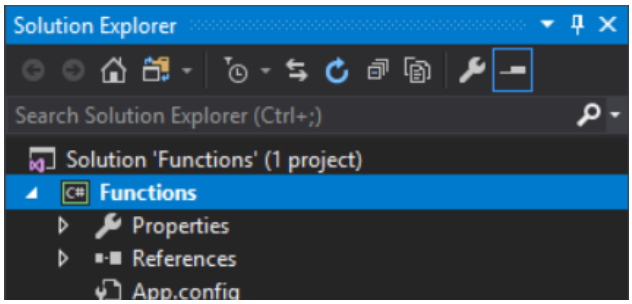
Review Sheet – Functions And Intro To Classes

Making Classes

To make a new class, go to the **Solution Explorer** on the right hand side of your screen and **right mouse click** on the name of your project**.** You'll see a pop-up menu. Go to:



1. Add
2. Class
3. Name your class **Game.cs**
4. Click **Add**

Making Functions

Example function:

```
public void Start(int data)
{
    // Code to execute
}
```

Start is the function's name. The **parentheses** are for **arguments** it can be left empty.
void means it doesn't return any values.
public means other classes / files can use this function.
The code between the **curly braces** is what runs when you use a function.

privacy return value Name(arguments)
{
Run the code here
}

Intro To Classes

A class is a template from which you can make copies of it as variables, called **instances** or **objects.**
To make a new instance of a class:

ClassName instanceName **=** new **ClassName();**

```
Game game = new Game();
```

The above code says, "Make a new **object** from the **Game class template.**"

```
game.Start();
```

The above code uses the **public Start** function from the **game** variable instance.

Functions

Functions are re-usable chunks of code that you can call upon by entering the functions name.

```
public void Start()
        {
            MonsterEncounter();
            MonsterAttacks();
            MonsterAttacks();
        }
```

The above code calls the **MonsterEncounter** function once, and the **MonsterAttacks** function 2 times. Using functions this way saves time, makes your code easier to read, and means can fix errors and make changes in less places.

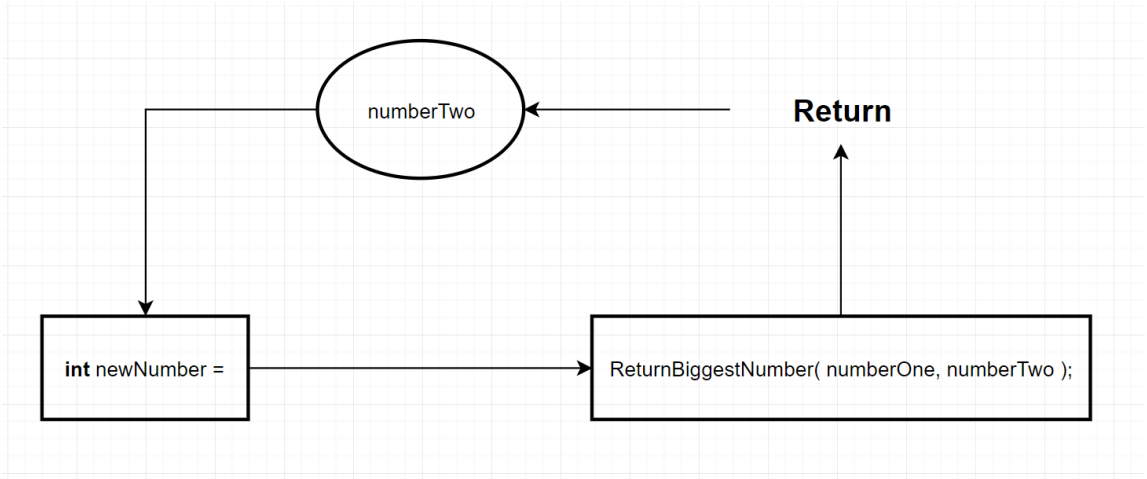
```
        public void MonsterAttacks (int data)
        {
              int combo;
              // Code to execute
        }
```

Above, **int data** is an argument and temporary variable. It only exists while the function runs and cannot be used outside of the function. **int combo** is known as a **local variable** and also only exists within the function: it cannot be used anywhere else.

```
MonsterAttacks(monsterDamage * 2);
```

When you use a function that takes **arguments**, whatever values or variables you put between the **parentheses** are given to the **temporary** variable that's created in the **argument.** In the code above, the temporary variable **int data** from the **MonsterAttacks** function would be set to the value of **monsterDamage * 2.**

Returning Values



```
int biggestNumber = ReturnBiggestNumber(numberOne, numberTwo);
```

The above int **biggestNumber** will be set to whatever value the function **ReturnBiggestNumber** returns. It could also look like:

```
int biggestNumber = value returned by ReturnBiggestNumber;
```

or

```
int biggestNumber = numberTwo;
```

or

```
int biggestNumber = 10
```

The above examples are assuming the value of the variable **numberTwo** is what is returned by the function, or assuming that the value of **numberTwo** is **10**.

Naming Conventions

Variables are named with **camelCase.** Functions and classes are named with **PascalCase.**

PascalCase example: **SwampMonster** or **PlayerController**
camelCase example: **swampMonster** or **playerController**