Classes

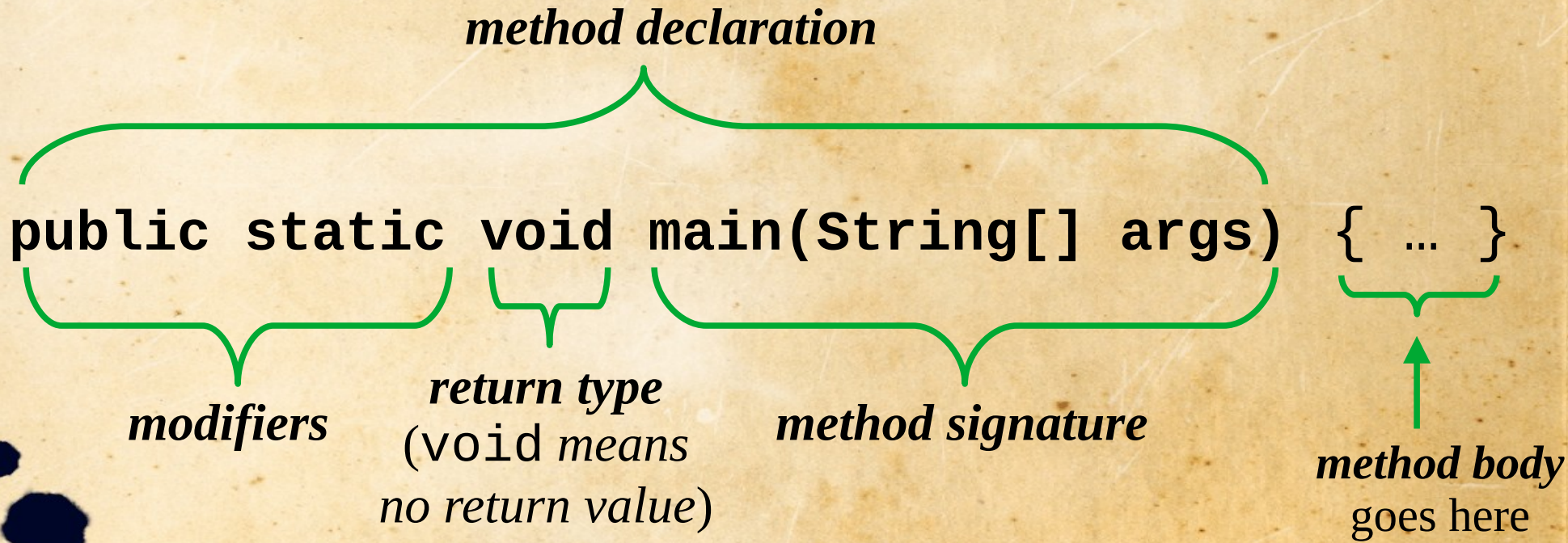
Writing `public` `static` Methods

Lecture Contents

- Review
  - Method Declaration Syntax
  - `static` Methods
  - Parameters
  - Return Values

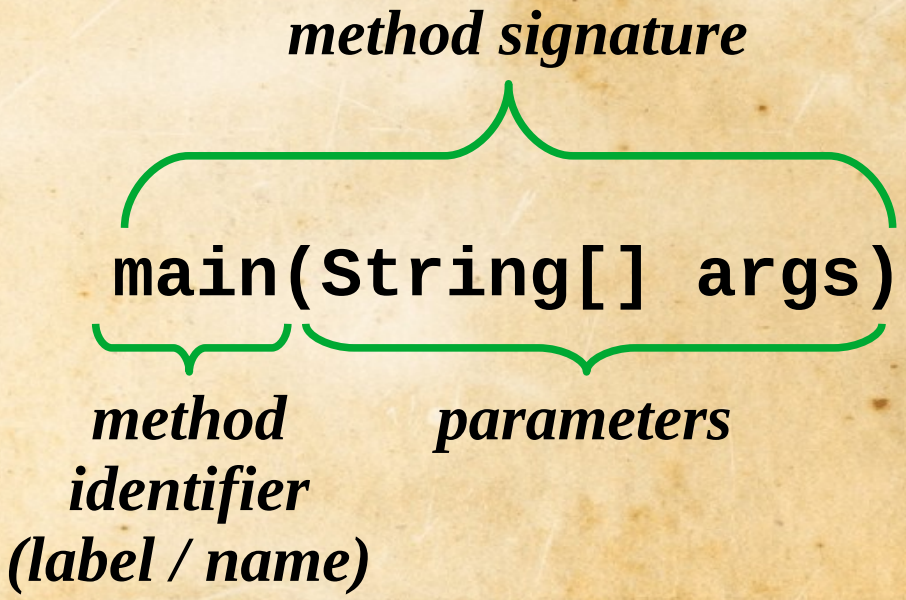
Method Declaration Syntax

- These are the parts of the method declaration.



Method Declaration Syntax

- These are the parts of the ***method signature***.
  - The method signature must be unique within a class.



Calling a Simple `static` Method

- First examine this method header.
  - What *type* of value does this method return?

```
public static void myMethod()
```

Calling a Simple `static` Method

- First examine this method header.
  - What *type* of value does this method return?
  - What is the *method identifier* for this method?

```
public static void myMethod()
```

this method does not return any value

Calling a Simple `static` Method

- First examine this method header.
  - What *type* of value does this method return?
  - What is the *method identifier* for this method?
  - What *type* and *how many* parameters does this method take?

method identifier is **`myMethod`**

```
public static void myMethod()
```

this method does not return any value

Calling a Simple `static` Method

- First examine this method header.
  - What *type* of value does this method return?
  - What is the *method identifier* for this method?
  - What *type* and *how many* parameters does this method take?

method identifier is **`myMethod`**

```
public static void myMethod()
```

this method does not return any value

this method takes no parameters

Calling a Simple `static` Method

- First examine this method header.
  - What is the *method identifier* for this method?
  - What is the ***method signature*** for this method?

method identifier
is **`myMethod`**

```
public static void myMethod()
```

method signature
includes method identifier
and parameter list

Calling a Simple `static` Method

- For a Java program, execution **starts** with the `main` method,
- Execution **stops** when the end of the `main` method is reached.

```
public class HelloWorld {

   public static void main(String[] args) {
      System.out.println("Main program.");
   }

}
```

```
Main program.
```

Calling a Simple `static` Method

- For a Java program, execution **starts** with the `main` method,
- Execution **stops** when the end of the `main` method is reached.

```
public class HelloWorld {

   public static void main(String[] args) {
      System.out.println("Main program.");
   }

   public static void myMethod( ) {
      System.out.println("Hello World!");
   }
}
```

```
Main program.
```

Calling a Simple `static` Method

- Methods in a class may be placed in any order.
  - The `main` method may come before or after other methods
  - Methods are within a class, NOT within another method

```
public class HelloWorld {

   public static void myMethod( ) {
      System.out.println("Hello World!");
   }

   public static void main(String[] args) {
      System.out.println("Main program.");
   }
}
```

```
Main program.
```

Calling a Simple `static` Method

- In the code below, the method `myMethod` is ***called*** using the method ***identifier***, followed by a set of parentheses.

```
public class HelloWorld {

   public static void main(String[] args) {
      System.out.println("Main program.");
      myMethod();
   }

   public static void myMethod( ) {
      System.out.println("Hello World!");
   }
}
```

Calling a Simple `static` Method

- In the code below, the method `myMethod` is ***called*** using the method ***identifier***, followed by a set of parentheses.

```
public class HelloWorld {

   public static void main(String[] args) {
      System.out.println("Main program.");
      myMethod();
   }

   public static void myMethod( ) {
      System.out.println("Hello World!");
   }
}
```

Calling a Simple `static` Method

- In the code below, the method `myMethod` is ***called*** using the method ***identifier***, followed by a set of parentheses.

```
public class HelloWorld {

   public static void main(String[] args) {
      System.out.println("Main program.");
      myMethod();
   }

   public static void myMethod( ) {
      System.out.println("Hello World!");
   }
}
```

```
Main program.
Hello World!
```

Calling a Simple `static` Method

- When the end of a method is reached, the program flow returns to where the method was called from, and continues execution.

```
public class HelloWorld {

  public static void main(String[] args) {
     System.out.println("Main program.");
     myMethod();
     System.out.println("End of program.");
  }

  public static void myMethod( ) {
     System.out.println("Hello World!");
  }
}
```

```
Main program.
Hello World!
End of program.
```

Calling a Simple `static` Method

- When a `return` statement is encountered, the method does not continue. Program control returns to where the method was called from.

```
public class HelloWorld {

  public static void main(String[] args) {
     System.out.println("Main program.");
     myMethod();
     System.out.println("End of program.");
  }

  public static void myMethod( ) {
     System.out.println("Hello World!");
     return;
     System.out.println("Not printed.");
  }
}
```

Calling a Simple `static` Method

- When a `return` statement is encountered, the method does not continue. Program control returns to where the method was called from.

```
public class HelloWorld {

  public static void main(String[] args) {
     System.out.println("Main program.");
     myMethod();
     System.out.println("End of program.");
  }

  public static void myMethod( ) {
     System.out.println("Hello World!");
     return;
     System.out.println("Not printed.");
  }
}
```

```
Main program.
Hello World!
End of program.
```

Note: the compiler might highlight the error and not allow this code to run.

Lecture Contents

- Review
  - `static` Methods
  - **Methods with Parameters**
  - Return Values

Review of Declaring a Variable

- We declare a local variable using the syntax:
  <*type*> <*identifier*>
- Examples:

```
int x;
String name;
```

- Or we can declare and initialize the variable:

```
double circumference = 2 * Math.PI * 2.5;
String str = "Hello World!";
```

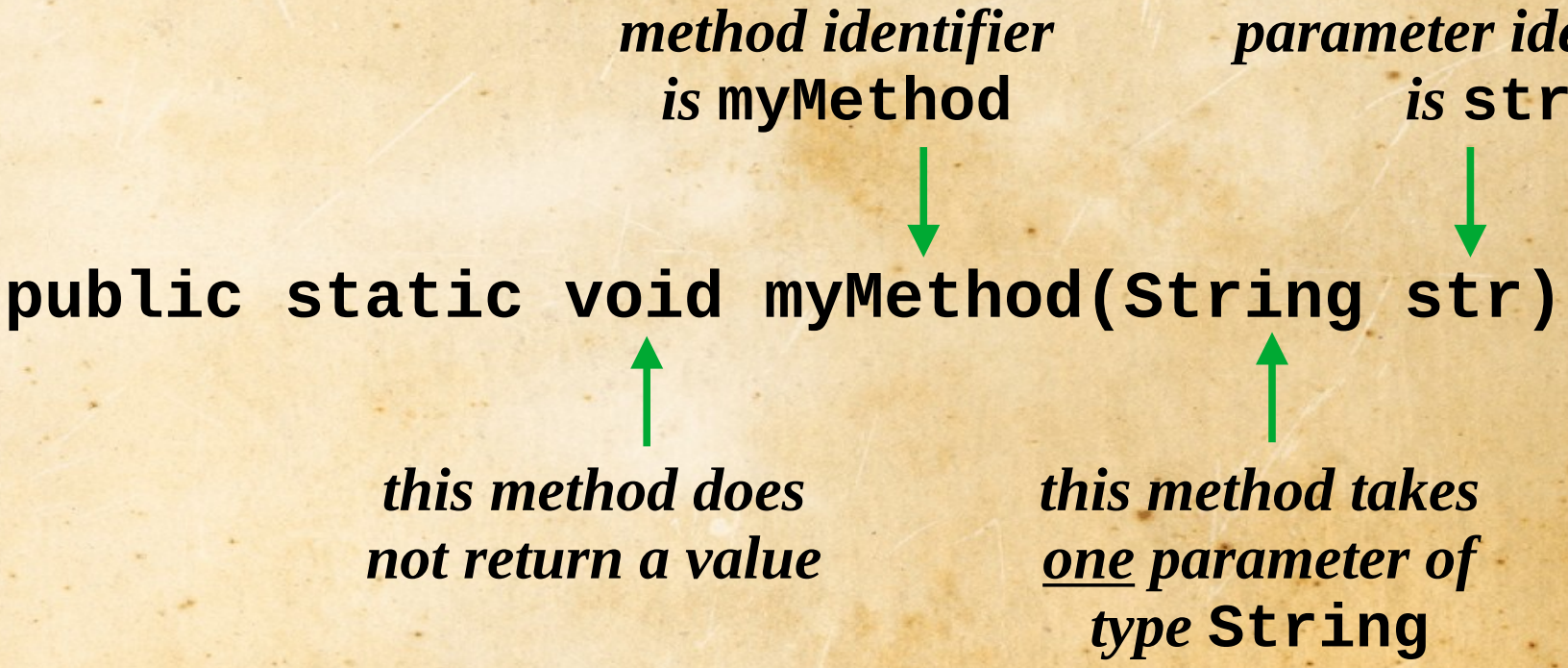
Method Parameters

- Parameters are defined in the *method signature*.
- The parameter list is enclosed in parentheses and comes after the *method identifier*.
- A ***parameter*** is a local variable.
  - Notice the parameter declaration looks like any other variable declaration

```
public static void myMethod(String str)
```

Calling a Simple `static` Method

- Examine this method header.



A Method with One Parameter

```
public class HelloWorld {

   public static void main(String[] args) {
        myMethod("Hello World!");
   }

   public static void myMethod(String s ) {
      System.out.println(s);
      System.out.println(s);
   }
}
```

A Method with One Parameter

```
public class HelloWorld {

   public static void main(String[] args) {
        myMethod("Hello World!");
   }

   public static void myMethod(String s ) {
      System.out.println(s);
      System.out.println(s);
   }
}
```

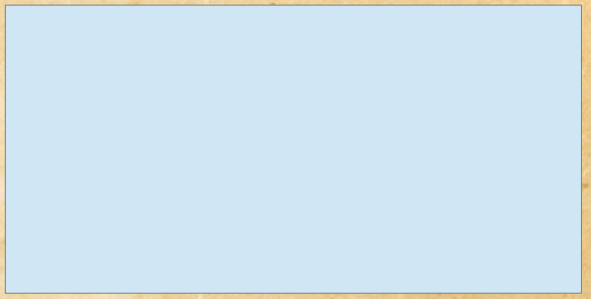
```
Hello World!
Hello World!
```

Multiple Parameters

```
public class HelloWorld {

    public static void main(String[] args) {
        printTime(3,30);
    }

    public static void printTime(int hour, int min) {
        System.out.println(hour + ":" + min);
    }
}
```



Multiple Parameters

```
public class HelloWorld {

    public static void main(String[] args) {
        printTime(3,30);
    }

    public static void printTime(int hour, int min) {
        System.out.println(hour + ":" + min);
    }
}
```

```
3:30
```

Write the method “writeSquare”

```
public class HelloWorld {

   public static void main(String[] args) {
        int i = 3;
        writeSquare(i);
        writeSquare(i+2);
   }

   public static void writeSquare(int x) {
      // TODO: Write the code for this method!!
   }
}
```

```
The square of 3 is 9.
The square of 5 is 25.
```

Write the method “writeSquare”

```
public class HelloWorld {

   public static void main(String[] args) {
        int i = 3;
        writeSquare(i);
        writeSquare(i+2);
   }

   public static void writeSquare(int x) {
      System.out.println("The square of " + x + " is " + x*x + ".");
   }
}
```

```
The square of 3 is 9.
The square of 5 is 25.
```

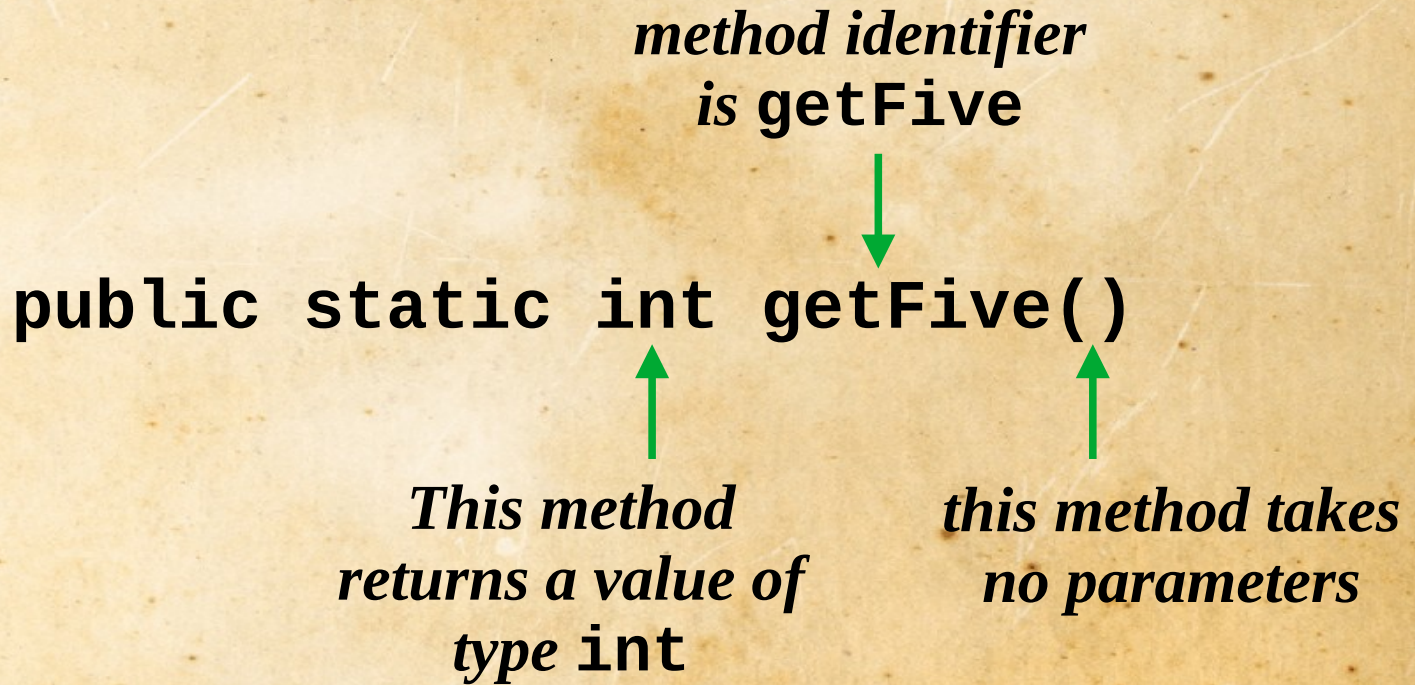
Calling a Simple `static` Method

- Examine this method header.
  - What *type* of value does this method return
  - What is the method identifier?
  - How many parameters does this method take?

```
public static int getFive()
```

Calling a Simple `static` Method

- Examine this method header.



Method Return Values

```
public class HelloWorld {

   public static void main(String[] args) {
      int i = getFive();
      System.out.println("The method returned " + i + ".");
    }

   public static int getFive() {
      return 5;
   }
}
```

```
The method returned 5.
```

Write the method “getSquare”

```
public class HelloWorld {

   public static void main(String[] args) {
      int i = 3;
      int j = getSquare(i);
      System.out.println("The square of " + i + " is " + j + ".");
    }

   public static int getSquare(int x) {
      // TODO: Write the code for this method!!
   }
}
```

```
The square of 3 is 9.
```

Write the method “getSquare”

```
public class HelloWorld {

   public static void main(String[] args) {
      int i = 3;
      int j = getSquare(i);
      System.out.println("The square of " + i + " is " + j + ".");
    }

   public static int getSquare(int x) {
      return x * x;
   }
}
```

```
The square of 3 is 9.
```

Practice: `calculateBMI` method

- A person's body mass index (BMI) is calculated by the person's weight in kilograms divided by the square of the person's heigh.

$$BMI = \frac{weight\,[kg]}{(height\,[m])^2}$$

- Write a method `calculateBMI` that takes two double parameters – `weight` and `height`, and calculates the person's BMI.

Classes

Writing `public` `static` Methods