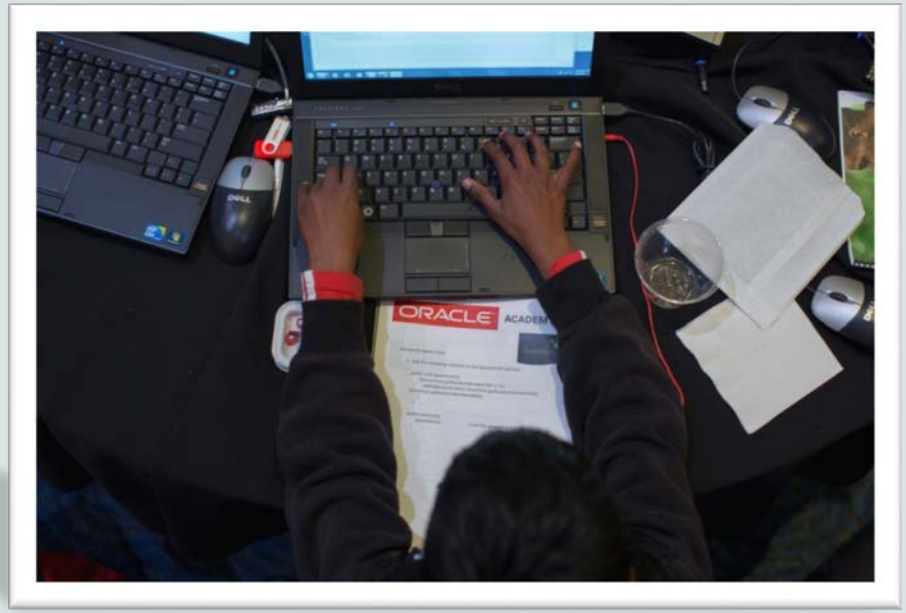

ORACLE
ACADEMY

Java Fundamentals

7-3
The Static Modifier and Nested Classes



ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Create static variables
- Use static variables
- Create static methods
- Use static methods
- Create static classes
- Use static classes

Static Modifier

- Using instance variables, each instance of a class created with the keyword new creates a copy of all instance variables in that class.
- For example, in the Employee class below, a unique copy of lastname and firstname is created for each new Employee object that is created in a Driver Class.

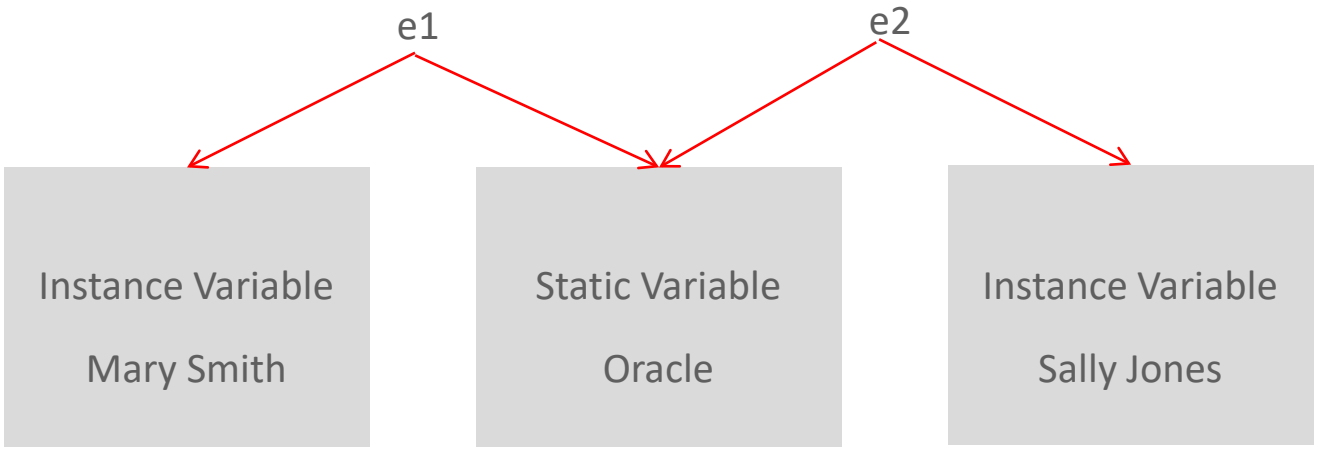
```
public class Employee{
        private String lastname;
        private String firstname;
        ...more code
}
//create two Employees in a main method:
Employee e1 = new Employee("Smith", "Mary");
Employee e2 = new Employee("Jones", "Sally");
```

Static Keyword

- Static is a keyword in Java that modifies the association of an item to a class.
- Contents of a class that are identified as static are shared across all instances of the class.
- This means all instances of the class share one copy of the static items, and each have their own unique copies of instance items, or non-static items.

Static Example

- Consider initializing a static String with the value "Oracle" called myCompany that represents the employer's company.
- Each instance of Employee would still have their unique instance variables, but would share the static variable.



Static Variables

- Static variables
- Are also known as class variables.
- Are declared with the static keyword.
- Have only one copy in memory, as opposed to instance variables, which hold one copy per instance.
- Are shared by object instances.
- Hold the same value for all class instances.

Static Variables

- Public access for static variables:
  - If public, they can be modified directly by other classes.
  - Consider making the variable a constant by using the keyword final to prevent modifications.
  - Example:

```
public static final int MODEL_NUM = 883;
```

Programming Practices and Static Variables

- Good programming practice initializes static variables with values, rather than relying on the default null and 0 values.
- The values initially assigned can be changed as long as the class is active in JVM memory.
- Garbage collection removes it from memory and the initial values assigned will return the next time you use it.

Declaring a Static Variable

- To declare a static variable, include the keyword static as shown below.
- Can be public, protected, default, or private.
- Should have assigned values, but automatically are assigned null values for class instances: an empty string or 0 for primitive numbers.
- Should act as constants with the final keyword when they use a public, protected, or default access.

```
public class Nesting {
  // Declare public static variable.
  public static final int MODEL_NUM = 883;
  ...
}
```

Changes to a Static Variable

- Static variables that are not final can be read or assigned new values by using the optional keyword **this** in instance methods.
- Changes by instance methods are changed for all instances.
- A change to a static variable may indicate that the class should be limited to only one object.
- This is known as the Singleton pattern.

```
...
private static String myCompany = "Oracle";

public void setMyCompany(String s) {
  this.mycompany = s;
}
...
```

Static Variable Example

- Create a class called Turtle that contains a variable named food. This variable is static since all of our turtles eat the same food.
- The Turtle class will have one more variable named age.
- Since each turtle is a different age, it is best to make this variable a private instance variable rather than a static one.

```
public class Turtle {
   public static String food = "Turtle Food";
   private int age;

  public Turtle(int age){
   this.age = age;
  }
}
```

Accessing Static Variables

- Instance variables require an instance of the class to exist before access is possible.

```
public class Turtle {
   public static String food = "Turtle Feed";
   private int age;
   ...
}
```

- You can access static variables without creating an instance of the class.
- In a main method, this statement would print out the variable food without any instance reference.

```
System.out.println("I feed " + Turtle.food + " to all of my turtles!");
```

Notation to Access Static Variables

- Generally, static variables are accessed by the notation:

```
ClassName.variableName;
```

Static Modifier and Methods

- Static or class methods exist once and are shared by all class instances.
- May be used by other class methods or instance methods based on their access modifier.
- Cannot access non-static, or instance, variables. Static methods can only access static variables.
- Cannot access non-static, or instance, methods. Static methods can only access other static methods.
- Can be redefined in subclasses.
- Can be public, protected, default, or private.

Static Modifier and Methods

- There are differences between calling an instance method versus a class (static) method.
- For example, you must first create an instance and then use a dot notation to call an instance method; whereas, the class name, a dot notation, and static method name calls a static method.

Static Modifier and Methods

- The static method provides a wrapper to construct an instance of a class.
- When the class has a private access constructor, a static method is one of two approaches to creating an instance of the class.

Turtle Class Example

- The Turtle class has a static variable that identifies the number of tanks we have available (numTanks) and an instance variable (tankNum) that tells us which tank the Turtle is in.

```
public class Turtle {
   public static String food = "Turtle Feed";
   public int age;
   public int tankNum;
   public static int numTanks = 3;
   public Turtle(int age){
      this.age = age;
      tankNum = (int)((Math.random()*numTanks)+1);
   }
   public void swim(){//implementation}
   public int getAge(){return age;}
   public int getTankOfResidence(){return tankNum;}
   public static String fishTank() {return "I have " + numTanks + " fish tanks.";
  }
}
```

tankNum variable

numTanks variable

Static Methods in Turtle Class Example

- Review the methods in the Turtle class.

```
public class Turtle {
   public static String food = "Turtle Feed";
   public int age;
   public int tankNum;
   public static int numTanks = 3;
   public Turtle(int age){
      this.age = age;
      tankNum = (int)((Math.random()*numTanks)+1);
   }
   public void swim(){//implementation}
   public int getAge(){return age;}
   public int getTankOfResidence(){return tankNum;}
   public static String fishTank() {return "I have " + numTanks + " fish tanks.";
  }
}
```

swim() is a instance method. Although each turtle can swim, the turtles may swim differently depending on their age.

fishTank() is a static method and it accesses a static variable (numTanks).

getAge() and getTankOfResidence() are instance, non-static, methods because they access non-static variables. Static methods cannot access non-static items.

Creating Class Instances Using Static Methods

- Another use of static methods is for creating class instances when the class constructor access is private, and the method is part of the same class.
- This is possible because the static method is publicly accessible with private access to the class.

```
...
private Nesting() {…implementation…}
...
public static Nesting getInstance() {
  Nesting nesting = new Nesting();
  return nesting; }
...
// Instantiate a private class with a method.
Nesting n1 = Nesting.getInstance();
...
```

Static Modifier and Classes

- Static or nested classes:
  - Can exist as nested classes.
  - Cannot exist as independent classes.

A nested class is a class that is created inside another a class.

Static Nested Classes

- Static nested classes
  - Are implemented inside other classes, and the other classes are known as container classes.
  - Can extend the behavior of the container class.
  - Can be overloaded like ordinary constructors.

Static Nested Classes

- The static nested class also provides the means for instantiating a containing class when it's constructor is configured with private access.
- This is the second way to instantiate a class that has a restricted or private access qualifier for its class constructors.

Static Nested Classes Example

```
public class Space {
  //Space class variables

  public static class Planet{
    //planet class variables and constructors
    public Planet() {...implementation...}
    public Planet(String name, int size)
    {...implementation...}
  }

  //more space class implementation
}
```

Terminology

Key terms used in this lesson included:

- Class method
- Class variable
- Inner class
- Nested class
- Static modifier
- Static method
- Static nested class
- Static variable

Summary

In this lesson, you should have learned how to:

- Create static variables
- Use static variables
- Create static methods
- Use static methods
- Create static classes
- Use static classes


ORACLE
ACADEMY