JAVAGURU INTRODUCTION TO JAVA

LESSON 6

STATIC KEYWORD OVERVIEW

STATIC KEYWORD OVERVIEW

- The keyword static indicates that the particular member belongs to a type itself, rather than to an instance of that type
- Only one instance of that static member is created which is shared across all instances of class
- Can be applied to the following elements:
  - Fields (variables)
  - Methods
  - Inner methods
  - Static code block

STATIC FIELDS

- Exactly a single copy of static field is created and shared among instances of that class
- No matter how many times class is initialized.. Always single copy of static field

1. STATIC FIELDS CODE EXAMPLE: MESSAGE CLASS

```java
public class Message {

    public static int instancesCreated = 0;

    private String text;

    public Message(String text) {
        this.text = text;

        System.out.println("Creating message = '" + text + "'");
        instancesCreated++;
    }

}
```

2. STATIC FIELDS CODE EXAMPLE: MESSAGE CLASS

Code

```
System.out.println("Created = " + Message.instancesCreated);
Message greeting = new Message("Hi!");
Message question = new Message("How are you?");
Message farewell = new Message("Goodbye!");
System.out.println("Created = " + Message.instancesCreated);
```

Console output

```
Created = 0
Creating message = 'Hi!'
Creating message = 'How are you?'
Creating message = 'Goodbye!'
Created = 3
```

REASONS TO USE STATIC FIELDS

- When the value of variable is independent of objects
- When the value is supposed to be shared across all objects

KEY POINTS TO REMEMBER

- Since static fields belong to a class, they can be accessed directly using class name and don't need any object reference
- Static variables can only be declared at the class level
- Static fields can be accessed without object initialization
- Although static field can be accessed through reference, access via class name is preferred

STATIC METHODS

- Also belong to a class instead of the object
- Can be called without creating the object of the class in which they reside
- Generally used to perform an operation that is not dependent upon instance creation
- Widely used to create utility classes so that they can be obtained without creating a new object of these classes

1. STATIC METHODS CODE EXAMPLE: MATHS CLASS

```java
public class QuickMaths {

    public static int min(int[] numbers) {
        if (numbers.length == 0) {
            return 0;
        }

        int min = numbers[0];

        for (int number : numbers) {
            if (number < min) {
                min = number;
            }
        }

        return min;
    }

}
```

2. STATIC METHODS CODE EXAMPLE: MATHS CLASS

Code

```
int[] values = {44, 65, 61, 16, 89};

int result = QuickMaths.min(values);

System.out.println("result = " + result);
```

Console output

```
result = 16

Process finished with exit code 0
```

REASONS TO USE STATIC METHODS

- To access or manipulate static variables and other static members that don't depend upon objects
- Widely used in stateless utility classes

KEY POINTS TO REMEMBER

- Static methods cannot be overridden
- Instance methods can directly access both instance methods and instance variables
- Instance methods can directly access both static variables and static methods
- Static methods can access all static variables and other static methods
- Static methods cannot access instance variables and instance methods directly; only via object reference