Static and non-static blocks in java:

In Java, static and non-static blocks are special code blocks used to initialize classes and objects. They are executed during different phases of the program's execution. They are considered as equivalent of method declaration and execution at the same time

1. **Static Blocks:** Static blocks are enclosed within a class and are executed only once when the class is loaded into memory. They are used to initialize static variables and perform any static initialization tasks. Static blocks are declared using the static keyword followed by a set of curly braces.

Syntax:

```
static {
   // static initialization code
}
```

Key points about static blocks:

- Static blocks are executed in the order they appear in the code.
- They are primarily used for initializing static variables or performing complex static initialization.
- Static blocks cannot access non-static variables or methods directly because they are executed before any object of the class is created.
- Exceptions thrown within static blocks are handled by the JVM.

Example:

javaCopy code

```
class MyClass {
   static int x;
   static {
      // static initialization code
      x = 10;
      System.out.println("Static block executed");
   }
}
```

2. **Non-Static Blocks**: Non-static blocks, also known as instance initialization blocks, are used to initialize instance variables of a class. They are executed each time an object of the class is created. Non-static blocks are declared without any specific keyword, just by using a set of curly braces.

Syntax:

javaCopy code

```
{
   // instance initialization code
}
```

Key points about non-static blocks:

- Non-static blocks are executed in the order they appear in the code, but before the constructor of the class.
- They are used to initialize instance variables, perform common tasks, or implement complex initialization logic.
- Non-static blocks can access both static and non-static variables and methods of the class.
- Exceptions thrown within non-static blocks are propagated to the caller.

Example:

javaCopy code

```
class MyClass {
   int y;
   {
      // instance initialization code
      y = 20;
      System.out.println("Non-static block executed");
   }
}
```

Usage scenario: Static blocks are commonly used for initializing static variables, such as loading configuration files or establishing database connections. Non-static blocks are useful when you want to perform certain actions or calculations that need to be executed for each instance of a class before the object is created.

It's worth noting that static and non-static blocks cannot be explicitly called or invoked. They are executed automatically during the class loading or object creation process, respectively. So there no access modifier exists for static and non static block

Overall, static and non-static blocks provide flexibility in initializing classes and objects, allowing you to execute code at specific points in the program's lifecycle.

Example:

```
package util.blocks;

public class Parent {
   {
      System.out.println("executing non static block of parent");
   }
```

```
  static{
     System.out.println("executing static block of parent");
  }

  public Parent() {
     System.out.println("calling constructor of parent");
  }
}

package util.blocks;

public class Person extends Parent{




  static{
     System.out.println("executing static block");
  }

  static{
     System.out.println("executing static block second block");
  }

  {//riwo
     System.out.println("executing non static block second block");
     this.name = "No name assigned";
  }
  //method declaration and calling at the same time
  {
     System.out.println("executing non static block");
     this.name = "No name assigned";
  }


  public Person() {
     System.out.println("calling constructor");
  }

  private String name;

  public String getName() {
     return name;
  }
```



```
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                '}';
    }
}

package util.blocks;

public class PersonDemo {

    public static void main(String[] args) {
        Person person1 = new Person();
        //Person person2 = new Person();
    }
}
```

Sequence of execution of constructor and static/nonstatic blocks in java is as below.

Calling constructor for value SUNDAY
Calling constructor for value MONDAY
Calling constructor for value TUESDAY
Calling constructor for value WEDNESDAY
Calling constructor for value THURSDAY
Calling constructor for value FRIDAY
Calling constructor for value SATURDAY