1) **Why do we need `static` keyword in Java? Explain with an example:**

The `static` keyword in Java is used to indicate that a particular member belongs to the class itself, rather than instances of the class. This means that a `static` member is shared among all instances of the class and can be accessed without creating an instance

```
class Example {

    static int count = 0;



    Example() {

        count++;

    }



    static void displayCount() {

        System.out.println("Count: " + count);

    }

}



public class Main {

    public static void main(String[] args) {

        Example obj1 = new Example();

        Example obj2 = new Example();

        Example.displayCount(); // Output: Count: 2

    }

}
```

In this example, `count` is a static variable, meaning it is shared among all instances of the `Example` class. The `displayCount` method is a static method, which can be called without creating an instance of the class.

2) **What is class loading and how does the Java program actually execute?**

Class loading is the process by which the Java Virtual Machine (JVM) loads class files into memory. The process of executing a Java program involves several steps:

- **Loading:** The JVM loads the `.class` files into memory using the class loader.
- **Linking:** This involves verifying the bytecode, preparing static variables, and resolving references to other classes.
- **Initialization:** The JVM initializes the class, which includes executing static initializers and static blocks.
- **Execution:** The JVM executes the `main` method of the class specified by the user.

Here's a simplified flow of a Java program execution:

1. Class Loader loads the class files.
2. The JVM verifies the bytecode.
3. Static initializers and static blocks are executed.
4. The `main` method is called.

3) **Can we mark a local variable as static?**

No, we cannot mark a local variable as `static` in Java. Local variables are specific to the block of code in which they are declared and are destroyed once the block is exited. The `static` keyword can only be used with class-level variables and methods.

4) **Why is the static block executed before the main method in Java?**

The static block is executed before the `main` method because it is part of the class initialization process. When a class is loaded into memory, any static blocks are executed in the order they appear. This ensures that any necessary setup is done before the class is used, including the execution of the `main` method.

5) **Why is a static method also called a class method?**

A static method is called a class method because it belongs to the class itself rather than to any particular instance of the class. It can be invoked without creating an instance of the class, using the class name directly.

```
class MyClass {

   static void myStaticMethod() {

      System.out.println("This is a static method.");

   }

}
```

```
public class Main {

    public static void main(String[] args) {

        MyClass.myStaticMethod(); // No need to create an instance of MyClass

    }

}
```

6) What is the use of static blocks in Java?

Static blocks are used for static initializations of a class. They are useful for initializing static variables, performing setup that is required before the class is used, or executing any logic that needs to run once when the class is first loaded.

```
class Example {

    static int count;



    static {

        count = 10;

        System.out.println("Static block executed.");

    }

}



public class Main {

    public static void main(String[] args) {

        System.out.println("Count: " + Example.count);

    }

}
```

7) Difference between Static and Instance variables:

- **Static Variables:**
  - Belong to the class.
  - Shared among all instances of the class.
  - Initialized only once at the start of the execution.

        - Can be accessed using the class name.

- **Instance Variables:**
    - Belong to a specific instance of the class.
    - Each instance has its own copy.
    - Initialized each time a new instance is created.
    - Accessed using the instance of the class.

8) Difference between static and non-static members:

- **Static Members:**
    - Belong to the class rather than any instance.
    - Include static variables and static methods.
    - Can be accessed without creating an instance of the class.
    - Can access only other static members directly.
- **Non-Static Members:**
    - Belong to an instance of the class.
    - Include instance variables and instance methods.
    - Require an instance of the class to be accessed.
    - Can access both static and non-static members.

By understanding these concepts, you can effectively use static and non-static members to design your Java classes and manage shared resources efficiently.