Java Object Oriented Programming

Static fields and methods

Hiba ALQASIR

2021-2022

télécom saint-étienne

école d'ingénieurs / nouvelles technologies

Static keyword

Static keyword applies to:

- fields
- methods
- initialization blocks
- nested classes

Memory management

Static field

- Also called *class variable*
- Common to all the objects, not unique for each object
- Can be accessed without having to create an instance of a class
- Can be accessed directly in static and non-static methods
- Saves memory (gets memory only once at the time of class loading)

Static field

```java
public class MyClass {

    int numObjectsMade = 0;

    MyClass() {
        numObjectsMade += 1;
    }

    public static void main(String[] args) {

        MyClass obj1 = new MyClass();
        System.out.println(obj1.numObjectsMade);
        MyClass obj2 = new MyClass();
        System.out.println(obj2.numObjectsMade);
        MyClass obj3 = new MyClass();
        System.out.println(obj3.numObjectsMade);
    }
}
```

Output

```
1
1
1
```

Static field

```java
public class MyClass {

    static int numObjectsMade = 0;

    MyClass() {
        numObjectsMade += 1;
    }

    public static void main(String[] args) {

        MyClass obj1 = new MyClass();
        System.out.println(MyClass.numObjectsMade);
        MyClass obj2 = new MyClass();
        System.out.println(MyClass.numObjectsMade);
        MyClass obj3 = new MyClass();
        System.out.println(MyClass.numObjectsMade);
    }
}
```

Output

```
1
2
3
```

Constants

A variable preceded by **static** and **final** modifiers.

```
1 public class MyClass {
2
3     static final String I_AM_CONSTANT = "I cannot be reassigned";
4
5     public static void main(String[] args) {
6         System.out.println(I_AM_CONSTANT);
7         I_AM_CONSTANT = "Whatever";
8     }
9 }
```

Output

```
Exception in thread "main" java.lang.Error:  Unresolved compila-
tion problem:
The final field MyClass.I_AM_CONSTANT cannot be assigned
```

Static method

- Also called *class method*
- Belongs to the class rather than object of a class
- Can only access static data
- Can be invoked without the need for creating an instance of a class

Static method

Non-static methods can reference static variables

```
1  public class MyClass {
2
3      static String name = "class variable";
4
5      void whoami() {
6          System.out.println(name);
7      }
8
9      public static void main(String[] args) {
10         MyClass obj = new MyClass();
11         obj.whoami();
12     }
13 }
```

Output

```
class variable
```

Static method

But not the other way around !
Static methods can NOT reference non-static variables

```java
public class MyClass {

    String name = "instance variable";

    static void whoami() {
        System.out.println(name);
    }

    public static void main(String[] args) {
        MyClass.whoami();
    }
}
```

Output

```
Exception in thread "main" java.lang.Error:  Unresolved
compilation problem:
cannot make a static reference to the non-static field name
```

Static method

```java
public class MyClass {

    String name = "instance variable";

    static void whoami(MyClass obj) {
        System.out.println(obj.name);
    }

    public static void main(String[] args) {
        MyClass obj = new MyClass();
        MyClass.whoami(obj);
    }
}
```

Output

```
instance variable
```

Static method

Why is the **main** method static?

```
1 public static void main(String[] arguments) {
2     // ...
3 }
```

- The main() method is where the compiler starts the program execution.
- The compiler should be able to call it before or without the creation of an object of the class.

Static method

What if the static modifier is removed from the signature of the **main** method?

```
1 public void main(String[] arguments) {
2     // ...
3 }
```

- Program compiles.
- However, at runtime:

Output

```
Error:  Main method is not static in class td4.MyClass, please
define the main method as:
public static void main(String[] args)
```

Static initialization block

- Used to initialize static data members
- Executed before the main method at the time of class loading

Static initialization block

```
1  class MyClass{
2
3      static {
4          System.out.println("This is a static block");
5      }
6
7      public static void main(String args[]){
8          System.out.println("This is the main method");
9      }
10 }
```

Output

```
This is a static block
This is the main method
```

Static nested class

A static class that is created inside another class

- Can be accessed by outer class name
- Cannot access non-static data members and methods
- Can access static data members of the outer class, including private

Static nested class

```java
public class OuterClass {

    private static String name = "class variable";

    static class InnerClass {
        void whoami() {
            System.out.println(name);
        }
    }

    public static void main(String args[]) {
        OuterClass.InnerClass obj = new OuterClass.InnerClass();
        obj.whoami();
    }
}
```

Output

```
class variable
```

Static nested class

```java
public class OuterClass {

    private String name = "instance variable";

    static class InnerClass {
        void whoami() {
            System.out.println(name);
        }
    }

    public static void main(String args[]) {
        OuterClass.InnerClass obj = new OuterClass.InnerClass();
        obj.whoami();
    }
}
```

Output

```
Exception in thread "main" java.lang.Error:  Unresolved compi-
lation problem:
Cannot make a static reference to the non-static field name
```

Exercise #1

Consider the following class:

```
1 public class Rectangle {
2     static int width;
3     int height;
4     int area(){
5         return width * height;
6     }
7 }
```

- What are the class variables?
- What are the instance variables?

Exercise #1

Consider the following main method in the class Rectangle, what's the output?

```
1 public static void main(String[] args) {
2     Rectangle r;
3     r.width = 4;
4     r.height = 5;
5     System.out.println("The rectangle's area is " + r.area());
6 }
```

Try to answer before compiling and executing.

Exercise #1

Consider the following main method in the class Rectangle, what's the output?

```
1  public static void main(String[] args) {
2      Rectangle r1 = new Rectangle();
3      r1.width = 1;
4      r1.height = 5;
5      Rectangle r2 = new Rectangle();
6      r2.width = 2;
7      r2.height = 6;
8      System.out.println("r1 = " + r1.height + " x " + r1.width);
9      System.out.println("r2 = " + r2.height + " x " + r2.width);
10     System.out.println("Rectangle.width = " + Rectangle.width);
11 }
```

Try to answer before compiling and executing it.

Exercise #2

Write a static method that takes three boolean parameters and returns true if at least two of them are true and false otherwise. *Do not use an if statement.*

```
1 public static void main(String[] args) {
2     System.out.println( method1(true, true, false) );
3     System.out.println( method1(true, false, true) );
4     System.out.println( method1(true, false, false) );
5     System.out.println( method1(false, false, true) );
6 }
```

Output:

```
true
true
false
false
```

télécom saint-étienne

Exercise #3

Write a static method that takes two integer arrays as arguments and returns true if they both have the same elements in the same order.

```
1  public static void main(String[] args) {
2      int[] a = { 1, 11, 7, 31, 5 };
3      int[] b = { 1, 11, 7, 31 };
4      int[] c = { 1, 11, 7, 31, 5 };
5      int[] d = { 5, 11, 7, 31, 5 };
6      System.out.println(method2(a, a));
7      System.out.println(method2(a, b));
8      System.out.println(method2(a, c));
9      System.out.println(method2(a, d));
10 }
```

Output:

```
true
false
true
false
```