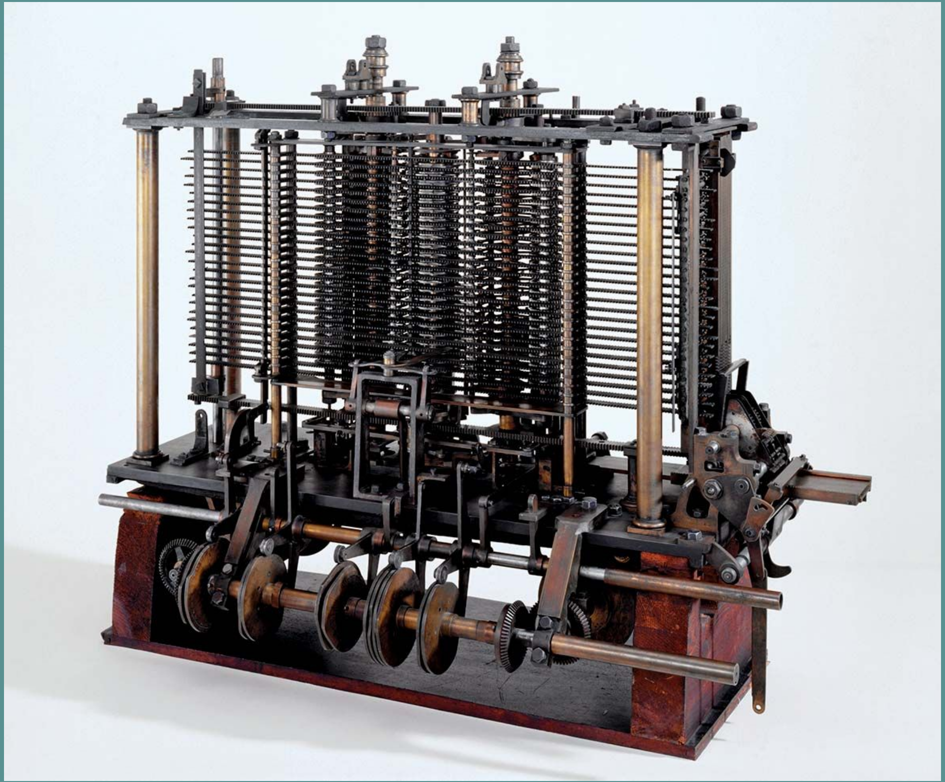
GWC Week 5

Review of Fundamentals and Methods

WIT Shout-Out of the Week: Agusta Ada Lovelace

- Born December 10, 1815 Ada was an English Mathematician and writer who is most known for her work on Charles Babbage’s mechanical general-purpose computer, the Analytical Engine.
- She wrote the first algorithm intended to be carried out by a machine and pioneered the ideas that would help shape the ideas of computational machines and their abilities and opportunities
- She opened the discussion to how the Analytical Engine could be used to help in calculations such as Bernouli Numbers (calculations to help predict a series of events often used in statistics)
- “The Enchantress of Numbers” - Charles Babbage



The Analytical Engine

Ada Lovelace

Video

https://www.youtube.com/watch?v=uBbVbqRvqTM

Be Brave, Not Perfect

https://www.ted.com/talks/reshma_saujani_teach_girls_bravery_not_perfection?language=en

Warm-Up

- Write a program that will ask a user for the base and length and then computes the area of a triangle
  - Hint: make sure to use int.Parse() to type cast your user input from string/char to an integer
- Write a program that asks a user for any word
  - Then loop through the word and print out the word line by line, except if you come across a vowel, then replace it with a “*”

Data Types

Data Types

What are some examples of data types we have seen?

Data Types

- int → integers (aka whole numbers)
- double → decimal numbers
- float → decimal numbers
- bool → true/false
- string → series of characters
- char → one single character

Creating a Variable

Unlike in Python, in C# you MUST SPECIFY THE VARIABLE DATA TYPE when creating a variable. This is because C#, like Java, is **strong-typed**.

```
int x = 4;
double y = 4.56;
string input = “Hello World”;
```

Switching Between Data Types -- Type Cast

Sometimes, we need to switch a data type from one type to another.

Ex: when you use Console.ReadLine(), it returns a **string** type. If you asked for a number from the user, you need to convert the **string** to an **int**, otherwise YOU WILL RECEIVE AN ERROR.

How? Use int.Parse(). The string you want to convert to an integer goes INSIDE the parentheses.

- Note: You can also use double.Parse() to get decimal numbers

Switching Between Data Types -- Type Cast

Switching Data Types is not always necessary

Ex: going from int to double or double to int
→ going from int to double = UNNECESSARY
→ going from double to int = NECESSARY

This is because if you go from double to int, you are LOSING DATA (double is a decimal, int is a whole number, so switching from double to int cuts off the decimal part of the double)

Switching Between Data Types

Another technique: Casting
- Casting allows you to switch certain related data types

If I want to turn a decimal into a whole number, I have to CAST it to a different type.

```
double x = 3.44;  What is the
int y = (int)x;   value of y?
```

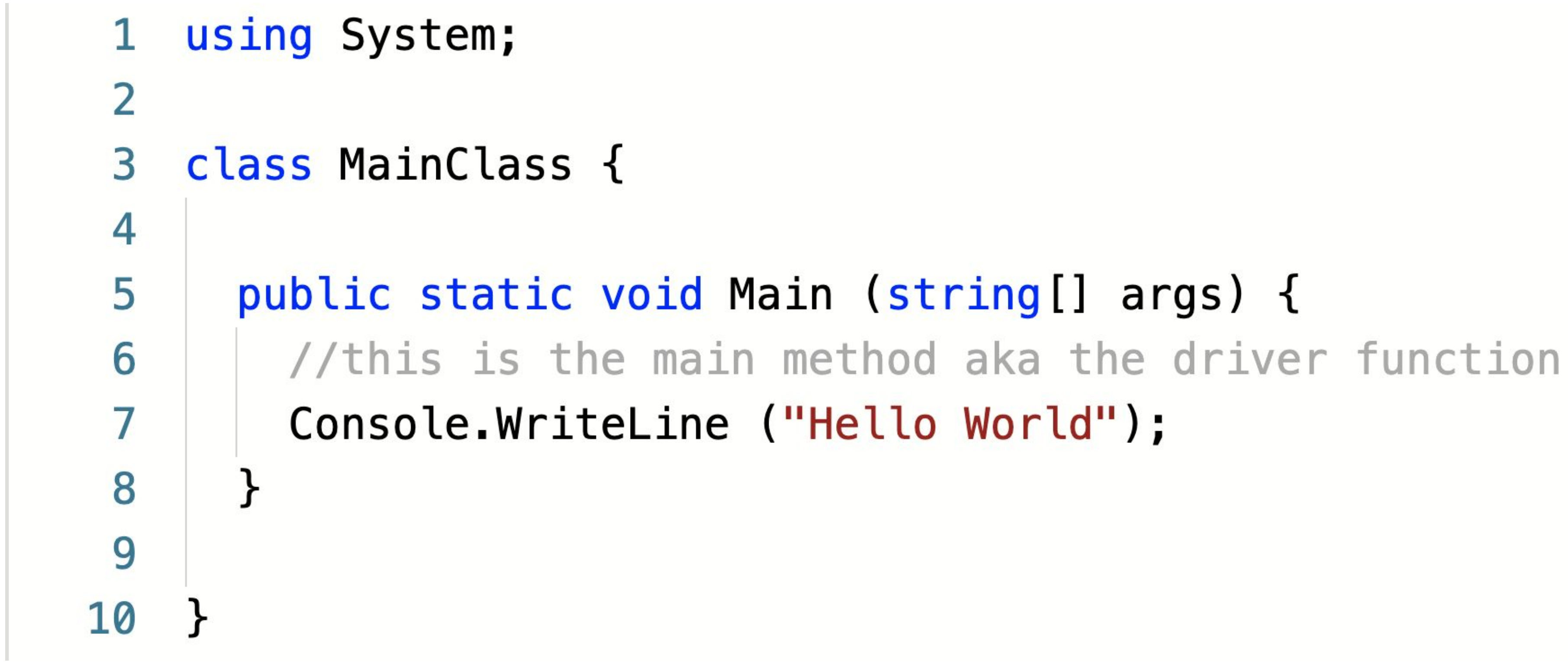
Methods (aka Functions)

Methods (aka Functions)

- Containers of code that allow you to perform a specific portion of code that is ***reusable***
- Methods for a program are contained inside of a **Class** but we will discuss those more later
- We’ve already seen several different methods
  - Console.Write()
  - Console.Read()
  - Main (string[] args)

Main Methods: Driver Programs

- In each C# program that you create the Main method is the one special container of code that is used to execute the overarching program.
- The main method is where you “Call” or execute the other methods (or functions) that you created to use to run the entire program
- ***<u>For every program that you create you will need to create a main method</u>***
- In Repl.it, the main method is the first one that you see:



```csharp
using System;

class MainClass {

  public static void Main (string[] args) {
    //this is the main method aka the driver function
    Console.WriteLine ("Hello World");
  }

}
```

This is a main method, you can tell because it’s named “Main” and is makes use of the (string[] args) parameters

Parts of a Method

- Methods are pieces of code that are run by receiving both input and giving output
  1. **input** comes either from ***parameters*** or from user input or defined input
  2. **Output** for a method is data given back based on a ***return-type***
- Methods also make use of a ***Signature*** or method name that is “called” later to execute the portion of code you’ve defined in a method

Method Signature (Method Name)

- Each Method you write needs a signature or name that you can reference later to run the code inside of that method
- Names should be related to purpose of the method
- camelCase or use_underscores

```
public static int methodName(int a, int b) {
   // body
}
```

Parts of a Method: Return Type

- Each method you write will either return a value or not
- Since C# is strong typed like Java, we must define what type of value the method will return (ex: int, double,float, string)
- Methods that do not return anything but simply execute some operation will be a **void** return type

```
public static int methodName(int a, int b) {
    // body
}
```

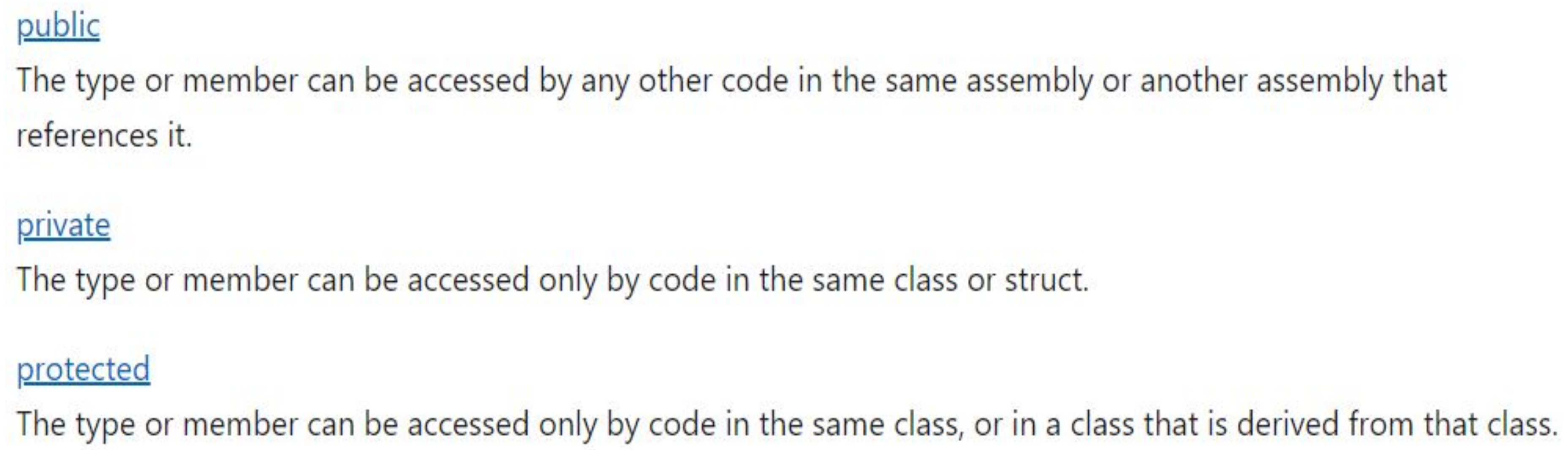
Return Type

- Think back to data types from last week
  - int
  - double
  - bool
  - string
  - char
  - etc.

```
public static double methodName(int a, int b)
{
    // body
}
```

Parts of a Method: Access Modifiers (or property types)

- Each method that you write should be defined by a what is called an <u>***Access Modifier:***</u> the level to which this portion of code can be accessed by other parts of the program



public

The type or member can be accessed by any other code in the same assembly or another assembly that references it.

private

The type or member can be accessed only by code in the same class or struct.

protected

The type or member can be accessed only by code in the same class, or in a class that is derived from that class.

Method Examples

```
public static int methodName(int a, int b) {
   // body
}
```

```
public static double methodName(int a, int b) {
   // body
}
```

```
public static string methodName(int a, int b) {
   // body
}
```

Parameters

```
public static int methodName(int a, int b)
{
   // body
}
```

- This is the data that will be used in the method that are passed in to the method -- they are specified inside of the parenthesis next to the method name
- Must declare a data type for the values (above we use int)
- Can have multiple parameters
  - Just put a comma between them as shown above
- Can have 0 parameters
  - Ex: Console.ReadLine();

Parameters (continued)

```
public static int methodName(int a, int b)
{
   // body
}
```

- You only specify the data type when CREATING the method, when you call it in your main method you just include the variables
- Parameters are a way to PASS DATA BETWEEN METHODS

```
int x = 0;
int y = 3;
sum(x,y)
```

How to Call a Method

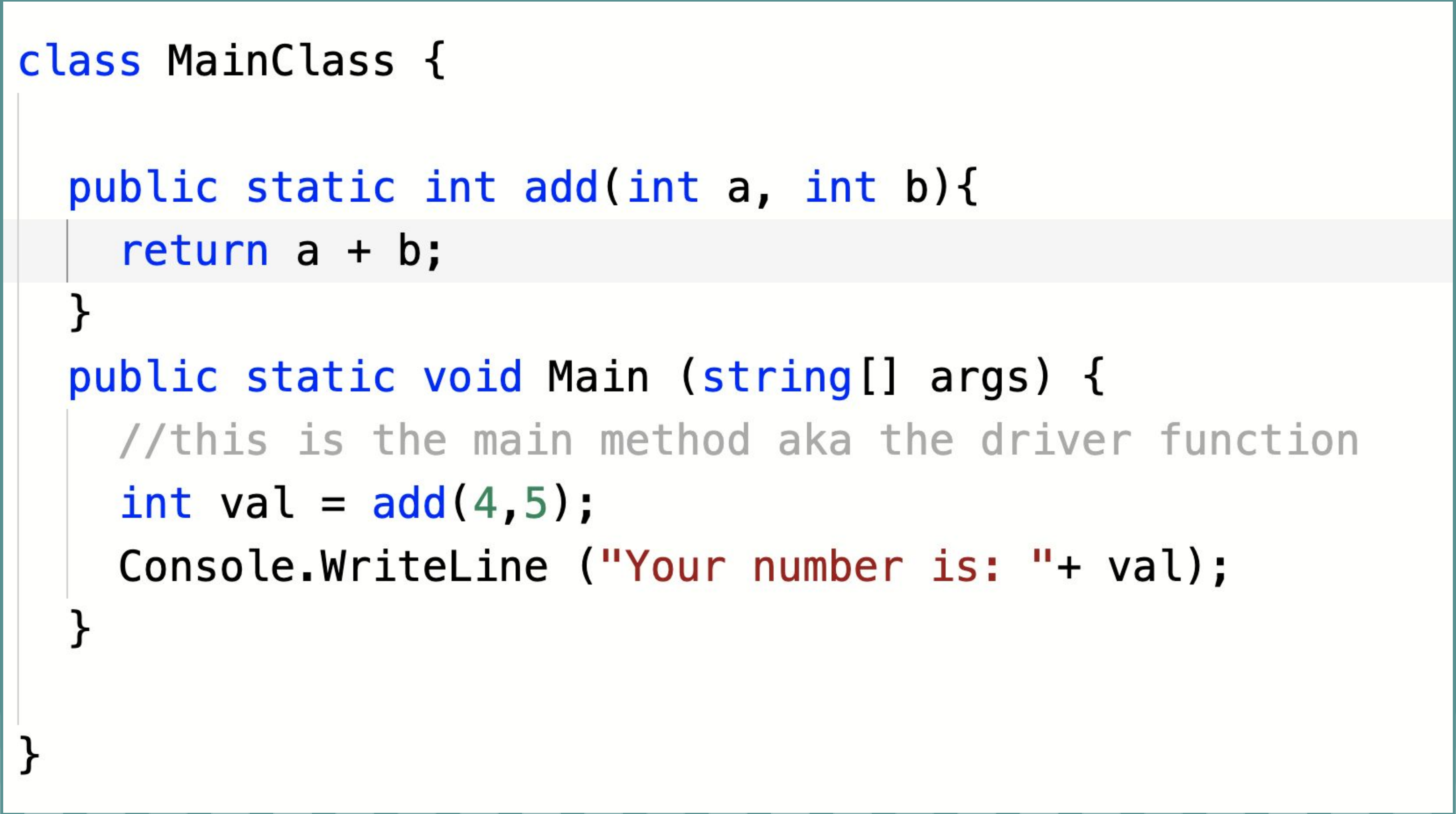
- To ***"Call"*** or execute/run a method we use the method name and any parameters that it requires
- Methods can be called in the main method, another method, or the same method (recursion)
- For example we call our add function from the previous example in the main method using:

```
add(4,5)
```

OR use variables: `add(x,y)`

Example: Addition Method

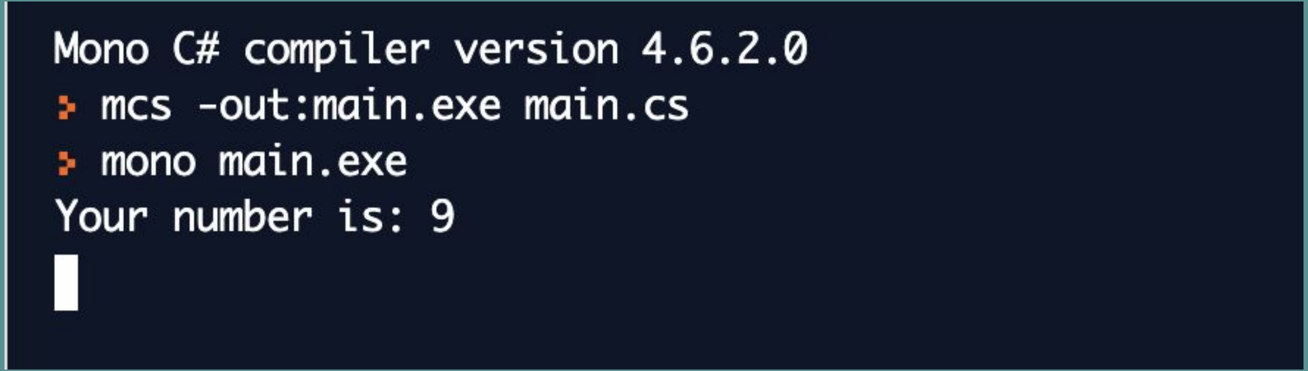
- Property/Access Modifier: we use **public** because we want to access this method in all other programs or methods that we create
- Method name: add (because we are going to add things together in this method)
- Use two or more parameters, but for this one we will just use two
- Code Body: add the two parameter variables together
- Return value: return the final answer



```
class MainClass {


  public static int add(int a, int b){
    return a + b;
  }
  public static void Main (string[] args) {
    //this is the main method aka the driver function
    int val = add(4,5);
    Console.WriteLine ("Your number is: "+ val);
  }

}
```



```
Mono C# compiler version 4.6.2.0
> mcs -out:main.exe main.cs
> mono main.exe
Your number is: 9
```

Activities

- We will be writing a calculator program

1. Write a method for each of the following math operations
   - Add
   - Subtract
   - Divide
   - Multiply
2. Write your main method so that someone can choose a number between 1 and 4 and then based on the number they choose it will run one of the math operations above
   a. Each math operation should get a user input for the parameters