Intro to Classes and Objects

Principles of Computer Programming I

Spring/Fall 20XX

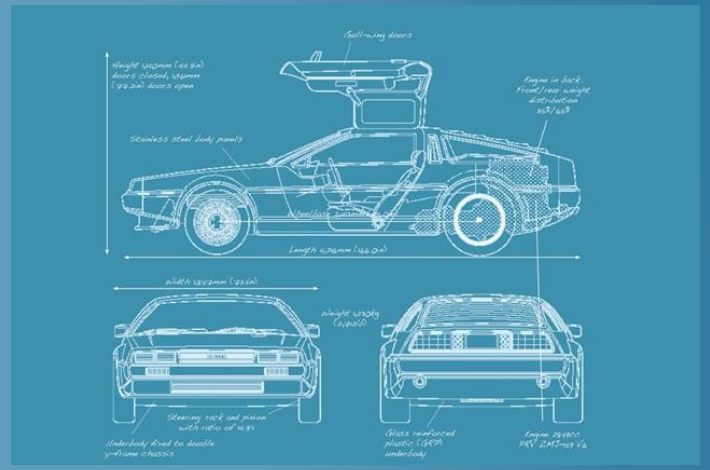
AUGUSTA UNIVERSITY

Outline

- Class and Object Basics
- Writing Our First Class
- Using a Class

Reminder: Classes and Objects

- Class = blueprint, template for object
  - Code that describes an object
- Object = single instance of class
  - Running code, with specific values/state
- Instantiate = create an object from a class
- Attribute = data stored in object
- Method = function that uses object's data
  - Defined in class, but executed on specific object (usually)

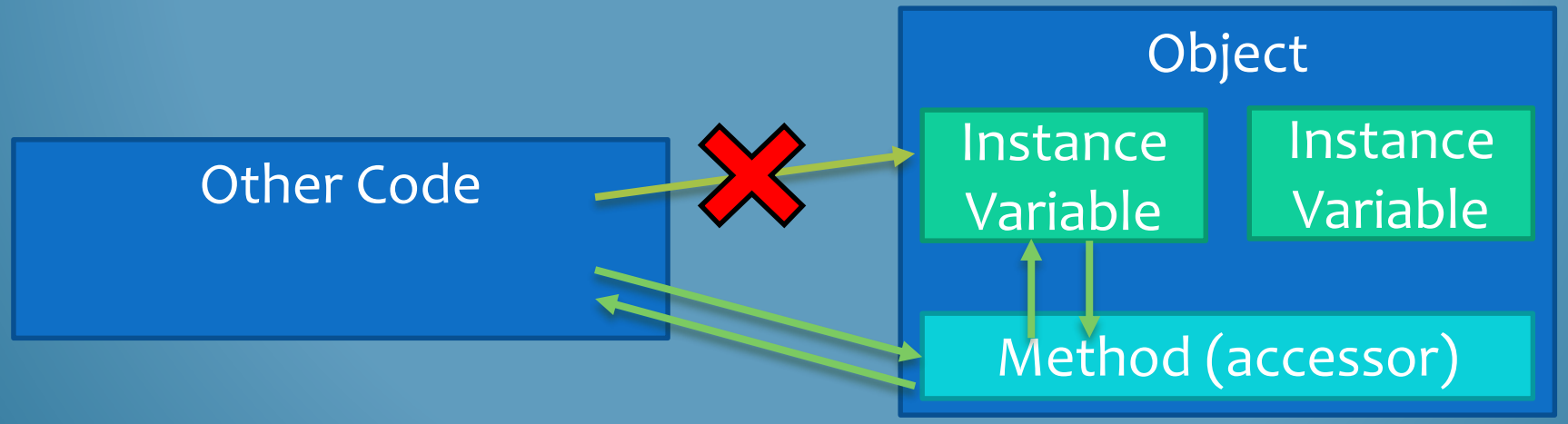


Class

Object

Encapsulation

- Attribute data is stored in **instance variables**
- Instance variables are "hidden" inside the object
- Other code cannot access instance variables directly; only the object's methods can access them
- **Accessor:** method that allows other code to access attributes

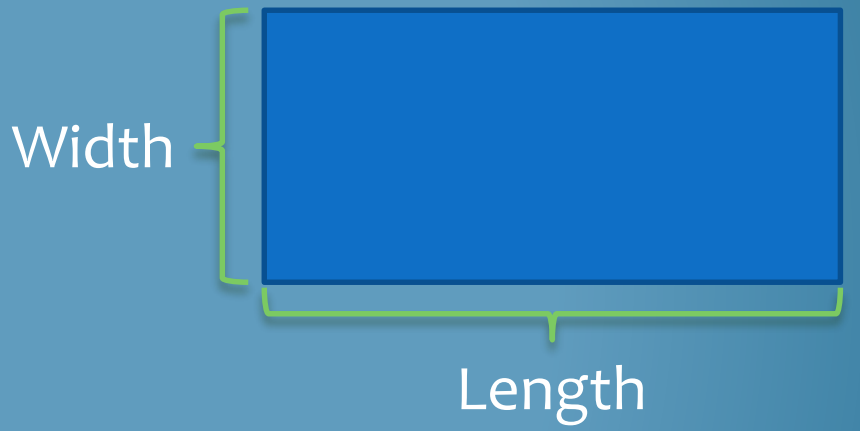


Outline

- Class and Object Basics
- **Writing Our First Class**
- Using a Class
- Object and Method Details
  - Instance variable modification
  - Return types and return values

The Rectangle Class

- Attributes:
  - Length
  - Width
- Methods:
  - Get and set length (accessors)
  - Get and set width (accessors)
  - Compute area



Code: Attributes and Setter

```
class Rectangle
{
  private int length;
  private int width;

  public void SetLength(int lengthParameter)
  {
    length = lengthParameter;
  }
  //continued…
```

Declare an instance variable

Access modifier: ensures that `length` cannot be accessed outside of `Rectangle`

Declare a method named `SetLength`

Parameter type: `int`

Access modifier: `SetLength` can be called by any code

Parameter name: `lengthParameter`

Method body: Assign parameter's value to instance variable

Code: More Accessors

```
public int GetLength()
{
  return length;
}
public void SetWidth(int widthParameter)
{
  width = widthParameter;
}
public int GetWidth()
{
  return width;
}
```

Declare a method named `GetLength`

Return type: when called, this method will return an `int` value to the caller

`return` keyword: directs method to return a value

Return type `void`: This method does not return a value

Assign parameter's value to instance variable

Just like `GetLength()`, but returns the value of `width` instead

Code: ComputeArea

Note: Not an accessor. Does not get or set an attribute

Return type: `int`

```
public int ComputeArea()
{
    return length * width;
}
}
```

`int * int = int`

`return` uses result of expression as value to return

End of class declaration

Expression computed first, using current values of `length` and `width`

Outline

- Class and Object Basics
- Writing Our First Class
- **Using a Class**

A Program That Uses Rectangle

```
class Program
{
  static void Main(string[] args)
  {
    Rectangle myRectangle = new Rectangle();
    myRectangle.SetLength(12);
    myRectangle.SetWidth(3);
    int area = myRectangle.ComputeArea();
    Console.WriteLine("Your rectangle's length is" +
      $"{myRectangle.GetLength()}, and its width is" +
      $"{myRectangle.GetWidth()}, so its area is {area}.");
  }
}
```

Declare a variable with type Rectangle

Instantiate a Rectangle object

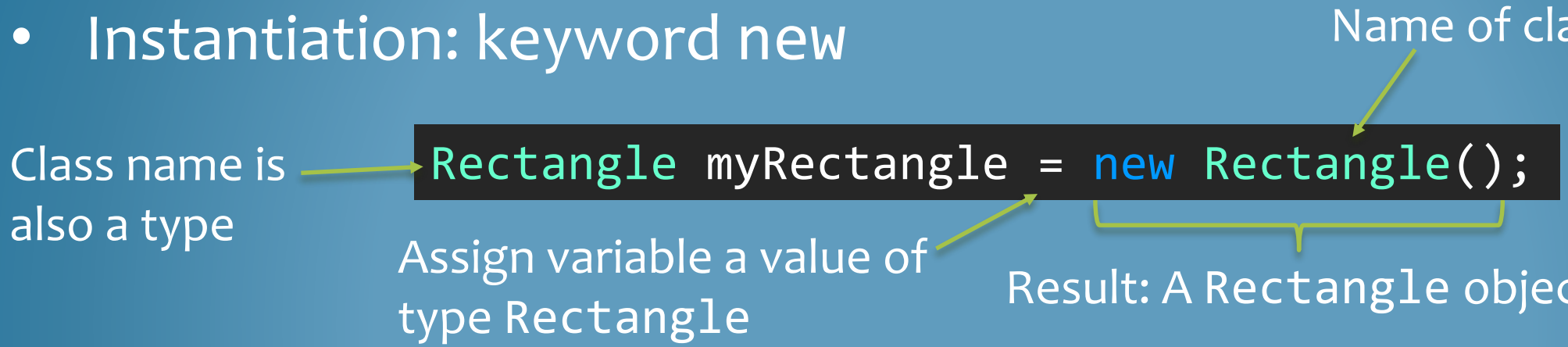
Call the SetLength method with argument 12

Call the ComputeArea method, store its return value in a variable

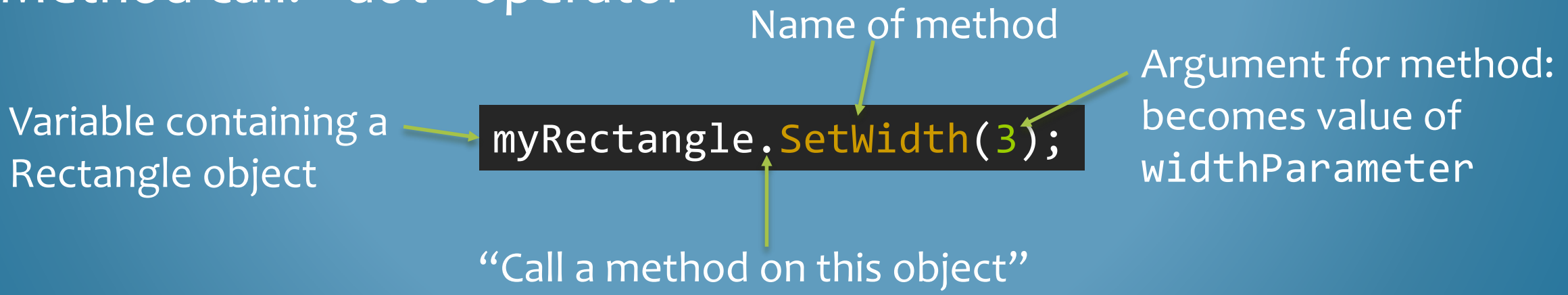
Call the GetLength and GetWidth methods, put their return values in a string

Syntax Details

- Instantiation: keyword new



- Method call: "dot" operator



Flow of Control

In Program.cs:

```
static void Main(string[] args)
{
  Rectangle myRectangle = new Rectangle();
  myRectangle.SetLength(12);
…
```

Value: 12

In Rectangle.cs:

```
public void SetLength(int lengthParameter)
{
  length = lengthParameter;
}
```

The length variable stored in myRectangle

Summary

- Class and Object Basics
- Writing Our First Class
- Using a Class