Java Refresh & *static*

CSCU9A3
Data Structures, Objects and Algorithms

David Cairns

Derived from *Big Java* by Cay Horstmann
John Wiley & Sons

HelloPrinter

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      // Send a greeting to standard output

      System.out.println("Hello, World!");
   }
}
```

Output:

```
Hello World!
```

Statements

- The body of the main method contains statements inside the curly brackets `{}`
  - Each statement ends in a semicolon `;`
  - Statements are executed one by one
  - Our method has a single statement:

```
System.out.println("Hello, World!");
```

which prints a line of text:

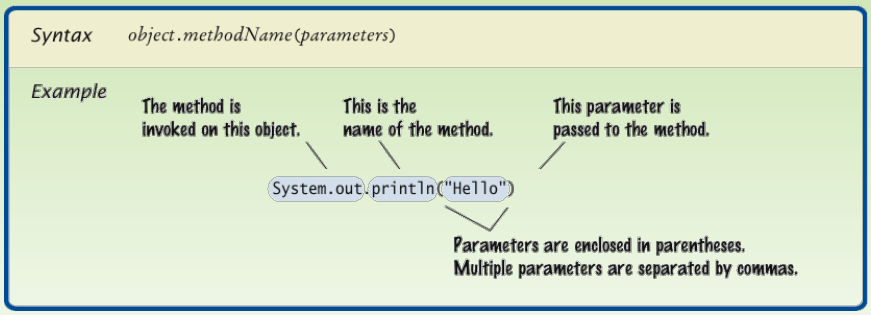
```
Hello, World
```


A Method Call

```
System.out.println("Hello, World!");
```

- This a method call. It requires:
  - The object that you want to use - System.out
  - The name of the method you want to use - println
  - Parameters enclosed in parentheses () containing any other information the method needs
    - in this case "Hello, World!" which is a parameter of type String

A Method Call



Errors

- Two key types of error – Compile Time Error & Run Time Error
  - Compile Time
    - Usually a Syntax Error
    - Sometimes Unknown Package/Library
    - Example: 2 Syntax Errors:

```
System.ou.println(“Hello World!);
```

  - Run Time Error
    - These happen when you run the program
    - Causes the program to take an action that the programmer did not intend – a logic error
    - An error in your logic, not the computer's
      - It just does what you tell it to…

```
System.out.println("Hello, Word!");
System.out.println(1/0);
```

The Program Development Process

1. Understand the Problem
2. Develop and describe an algorithm
3. Test the algorithm with different inputs
4. Translate the algorithm into code (e.g. Java, C++,C#)
5. Compile and test the program
   - Check to make sure it works with different inputs
   - Check it solves the original problem

Class Files (Byte Code)

- What do you expect to see when you load a class file into your text editor?

Class Files (Byte Code)

Answer: A sequence of random characters, some funny looking. Class files contain virtual machine instructions that are encoded as binary numbers.

A text editor will try to display a character that corresponds to the equivalent binary numbers. Many of these numbers correspond to invisible characters and therefore do not display properly.

Program Failures

- When you used your computer, you may have experienced a program that
  - "crashed"
    - quit unexpectedly
  - "hung"
    - failed to respond to your input
- Is that behavior a compile-time error or a run-time error?

Testing for Run Time Errors

- Why can't you test a program for run-time errors when it has compile time errors?

Testing for Run Time Errors

Answer: When a program has compiler errors, no class file is produced, and there is nothing to run.

Structure of Simple Java Programs

- The previous 'Hello World' example is one of the simplest programs you can write in Java but its structure is not a good basis for testing programming ideas. You are highly likely to fall into the static trap…

Example : The *static* trap

```
public class StaticHello
{
     public static void main(String[] args)
     {
        go(); // Our problem starts here...
     }

     public void go()
     {
          // Start your real code here...
          print("Hello, World!");
          print("What next?");
     }

     public void print(String message)
     {
          System.out.println(message);
     }
}
```

Correct Program Structure

```
public class Hello
{
   public static void main(String[] args)
   {
       // Create an instance of an object we want to use
       Hello h = new Hello();
       // Now call one of its methods
       h.go();
   }

     public void go()
     {
          // Start your real code here...
          print("Hello, World!");
          print("What next?");
     }

     public void print(String message)
     {
          System.out.println(message);
     }
}
```

Use the above code as the basis for testing out programming concepts, e.g. parameter passing, returning values (instead of using void), loops etc.

Problems with *static* variables

```
public class Fruit
{
     private String name;
     private static String colour;

     // Our constructor...
     public Fruit(String nm, String col)
     {
          name = nm;
          colour = col;
     }

     // You can create toString methods for your classes that
     // describe them. You can then just call println or similar
     // methods with a reference to your object and it will use
     // the toString method to get a String version of your object.
     public String toString()
     {
          return "Name:" + name + " Colour: " + colour;
     }
}
```

Problems with *static* variables

```
public class Bowl
{
   public static void main(String[] args)
   {
        Bowl b = new Bowl();
        b.go();
   }

   public void go()
   {
        Fruit banana = new Fruit("Banana","Yellow");
        System.out.println(banana);

        Fruit strawberry = new Fruit("Strawberry","Red");
        System.out.println(strawberry);

        System.out.println("All looks good so far...");
        System.out.println();

        System.out.println(banana);
   }
}
```


Problems with *static* variables

The strange behaviour occurred due to the declaration of the attribute colour as static. This results in all Fruit objects sharing just the one colour attribute. If you change the value of colour in one Fruit object, all Fruit objects will have the same value for colour.

```
private static String colour;
```

It should just be:

```
private String colour;
```

You are most likely to have tried to make something static in order to keep the compiler happy. If you use the correct code structure, the compile issue will not arise.

main method

- The method 'main' is special. The *virtual machine* will look for a `main` method when it tries to start a program. The goal of `main` is to act as a starting point for your program.
- Because of the role it plays, it is static
  - There is only ever one instance of a static method and it is associated with the class it appears in, not the objects that are made from that class.
  - static methods are more generally used where you do not need to create an instance of an object just to use a useful method it has
    - For example, it is a lot less bother to write

      ```
      s = Math.sin(0.3);
      ```

    - than

      ```
      Math m = new Math();
      s = m.sin(0.3);
      ```

    - The sin method in the Math class is static
  - You should generally avoid creating your own static methods or variables for now.