TELUSKO

12-Static Block

What is a Static Block?

A **static block** in Java is a block of code associated with the static keyword. This block is used for the static initialization of a class and is executed only once when the class is loaded into memory for the first time. Unlike other code blocks, a static block runs before any object of the class is created and even before the constructor is called.

Characteristics of a Static Block

- **Special Java Block:** A static block is enclosed within {} and marked with the static keyword. The code inside this block executes once and is used to initialize static variables or perform startup tasks for the class.
- **Initialization:** The primary use of a static block is to initialize static variables or execute code that must run only once when the class is loaded.

Syntax of a Static Block

```
static {
    // Initialization code or static variables
}
```

Example of a Static Block

Let's look at an example to understand how a static block works:

```java
class Mobile {
    String brand;
    int price;
    static String name;

    // Constructor to initialize instance variables
    public Mobile() {
        brand = "";
        price = 200;
        name = "Phone";

        System.out.println("Inside constructor");
    }

    // Static block to initialize static variables
    static {
        name = "Phone";
        System.out.println("In static block");
    }

    public void show() {
        System.out.println("Brand: " + brand + ", Price: " + price + ", Name: " + name);
    }
}

public class Demo {
    public static void main(String[] args) {
        Mobile obj = new Mobile();
        obj.brand = "Apple";
        obj.price = 1500;
        Mobile.name = "SmartPhone";
        obj.show();
    }
}

```

TELUSKO

Explanation of the Code

1. **Class Declaration:**
    - class Mobile { ... }: This declares a class named Mobile. Inside this class, we define variables and methods that describe the characteristics and behaviors of a mobile phone.
2. **Instance Variables:**
    - String brand; int price;: These are instance variables, meaning each object of the Mobile class will have its own copy of these variables.
    - static String name;: This is a static variable, meaning all objects of the Mobile class share the same value for this variable.
3. **Constructor:**
    - public Mobile() { ... }: This is a constructor method that initializes instance variables. The constructor is called every time a new object of the class is created.
    - brand = ""; price = 200; name = "Phone";: These lines set default values for the instance variables when a new Mobile object is created.
4. **Static Block:**
    - static { ... }: This block is executed when the class is loaded into memory, before any objects are created.
    - name = "Phone"; System.out.println("In static block");: This sets the static variable name and prints a message to the console.
5. **Instance Method:**
    - public void show() { ... }: This method displays the values of the brand, price, and name variables.
6. **Main Method:**
    - public static void main(String[] args) { ... }: This is the entry point of the program. Inside this method:
    - Mobile obj = new Mobile();: A new Mobile object is created, which calls the constructor.
    - obj.brand = "Apple"; obj.price = 1500; Mobile.name = "SmartPhone";: The brand and price instance variables are updated for the obj object, and the static name variable is set for the class.
    - obj.show();: This calls the show() method to display the object's properties.

Output Explanation

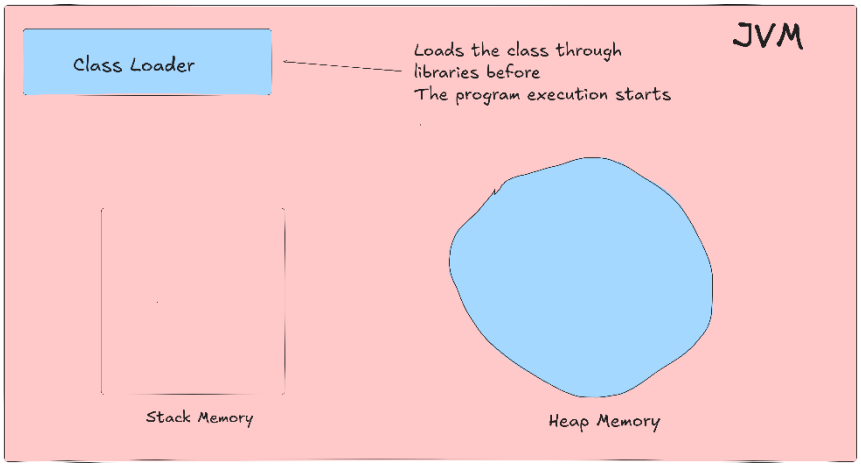
```
Output    Generated Files

In static block
Inside constructor
Brand: Apple, Price: 1500, Name: SmartPhone
```

When you run this code, you'll see the following output:

- **Static Block First:** The message "In static block" is printed first because the static block runs as soon as the class is loaded, even before the constructor.
- **Constructor Next:** The message "Inside constructor" is printed next, as the constructor is called when the object obj is created.
- **Show Method:** Finally, the show() method prints the updated values of the instance and static variables.

Understanding Static Block Execution Order

- **Static Block Before Constructor:** The static block runs before the constructor because it is associated with the class, not with individual objects. When the JVM loads the class into memory, it runs the static block to initialize any static variables.



- **Class Loading Without Object Creation:** If you want the static block to execute even without creating an object, you can force the class to load using ***Class.forName("Mobile");***. This method loads the class and triggers the static block without requiring object creation.

FAQs on Static Block

1. **Q: What is a static block in Java?**
    - A: A static block in Java is a block of code associated with the static keyword. It is used to initialize static variables or perform tasks that need to run once when the class is loaded into memory.
2. **Q: When is the static block executed?**
    - A: The static block is executed when the class is loaded into memory by the JVM, before any objects of the class are created and before the constructor is called.
3. **Q: Can a class have multiple static blocks?**
    - A: Yes, a class can have multiple static blocks. They are executed in the order they appear in the class.
4. **Q: What is the purpose of using a static block?**
    - A: The static block is primarily used to initialize static variables or execute code that should only run once when the class is loaded.
5. **Q: How can you force a static block to execute without creating an object?**
    - A: You can force the static block to execute by using Class.forName("ClassName");. This method loads the class into memory and triggers the static block.
6. **Q: Is a static block mandatory in a class?**
    - A: No, a static block is not mandatory. It is used when necessary, typically for static variable initialization.