Beyond the Single main() Method

- Many classes can be written using only one main() method call, containing all our code.
- Many classes will have more than one method.
- Some classes we write won’t have a main() method.
- In fact, much code you have written is built this way…

You might only write a single main() method, but:

1. Your code may use classes written by the instructor, like Oval, Rectangle, etc., that don’t have their own main() methods.
2. Your code may use built-in classes, like String, Scanner, etc., that also don’t have their own main() methods.

In each such case, you have used many other methods, like setLocation() or length().

Now it’s time to create/write your own methods to do things.

Method Syntax

- Methods are placed inside of classes
- They are not placed inside of other methods: each is a separate code block.
- Each can have its own variable declarations, with self-contained scope, which means that they are not visible outside of the method.

Multiple Methods in a Class

- We can write as many methods in a class as we like.
- Each will start with its own method declaration, and will contain a block of valid code.
- Each must be complete before another method starts.

```java
public class MethodRunner {
    public void sayHello() {
        System.out.println("Hello!");
    }
    private void sayGoodbye() {
        System.out.println("Goodbye!");
    }
}
```
private vs. public Methods

- Methods can be either **public** or **private**
  - This **does not** affect compiling the code at all
  - This **does not** affect what the code actually does at all
  - Instead, it affects **where** we can legally **access** (i.e., **call**) it

```java
public class MethodRunner {
    public void sayHello() {
        System.out.println("Hello!");
    }
    private void sayGoodbye() {
        System.out.println("Goodbye!");
    }
}
```

**A method with public access**

**A method with private access**

private vs. public Method Access

- If we make a method in a class C **private**, it can only be called from within the class C itself
- If we make a method in C **public**, it can be called:
  1. from within the class C itself
  2. by **any other** object of type C (created using the C() constructor)

```java
public class MethodRunner {
    public void sayHello() {
        // code here
    }
    private void sayGoodbye() {
        // code here
    }
}
```

```
public class Foo {
    public static void main(String[] args) {
        MethodRunner runner = new MethodRunner();
        runner.sayHello();
    }
}
```

**Since sayHello() is public, we can call it here in the separate Foo class**

public vs. private Method Access

- **In this example**, the public method can be called in **any other class** in the usual way:
  1. Instantiate object of class type
  2. Call method we wish to use

```java
public class MethodRunner {
    public void sayHello() {
        // code here
    }
    private void sayGoodbye() {
        // code here
    }
}
```

```
public class Foo {
    public static void main(String[] args) {
        MethodRunner runner = new MethodRunner();
        runner.sayHello();
    }
}
```

**Since sayHello() is public, we can call it here in the separate Foo class**

```
public class MethodRunner {
    public void sayHello() {
        // code here
    }
    private void sayGoodbye() {
        // code here
    }
}
```

```
public class Foo {
    public static void main(String[] args) {
        MethodRunner runner = new MethodRunner();
        runner.sayHello();
    }
}
```

**A private method can’t be called in any other class, even if we create objects and try**

```java
public class MethodRunner {
    public void sayHello() {
        // code here
    }
    private void sayGoodbye() {
        // code here
    }
}
```

```
public class Bar {
    public static void main(String[] args) {
        MethodRunner runner = new MethodRunner();
        runner.sayGoodbye(); // ERROR!!
    }
}
```

**sayGoodbye() is private, and so we can’t call it here**

This code results in a **compile error**
**public vs. private Method Access**

- Inside a single class, it **doesn't matter** whether methods are public or private.
- **Any other** code can call **every method** in the same class.
- Since we are inside the same class, we don't need an object to call the method.
- The compiler/JVM automatically looks for any methods we call this way inside the class itself.

```java
public class MethodRunner {
  public void sayHello() {
    // code here
  }
  private void sayGoodbye() {
    // code here
  }
  private void sayBoth() {
    sayHello();
    sayGoodbye();
  }
}
```

Here, we can **directly call either** of the other methods.

**Exercise**

- Download the code example for Class 32.
- Create a new Java project called MethodRunner and import the Java files into the src folder.
- Create a new Java class called MethodRunner3.
- Using MethodRunner as a template, write methods in your new class. They must include a constructor and at least two other methods each of which prints something to the screen. One should be public and the other private.
- In Main.java, create a MethodRunner3 object and call your methods. Observe what happens.

**Constructors**

- Constructor method
  - Used to create an object
  - Has same name as class
  - Has no return type – this is NOT the same as void

```java
DrawingGizmo<br>  <<constructor>>
DrawingGizmo();
<<update>>
void moveForward();
void turnClockwise();
void turnCounterclockwise();
void dontDraw();
void draw();
void delay2Sec();
```

**Using Constructors**

- How do you call a constructor?
  - You've already done so many times!
  - Use **new**
  - Oval o = new Oval();
  - Random rand = new Random();
  - Window win = new Window();
  - String s = new String();
  - Car Giulia = new Car();

These are calls to the constructors for the Oval, Random, Window, String, and Car classes.
Multiple Constructors

- Can have multiple constructors in a class: 
  - Must have different parameter lists 
  - Java "knows" which one to use based on the parameter list you provide in the call.

<table>
<thead>
<tr>
<th>DrawingGizmo</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;constructor&gt;&gt;</td>
</tr>
<tr>
<td>DrawingGizmo()</td>
</tr>
<tr>
<td>DrawingGizmo(Color, Color)</td>
</tr>
<tr>
<td>&lt;&lt;update&gt;&gt;</td>
</tr>
<tr>
<td>void moveForward()</td>
</tr>
<tr>
<td>void turnClockwise()</td>
</tr>
<tr>
<td>void turnCounterclockwise()</td>
</tr>
<tr>
<td>void dontDraw()</td>
</tr>
<tr>
<td>void draw()</td>
</tr>
<tr>
<td>void delay2Sec()</td>
</tr>
</tbody>
</table>

Multiple Constructors

Car Giulia = new Car();

**Calls a constructor that takes no parameters.**

Car MC40 = new Car("Mini", "Cooper S", 2004, 25000, 180, 190);

**Calls a constructor that takes six parameters:** String, String, int, int, int, int

This Week & Next

- **Meetings this week:**
  - Monday/Wednesday: Lab assignments
  - Tuesday/Friday: Recorded lectures

- **Reading 05:** Ch. 6 due Friday April 3 at 5:00 PM
- **Reading 06:** Ch. 7 due Friday April 10 at 5:00 PM
- **Program 04:** due Tuesday April 7 at 11:59 PM
- **Office Hours:** via the interwebs
  - Monday/Tuesday/Wednesday/Friday: 9:00 AM–11:00 AM
  - https://kube-0.cs.uwlax.edu:8443/ZombieApocalypseOfficeHours