STRINGS

STRING VARIABLES

- A string consists of letters, numbers, and special characters strung together.
- Use "String" or "string" to declare a string variable.

STRING VARIABLES

- Strings are immutable!
- This means that once they are created, they cannot be changed!

CREATING STRINGS

Examples

```
String word ="Hello!";
char[] helloArray = { 'H', 'e', 'l', 'l', 'o'};
String helloWord = new String(helloArray);
```

JOINING STRINGS

- A string may be joined with another string by using the plus symbol (+). However, this will convert any other data type to a string.
- Another way is to use the string.Concat method, which takes two strings as parameters, and returns a new joined string.

COMPARING STRINGS

- You can compare strings in C# using ==, String.Compare, or using the Equals method of the string object
- Equals method can be used to ignore the case when comparing two strings

COMPARING STRINGS

Examples

```
// return true if the firstname is equal to Frank
firstName.Equals("Frank")
// return 0 if the two strings are equal(ignore case)
String.Compare(firstname, "Frank", true)
// equal to an empty string
firstName.Equals("")
// not equal to a string literal
!lastName.Equals("Jones")
// not equal to a null value
firstName != null
```

STRING FUNCTIONS

Examples

- int IndexOf(String str): Returns the index of the first occurrence of a certain substring. If the substring is not found, the function returns -1
- int LastIndexOf(String str): Returns the index of the rightmost occurrence of the certain substring.
- bool EndsWith(String suffix): Checks if the string ends with the a certain suffix.

STRING FUNCTIONS

Examples

- String Replace(char oldChar, char newChar): Returns a copy of the string that has oldChar replaced with newChar.
- String[] Split(separator(s)): Splits the string around matches of given char separator(s), and returns the words as an array of strings.

STRING FUNCTIONS

Examples

- String substring(int beginIndex): Returns a new a substring that starts at a specified index
- String ToUpper(): Returns a string that has all upper case chars.
- String trim(): Omits leading and trailing whitespaces.

STRINGBUILDER

- Strings can leave many unused objects in the memory when you do a lot of operations on them, as a new copy is made after each operation.
- It is better to use StringBuilder when you do a lot of string operations.
- Unlike Strings, objects of type StringBuilder are mutable, so they can be modified.

STRINGBUILDER EXAMPLE

```
StringBuilder strBuff = new StringBuilder("test!");

strBuff.Append("\t Super!");

Console.WriteLine(strBuff);
```

RECAP

WHAT YOU SHOULD KNOW AT THIS POINT:

- What are strings
- How to define and initialize strings
- Joining strings
- Comparing strings
- String functions
- Mutable strings (StringBuilder)