using the “static” key.