



Logols Learning

WEEKEND WEB DEVELOPMENT BOOT CAMP

TRAINING: C#



Visual Studio Code

- Let's take a tour of Visual Studio Code
 - To open code, you only open a folder that the code exists in.
 - Click the File menu and Open Folder option to open a folder.
 - There is an Explorer window used to view files. Use the view menu to open the explorer window.
 - Double click a file in the Explorer window to open it.
 - Multiple windows will open in different tabs.
 - Click the X on the tab to close a tab.
 - Right click and choose close all to close all tabs.
 - Use the output window to view code output. Use the view menu to open the output window.
 - There is a built in terminal window. Use the view menu to open the terminal window.
 - You can open multiple terminal windows by clicking the + button.

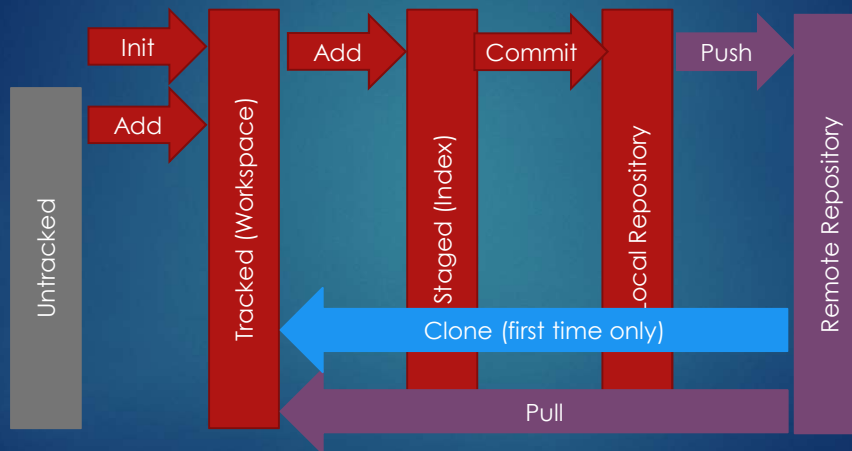
Git

- ▶ Version Control System (VCS)
- ▶ Repository
 - ▶ Central location for code
 - ▶ Keeps a history
 - ▶ Different versions of code
- ▶ Github – hosts repositories



- Git is a version control system.
- There are many different version control systems, but Git is probably one of the most popular right now.
- In Git, everything is contained inside a repository.
- A remote repository is a central location for code.
 - This allows for different developers working on the same project to be able to collaborate together and share files and changes.
- Git keeps a history of all changes including files added, modified, and deleted.
- Git allows for each developer to be working on a slightly different version of the code on their local computer and to commit their differences to the remote repository when they are ready.
- Github is a web based service that hosts Git repositories.

Git Workflow



- This is the main workflow for Git.
- There are additional steps that could be added, but this should be everything you need to know to get started.
- You can start a new repository by initializing it or you can take down an existing remote repository by cloning it to your machine.
- All files start out untracked, which means Git does not know about them.
- Once the repository is initialized or cloned, files can be added so they will be tracked by Git.
 - Tracked means that Git will keep track of modifications to a file.
- Once a modification has occurred to a file you can stage that modification along with other file modifications.
- When a commit is done, all staged modifications will go into your local repository.
- Notice that everything so far is just on your local computer. No one else can see changes.
- Once you are ready you can push commits to the remote repository.
- At any time you can pull from the remote repository to get updates.



Git / Github

- Let's setup a Github repository that you can use for this bootcamp.
 - Go to <https://github.com>.
 - Sign in or setup an account
 - Click the start a new project button
 - Enter a name and click the initialize this repository with a README check box.
 - Click the Create a Repository button.
- Clone the repository to setup a local repository and working directory.
 - Open Visual Studio Code
 - Press ctrl + shift + p to open the command palette
 - Type Git: clone
 - In Github click the Clone or download button and copy the url from the url text box.
 - Paste the url into Visual Studio Code
 - Select the directory to clone to
- Create directories needed, commit, and push to the remote repository
 - Open the directory that you cloned to in Visual Studio Code
 - Add a directory named Day1
 - Under that add a directory named TestGit
 - Under that add a file named testfile.txt
 - Click on the Source Control Window

- You should see a change was picked up for our new file.
- Click the plus icon next to the file to stage the modification
- In the text box, type the comment, "first file" and press ctrl + Enter
- Press the ellipse button (three dots) and then click push branch
 - Login to Github if needed
- Refresh the Github page and you should now see your page
- Make a change to the text of the text file and go through the same steps to push the change to the remote repository
- Look at the Github page to see the change.
- Click the ellipse and select pull to pull down changes.
- You will repeat this process to add day 1 examples to the Day1 directory and other examples to directories corresponding to the day of the session.
- Add gitignore
 - You do not want all changes in git.
 - When you build, .net creates a bunch of files. You don't need the build files.
 - Use a .gitignore file to tell Git not to track files.
 - Copy the text from this file:
<https://github.com/github/gitignore/blob/master/VisualStudio.gitignore>
 - Paste it into notepad and save a .gitignore in your directory that you cloned to.
 - Follow prior steps to push this file to the remote repository.

.Net Command Line Interface (CLI)

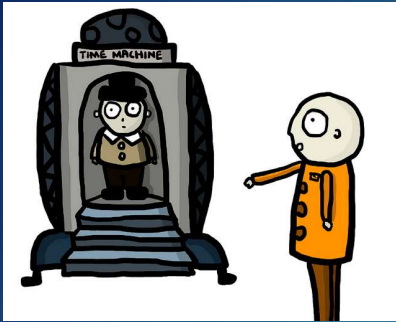
- ▶ Commands within the command line
- ▶ Entered in the terminal window
- ▶ Basic Commands
 - ▶ new, build, run
- ▶ Project Modification Commands
 - ▶ add/remove package, add/remove reference

- The .Net Core Command Line Interface (CLI) allows you to enter .Net commands into the command line.
- We will use the terminal built into Visual Studio Code for ease of use.
- There are some basic commands such as the following:
 - new – used to create new projects, files, or solutions
 - build – used to build a project.
 - run – used to run a project.
- There are also commands to modify a project such as the following:
 - add package – used to add a Nuget package to a project.
 - remove package – used to remove a Nuget package from a project.
 - add reference – used to add a reference to a project.
 - remove reference – used to remove a reference from a project.
- More information is available here: <https://docs.microsoft.com/en-us/dotnet/core/tools/?tabs=netcore2x>

CLI new Examples

- ▶ mkdir – create directory
- ▶ cd – change directory
- ▶ Console project:
 - ▶ dotnet new console
- ▶ Class Library project:
 - ▶ dotnet new classlib
- ▶ Web API project:
 - ▶ dotnet new webapi

- Here are some examples of using new within the CLI to create new projects.
- Console project:
 - dotnet new console
- Class Library project:
 - dotnet new classlib
- Web API project:
 - dotnet new webapi
- A full reference is available here: <https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet-new?tabs=netcore2x>



EXAMPLE

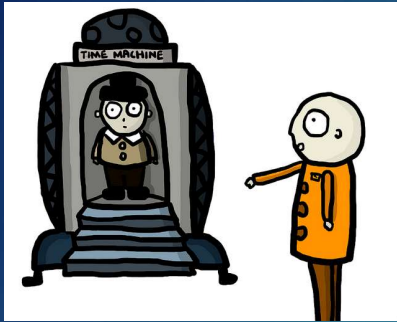
NEW CONSOLE APPLICATION IN VISUAL STUDIO CODE

Let's go through an example.

Statements

- ▶ Made up of:
 - ▶ Keywords
 - ▶ Expressions
 - ▶ Operators
- ▶ Statements end with a Semicolon ;
- ▶ Statements can span multiple lines
- ▶ Statement blocks contain multiple statements
 - ▶ Surrounded by curly braces { }
 - ▶ Can have blocks within blocks

- Statements are made up of keywords, expressions, and operators.
 - Keywords are known words that are part of the language and perform some type of designation.
 - Expressions are a combination of operators and operands. The operands can be values or variables.
 - Operators perform some action, such as in math we have operators like +, -, *, /
- In C# statements end with a semicolon ;
- Statements can span multiple lines, but must always have the semicolon at the end of the statement.
- A statement block is a block of multiple statements. These statements are surrounded by curly braces { } and are blocked together to designate the statements are grouped for some reason.
- Statement blocks can exist within other statement blocks.



EXAMPLE

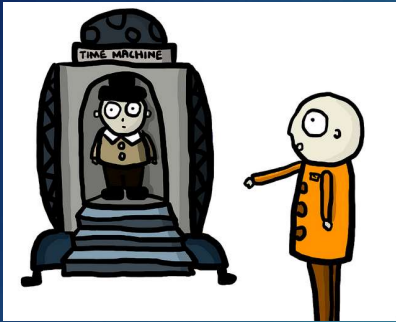
STATEMENTS AND STATEMENT BLOCKS

Let's go through an example.

C# Comments

- ▶ `//` this is a comment
 - ▶ Single line comments
- ▶ `/*` this is a multi line
comment `*/`
 - ▶ Multi-line comments

- Comments can appear on their own line or at the end of the line.
- Single line comments are done using two slashes.
- Multi-line comments can be done using `/*` and `*/`



EXAMPLE

[COMMENTS](#)

Let's go through an example.

Types

▶ Basic Built-In Types

- ▶ bool
- ▶ int
- ▶ decimal
- ▶ string
- ▶ array

- In .Net there are built-in types and user defined types.
- Built-in types are built into the framework.
- All other types are user defined types
- Anyone can create them by creating classes or structs.
- Many are provided by Microsoft in the framework.
- Some basic built in types are bool, int, decimal, string, and array
 - bool – represents a variable type that is true or false
 - int – represents a variable type that is a number with no decimal points.
 - decimal – represents a variable type that is a number with decimal points.
 - array – represents a variable type that has multiple values of the same type.
 - Each value in an array is contained in an index.
 - In C# an array index starts at 0 and increments or goes up by 1 for each additional index

Declaring String Variables

- ▶ Declaring Variables
 - ▶ `string myString;`
 - ▶ `string myString = "test string";`
- ▶ Using Variables
 - ▶ `Console.WriteLine(myString);`

- Variables are declared by specifying the type followed by a name.
- The value can be specified at the same time by setting it equal to a value.
- String variables will be null if they are not set to a value.

Declaring Number Variables

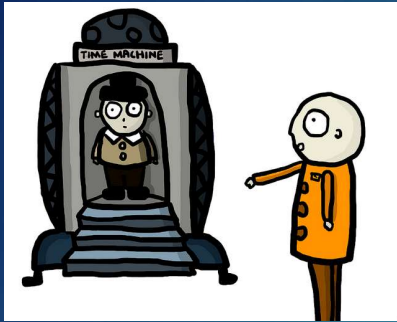
- ▶ Declaring Variables
 - ▶ `int myInt;`
 - ▶ `int myInt = 5;`
 - ▶ `decimal myDecimal = 5.234;`
- ▶ Using Variables
 - ▶ `Console.WriteLine(myDecimal);`

- Numbers will be 0 if they are not given a value.

Declaring Bool Variables

- ▶ Declaring Variables
 - ▶ `bool myBool;`
 - ▶ `bool myBool = true;`
- ▶ Using Variables
 - ▶ `Console.WriteLine(myBool);`

- A bool will be false if not given a value



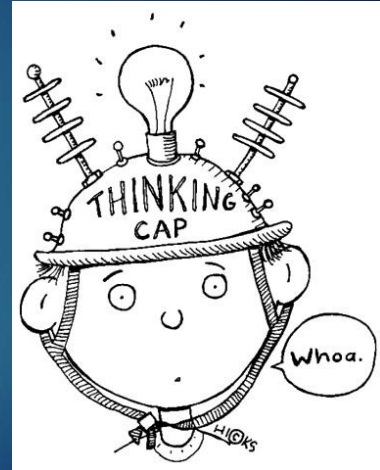
EXAMPLE

DECLARING VARIABLES

Let's go through an example.

ASSESSMENT

CLI, STATEMENTS, BLOCKS, COMMENTS, VARIABLES



- What does CLI stand for?
- How do you create a new project using the .Net CLI?
- Write on the board the command to create a new class library project.
- What is the command to create a new directory?
- What is the command to change a directory?
- What character ends every statement?
- What characters are used to denote a statement block?
- How is a single line comment specified? Multi-line comment?
- Write on the board a declaration of a string variable. Set its value at the same time.
- Write on the board a declaration of an int variable. Set its value at the same time.
- Write on the board a declaration of a bool variable. Set its value at the same time.

Comparison Operators

- ▶ **Do not compare with =**
- ▶ < Less Than
- ▶ > Greater Than
- ▶ <= Less Than or Equal To
- ▶ >= Greater Than or Equal To
- ▶ == Equal To
- ▶ != Not Equal To

- There are many comparison operators that can be used to compare variables
- There is less than, greater than, less than or equal to, greater than or equal to, equal to, and not equal to
- Notice that equal to has 2 equal signs.
- This is because 1 equal sign is an assignment and that's not what we want to do when comparing.

Logical Operators

- ▶ & And
- ▶ | Inclusive Or
- ▶ && Conditional And
- ▶ || Conditional Or

- You can have multiple conditions in one if statement.
- You can do this by using the logical operators
- You can have an and operator which checks if multiple conditions are true
- You can have an or operator which checks if one condition is true or another one is true.
- There are also short circuited and and or
- The difference here is if one condition fails because a value is null it will still evaluate the second condition.

If Statement

▶ Example:

```
bool myVariable = true;  
If (myVariable)  
{  
    console.WriteLine("true");  
}
```

- An if statement will conditionally run logic based upon the result of the condition.
- The condition must always evaluate to true or false or an error will occur.
- The conditional statements are surrounded by curly braces.

If-Else Statement

► Example:

```
bool myVariable = true;
If (myVariable)
{
    console.WriteLine("true");
}
else
{
    console.WriteLine("false");
}
```

- The else statement can be added to any if statement
- This means that if the if condition is false then the statements within the else will be run
- The else statements are also within curly braces

Nested If Statement

▶ Example:

```
bool myVariable = true;
bool myVariable2 = false;
If (myVariable)
{
    if(myVariable2)
    {
        console.WriteLine("true");
    }
}
```

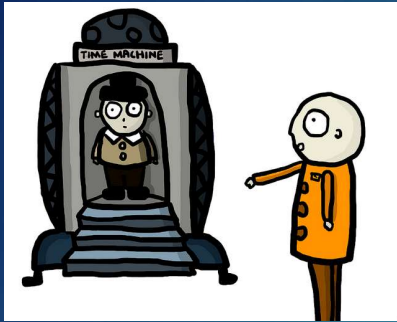
- If statements can be nested

If Multiple Else Statement

► Example:

```
bool myVariable = true;
bool myVariable2 = true;
If (myVariable)
{
    console.WriteLine("true");
}
else if(myVariable2)
{
    console.WriteLine("variable2 true");
}
else
...
```

- In addition to else statements there are else if statements
- With else if statements, there is an additional if condition if the original is false, if that one is true then the logic there will be run



EXAMPLE

IF ELSE STATEMENTS

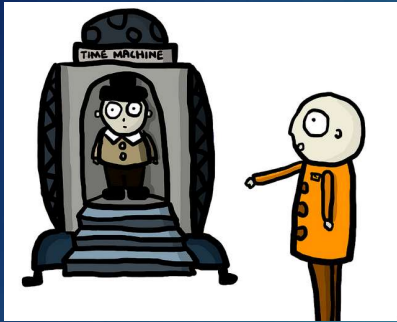
Let's go through some examples

Switch Statement

► Example

```
int myVariable;  
switch(myVariable)  
{  
    case 1:  
        Console.WriteLine("1");  
        break;  
    case 2:  
    case 3:  
        Console.WriteLine("2 or 3");  
        break;  
    default:  
        Console.WriteLine("default");  
        break;  
}
```

- You could chain a bunch of if else statements together, but a more succinct way of writing that would be with a switch statement.
- The switch statement evaluates multiple conditions to find which logic should be run.
- A break statement is needed at the end of the statements for a particular case to stop the logic from going into the next case.
- A default option can be entered if no other cases evaluate to true.



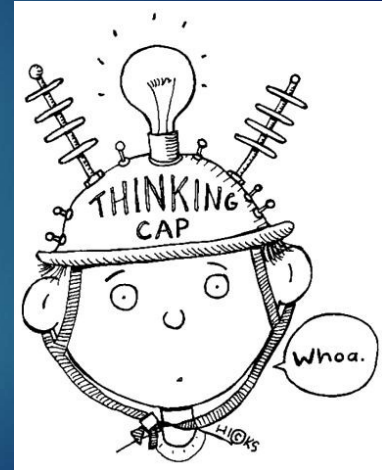
EXAMPLE

SWITCH STATEMENTS

Let's go through some examples

ASSESSMENT

CONDITIONAL STATEMENTS



- What are the logical operators?
- Write an if statement on the board that checks if a bool variable is true.
- Write an if else statement on the board that checks if a bool variable is true.
- Write a nested if statement on the board that checks two different bool variables.
- Write if, else, else if statements to write to the console the text representation of the numbers 1 to 3.
- Write a switch statement to write to the console the text representation of the numbers 1 to 3.

Assignment

- ▶ A status report is needed of all government employees. Statuses are:
 - ▶ 1: Alive, 2: Zombie, 3: Dead, 4: Unknown
- ▶ Given an int variable, write if else statements and console out the persons status.
- ▶ Using the same int variable, modify your code to perform the same operation with a switch statement.



Value and Reference Types

▶ Type System

▶ Value Types

- ▶ Contain data within it's own memory location.
- ▶ int, decimal, bool

▶ Reference Types

- ▶ Contain a pointer to a memory location.
- ▶ Require a new instance of an object.
- ▶ Are null if no instance of an object has been provided.
- ▶ string, array, class

- int, decimal, and bool are what as known as value types
- Value types contain data within their own memory location.
- string and array are reference types.
- Reference types only contain a pointer to data in memory.
- string is special and is declared just like value types.
- All other reference types require the use of the new keyword in order to create a new instance of an object.
- Prior to an instance being provided, a reference type will contain a null value, which means a lack of a value.

Declaring Arrays

- ▶ Declaring Variables
 - ▶ `int[] myArray;`
 - ▶ `myArray = new int [5];`
 - ▶ `myArray = new int[] {0, 1, 2, 3};`
 - ▶ `int[] myArray = new int[] {0, 1, 2, 3};`
 - ▶ `int[] myArray = {0, 1, 2, 3};`
- ▶ Using Variables
 - ▶ `myArray[5] = 6;`
 - ▶ `Console.WriteLine(myArray[5]);`
 - ▶ `myArray.Length`

- Arrays are declared with other data types followed by square brackets [].
- Because array is a reference type, it requires the new keyword to create a new instance of the array.
- Values can be set to the array one by one or all at once.
 - If this is the case then the array needs to be initialized with the number of indexes (values) that are required.
 - If all at once, then values are provided in curly braces {} and are comma separated.
 - The size of the array does not need to be specified if you are directly setting values.
- As a short hand the new keyword can be bypassed and directly set to values using the curly brace {} syntax.
- When using arrays, an index number can be specified inside of square brackets followed by the variable name. This will supply the value at that index in the array.

while Loop

▶ Example

```
int[] myArray = {0, 1, 2, 3};  
int counter = 0;  
  
while (counter < myArray.Length)  
{  
    Console.WriteLine(myArray[counter]);  
    counter++;  
}
```

- While loops will loop as long as the condition is true

do-while Loop

▶ Example

```
int[] myArray = {0, 1, 2, 3};  
int counter = 0;  
  
do  
{  
    Console.WriteLine(myArray[counter]);  
    counter++;  
} while (counter < myArray.Length);
```

- Do while is very similar to a while loop except that the evaluation of the condition is at the end instead of the beginning.

for Loop

► Example

```
int[] myArray = {0, 1, 2, 3};  
  
for(int counter = 0; counter < myArray.Length;  
counter++)  
{  
    Console.WriteLine(myArray[counter]);  
}
```

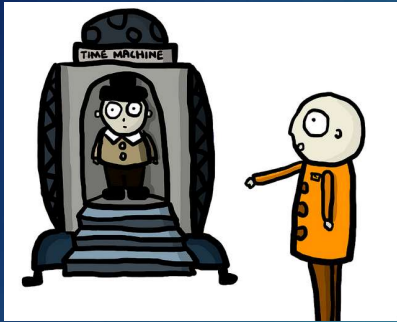
- The for loop takes 3 statements.
- The first statement is to initialize the variable
- The second statement is the condition to check
- The third statement is to update a variable.
- This is a bit more succinct than the while loop since it can do all things normally needed for a while loop in one line.

foreach Loop

► Example

```
int[] myArray = {0, 1, 2, 3};  
  
foreach(int value in myArray)  
{  
    Console.WriteLine(value);  
}
```

- The foreach loop is similar to the for loop
- The difference is that it allows you to iterate through an enumerable variable.
- An enumerable variable is a variable that has a type that implements IEnumerable interface.



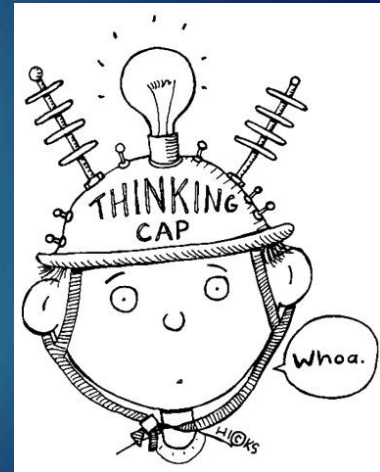
EXAMPLE

LOOPS

Let's go through some examples

ASSESSMENT

LOOPS



- What is the difference between a value and reference type?
- Is int a value or reference type? What about string?
- Write on the board a declaration of an array and initialize it with the numbers 1 to 3.
- Write a while loop that loops through an integer array 1 to 3 and writes each number to the console.
- Write a do while loop that loops through an integer array 1 to 3 and writes each number to the console.
- Write a for loop that loops through an integer array 1 to 3 and writes each number to the console.
- Write a foreach loop that loops through an integer array 1 to 3 and writes each number to the console.

Assignment

- ▶ A status report is needed of all government employees. Statuses are:
 - ▶ 1: Alive, 2: Zombie, 3: Dead, 4: Unknown
- ▶ Given an array of int variable, write loops with if else statements and console out everyone's status.
- ▶ Use all loop types.



Methods

- ▶ Smaller and Manageable
- ▶ Cohesive Actions
- ▶ Reusable
- ▶ Functions Return a Value
 - ▶ Only one value can be returned
- ▶ Voids do not Return a Value
- ▶ Parameters
- ▶ Method Overloads

- It would be difficult to deal with code that just went on endlessly.
- Therefore, code is broken down into methods
- Methods should be small and manageable
- Methods are statements grouped into cohesive actions
- Methods are reusable. You can call them multiple times without re-writing them.
- Functions are methods that return a value
 - Only one value can be returned from a function.
- Void methods do not return a value
- Methods can have parameters, which are values passed to the method to be used inside of the method.
- You can overload a method which means to have two methods with the same name.
 - If you do, it requires parameter types or the number of parameters (the signature) to be different.

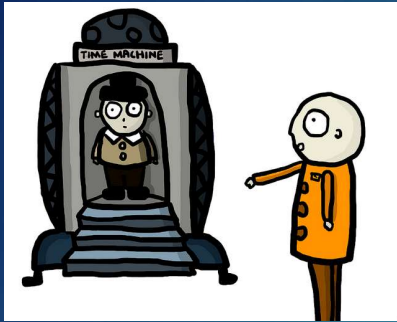
Method Syntax

```
[access modifier] [return type] [name]([type1] [parameter1],  
[type2] [parameter2])  
{  
    Statements...;  
}
```

▶ Example:

```
private int AddNumbers(int num1, int num2)  
{  
    Statements...;  
}
```

- This is the general syntax for a method.
 - The access modifier we will talk about later.
 - The return type is the type that should be returned. If there is no value to be returned then void should be used.
 - Next comes the name of the method.
 - After put in the parameters comma separated within the parenthesis ().
 - The parameters should be listed with the type followed by the parameter name.



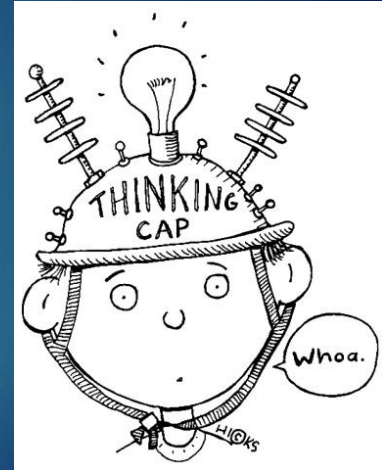
EXAMPLE

METHODS

Let's go through an example.

ASSESSMENT

METHODS



- Write a method definition that does not return anything and takes two integers as parameters.
- Write a method definition that returns a string and takes two integers as parameters.

Assignment

- ▶ A status report is needed of all government employees. Statuses are:
 - ▶ 1: Alive, 2: Zombie, 3: Dead, 4: Unknown
- ▶ Modify your previous program to create a method that handles the condition given a parameter for status that returns the status string.



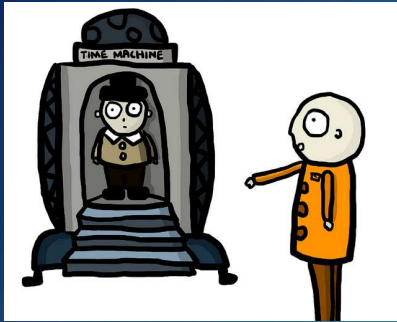
Working with Generic Types

- ▶ Type Safety
- ▶ Re-use
- ▶ Generic Collections –
System.Collections.Generic

- ▶ Example:

```
List<string> strings = new List<string>();  
strings.Add("test");  
List<int> ints = new List<int>();  
ints.Add(3);
```

- Generics allow for type safety while at the same time allowing the same code to work for multiple types.
- Needed in order to have logic be re-usable for different types otherwise, developers had to accept object. This lead to a lot of boxing and unboxing. Generics avoids this and keeps things type safe.
- Generics are used a lot with collections.
- Microsoft provides a number of generic collections within the System.Collections.Generic namespace
- An example is List.
- You can see here a list of string and then a list of integers both following the same code paths.



EXAMPLE

GENERIC

Let's go through an example.

ASSESSMENT

GENERIC



- What is the advantage of using generics?
- Write on the board a declaration of a generic list of integers and instantiate it.

Assignment

- ▶ A status report is needed of all government employees. Statuses are:
 - ▶ 1: Alive, 2: Zombie, 3: Dead, 4: Unknown
- ▶ Modify your previous program to create a generic list of status descriptions.



QUICK REVIEW

C#



- Write on the board the command to create a new class library project.
- What character ends every statement?
- What characters are used to denote a statement block?
- How is a single line comment specified? Multi-line comment?
- Write on the board a declaration of a string variable. Set its value at the same time.
- What are the logical operators?
- Write if, else, else if statements to write to the console the text representation of the numbers 1 to 3.
- Write a for loop that loops through an integer array 1 to 3 and writes each number to the console.
- Write a method definition that returns a string and takes two integers as parameters.
- Write on the board a declaration of a generic list of integers and instantiate it.

Additional Resources

- ▶ Code Katas
 - ▶ <https://www.codewars.com/>
- ▶ DotNet Fiddle
 - ▶ <https://dotnetfiddle.net/>
- ▶ Codeasy.net
 - ▶ <https://codeasy.net/welcome>
- ▶ Microsoft Virtual Academy
 - ▶ <https://mva.microsoft.com/>
- ▶ Microsoft Docs
 - ▶ <https://docs.microsoft.com/en-us/dotnet/csharp/index>

Keep Practicing!

- ▶ Try declaring different types of variables.
- ▶ Try different combinations of if, else statements.
- ▶ Try different combinations and logic for loops.
- ▶ Try creating different methods with different parameters and return types.
- ▶ Try different ways of working with the generic list.