

# GWC Week 6

Methods

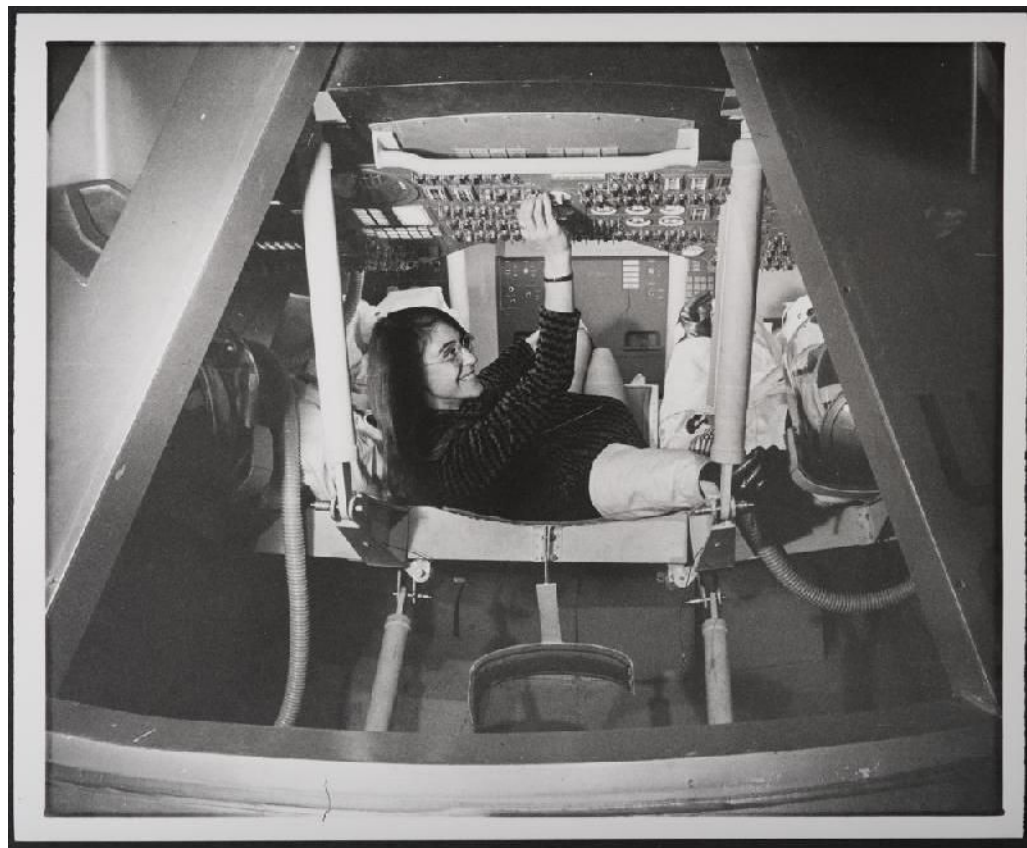


# WIT Shout-Out of the Week

## Margaret Heafield Hamilton



- An American Computer Scientist, Hamilton earned a bachelor's degree in mathematics from Earlham College in 1958
- Instead of attending graduate school she accepted a position developing software predicting weather in the meteorology department at MIT where her work helped make contributions to mathematics such as Chaos Theory
- Later she wrote software to help track unfriendly aircraft at MIT for the US Air Force
- She worked on the team developing software for the Apollo Space Mission with Nasa



# Video





## Data Types Refresher

- Must declare data type when CREATING variable, but not when CALLING variable
  - Int, double, boolean, char, string
- Typecasting for strings → use `.Parse()`;
  - Ex: `int x = int.Parse(s);` // s is a string
- Typecasting for numbers → use casting
  - Ex: `int x = (int)y;` // y is a double



# **Class Warm Up**





## Warm-up

- Write a C# program that takes the radius of a circle as input and calculate the perimeter and area of the circle.
- Write a C# program that takes three letters as input and display them in reverse order.

# Methods

See Week 5 Slides







## 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
- **For every program that you create you will need to create a main method**
- In Repl.it, the main method is the first one that you see:

```
1  using System;
2
3  class MainClass {
4
5      public static void Main (string[] args) {
6          //this is the main method aka the driver function
7          Console.WriteLine ("Hello World");
8      }
9
10 }
```

This is a main method, you can tell because it's named "Main" and it 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

Property of  
Marquette  
University

- 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 **Access Modifier**: 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)

```
int x = 0;
```

```
int y = 3;
```

```
sum(x,y)
```



# Method Example

```
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





# **Class Activity**





We will be writing a calculator program

1. Write a method for each of the following math operations
  - Add
  - Subtract
  - Multiply
  - Divide
2. Write your main method so that someone can enter +, -, \*, or /, and the correct mathematical operation will occur
  - a. Each math operation should get a user input for the parameters



**Your Turn**



## How to generate a Random Number

```
Random random = new Random();  
int randomnum = random.Next(0,6);
```



## Activities: Dice Rolling Simulation

- Write a program that asks a user for a “yes” or “no” answer to the question: Would you like to roll the dice? In a while loop until a person says “no”
- If a user says yes, then call a method named “roll” that will generate a random number between 0 and 6 and then return that integer ***this method should be public***
- If a user says no, then the program quits
- Use the random number generator code from the previous slide to help you with the roll method