CSC105

Building a Console Application

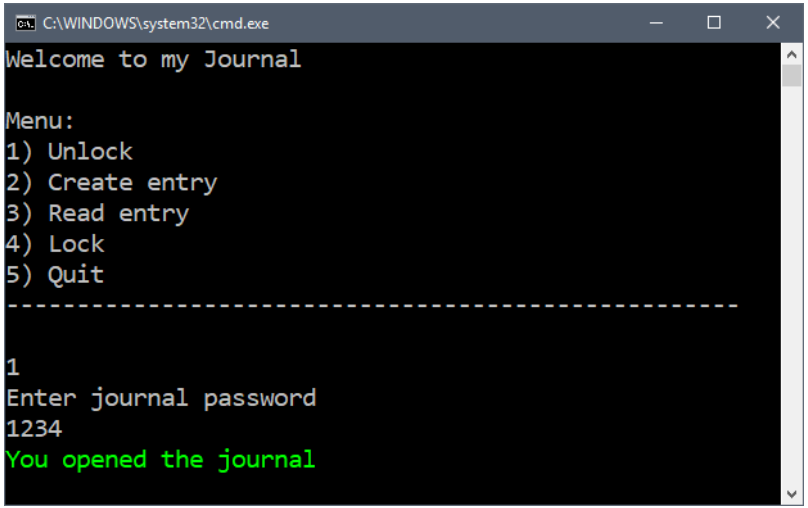
Microsoft

Xamarin University

What is the purpose of this course?

- This course focuses on applying your knowledge from previous courses to build a console application

You will build this application during the course



Objectives

1. Define a custom type using a class
2. Create properties to control access to data
3. Repeat a block of code with a loop
4. Create methods to add behavior to classes

Demonstration

Explore the completed solution

Xamarin University

Define a custom type using a class

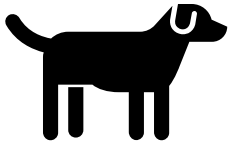
Tasks

1. Define a class
2. Add fields to store data
3. Create an instance of a class

What is a class?

- A *class* defines a custom type that often models the real world

Car



Dog

Message

How to create a class

- A C# class is defined using the keyword **`class`**

Starts with **`class`**

```
class Car
{
   ...
}
```

You choose a name

The body is delimited with braces

Data and behavior

- Classes contain fields to store data and methods to implement behavior

What the class *has* (fields)

What the class *does* (methods)

```
class Car
{
    public int Year;
    public string Make;
    public string Model;

    public void TurnOnRadio() {...}
    public void Accelerate(int delta) {...}
    public void Lock() {...}
}
```

How to create an instance

- To create an instance of a class you use the **new** keyword

Each car has its own memory

```
public class Program
{
    public static void Main()
    {
        Car bobsCar = new Car();
        Car samsCar = new Car();
        ...
    }
}
```

Exercise

Create the Journal class

Xamarin University

Create properties to control access to data

Tasks

1. Create a property
2. Define an auto-implemented property

What is a property?

- A *property* is a class member that provides read and/or write access to a piece of data stored in an object

```
class Car
{
   private int miles;
   public int Miles
   {
      get { return miles; }
      set {
            if (value >= 0)
               miles = value;
         }
   }
}
```

Field to store data

Retrieve value of the field

Load new value of field (**value** represents the new value being assigned)

How to use a property

- Use the assignment operator to load a new value into a property; use the property name to read the existing value

```
public static void Main()
{
    Car samsCar = new Car();
    samsCar.Miles = 10;
}
```

Assign new value to property

```
public static void Main()
{
    ...
    Console.WriteLine(samsCar.Miles);
}
```

Read existing value of property

Simple properties

- Some properties do not require logic inside the get and set accessors

```
class Car
{
    private string model;
    public string Model
    {
        get { return model; }
        set { model = value; }
    }
}
```

No validation code needed, any string value is legal

What is an auto-implemented property?

- An *auto-implemented property* is a property where the backing field is created automatically and the get/set accessors do not contain any logic

No backing field needed

```
class Car
{
    public string Model { get; set; }
}
```

Read/write the value with no additional logic

Exercise

Creating the JournalEntry class and property

Repeat a block of code with a loop

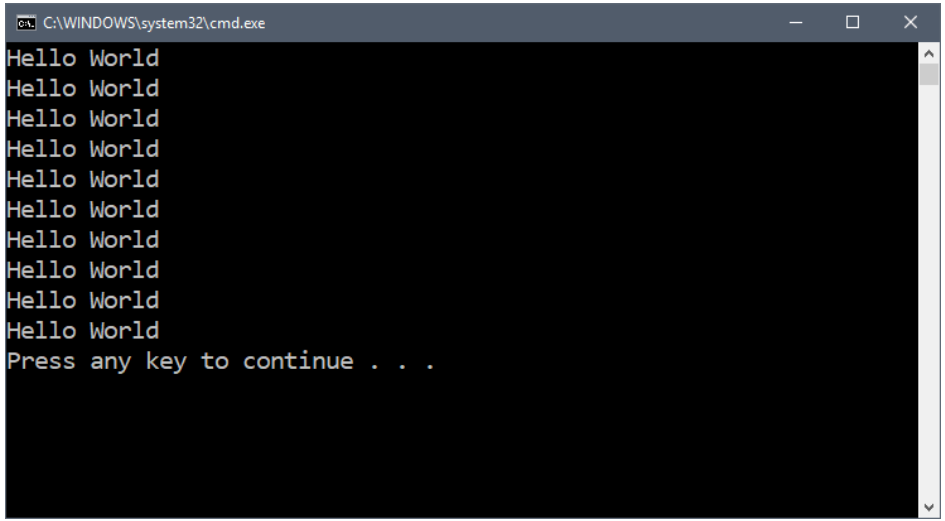
Xamarin University

Tasks

1. Create a do-while loop
2. Build a console menu using a loop

What is a loop?

- A *loop* is a statement that repeats a block of code for a specified number of times or until some condition is met

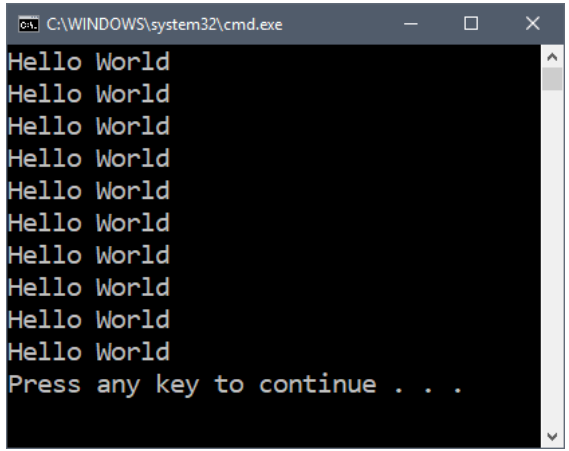


What is a do-while loop?

- A *do-while* loop is a loop that guarantees at least one execution and continues as long as the condition is true

```
int x = 0;
do
{
    Console.WriteLine("Hello World");
    x++;
} while (x < 10);
```

Executes while true



Console menu loop

- *do-while* loops are appropriate for creating console menus because they always run at least once

Present a list of options →

```
int choice;
do
{
   Console.WriteLine("\nMenu: ");
   Console.WriteLine("1) View Cars");
   Console.WriteLine("2) Add Car");
   Console.WriteLine("3) Remove Car");
   Console.WriteLine("4) Quit");
   choice = int.Parse(Console.ReadLine());
   ...
} while (choice != 4);
```

Exercise

Create a console menu

Xamarin University

Create methods to add behavior to classes

Xamarin University

Tasks

1. Define a method
2. Invoke a method

What is a method?

- A *method* is a code block containing C# statements that implement an operation related to the class

All of these operations are things you would do to a car →

```
class Car
{
    public void TurnOnRadio() {...}
    public void Accelerate(int delta) {...}
    public void Lock() {...}
}
```

How to create a method

- A method is defined with a name, return type, and parameter list

No data returned

Body of method

```
class Car
{
    public double Speed { get; set; }

    public void Accelerate(int delta)
    {
        Speed += delta;
    }
}
```

Single parameter

How to call a method

- A method is invoked using the method's name followed by parentheses

```
static void Main()
{
    Car samsCar = new Car();
    samsCar.Accelerate(5);
}
```

Parameter is passed to the method

Invoke **Accelerate** on the **samsCar** object

Exercise

Define and invoke methods

Summary

1. Define a custom type using a class
2. Create properties to control access to data
3. Repeat a block of code with a loop
4. Create methods to add behavior to classes

Thank You!

Microsoft