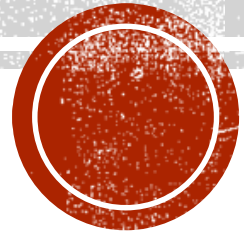
CHAPTER 13: OBJECT-ORIENTED PROGRAMMING

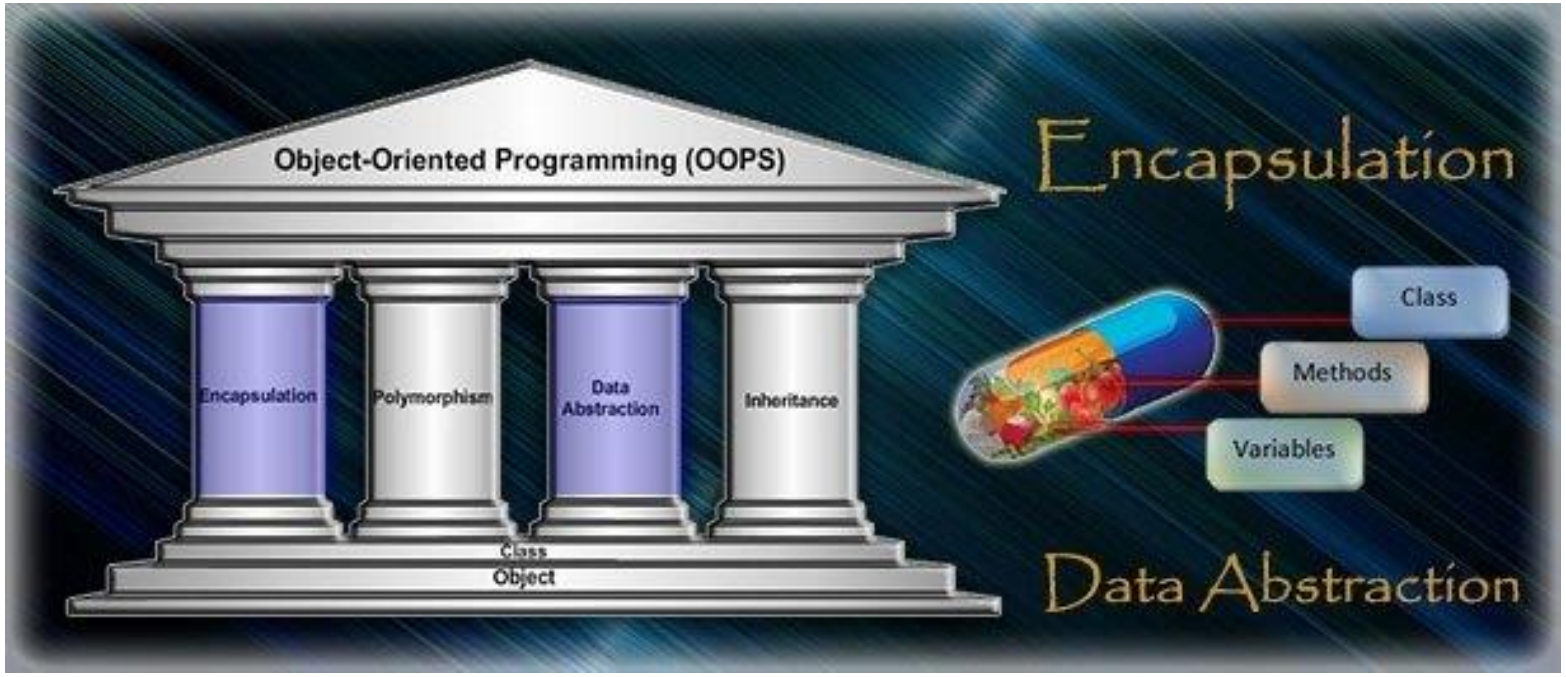
Fall 2019 – CSC 180 – Introduction to Programming

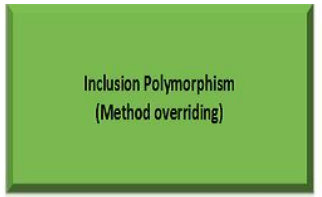


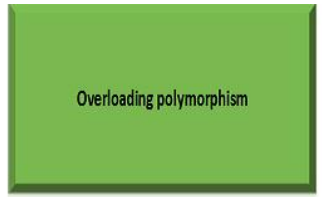
OBJECT-ORIENTED PROGRAMMING (OOP)

- The four pillars of OOP:
  - Encapsulation
  - Data abstraction
  - Polymorphism
  - Inheritance

Source:
http://themoderndeveloper.com/the-modern-developer/back-to-basics-three-or-four-oop-pillars/
See also:
https://www.c-sharpcorner.com/UploadFile/e6a07d/pillars-of-oop/

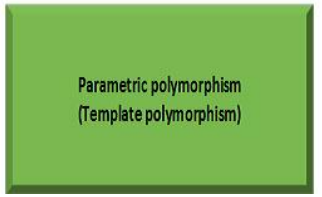






- How to implement polymorphism in C#
  - Overloading
  - Overriding
  - Templates

Source:
https://www.infoworld.com/article/3024718/how-to-implement-polymorphism-in-c.html



STATIC VS INSTANCE

- There are **two types of methods** in object-orientated languages:
  - the static method (sometimes called **class** methods)
  - the instance method (sometimes called **object** methods)

- **Static** methods are identified by the keyword **static** in the first line.
  - Any method that does not have the keyword static is an **instance** method.
    - **Static** methods can only access static fields
    - **Instance** methods can access both **static** and **non-static** fields.
      - The current object (this) is used for the non-static fields.

- Whenever you invoke a method "on" an object, it's an **instance** method.
  - For example, **ToLower** and the other methods we invoked on string objects are all **instance** methods.
  - When you invoke a method on an object, we are invoking the method on a particular instance of the class of objects. The specific instance becomes the current object, also known as this.
    - `this' is not valid in a **static** property, **static** method, or **static** field initializer.

CREATING STATIC CLASSES AND MEMBERS

- **Static** members means they are bound to the class, not the instance
  - One can add **static members** to **non-static classes**

- Use the **static** keyword to create a static class/static member
  - public **static** class Conversions
    {
      // Static members go here…. For example: **public static double FtoC(double f);**
    }

- Call members directly on the class name
  - int tempInC = Conversions.FtoC(99);
  - Console.WriteLine("Hello World!")

- Example 4: create a **Student** class that use a static variable to assign a unique ID to every student created.

- Example 5: create a static class called **Conversions** and add a few conversions in it.

THE TOSTRING METHOD

- Every object type has a method called ToString that returns a string representation of the object.
  - When you print an object using Write or WriteLine, the object's **ToString** method is invoked.
  - The default version of **ToString** returns a string that contains the type of the object.

- When you define a new object type, you can override the default behavior by providing a new method with the behavior you want.
  - The **override** tells the compiler that you know you are overriding the default method. The return type is string, naturally, and it takes no parameters.

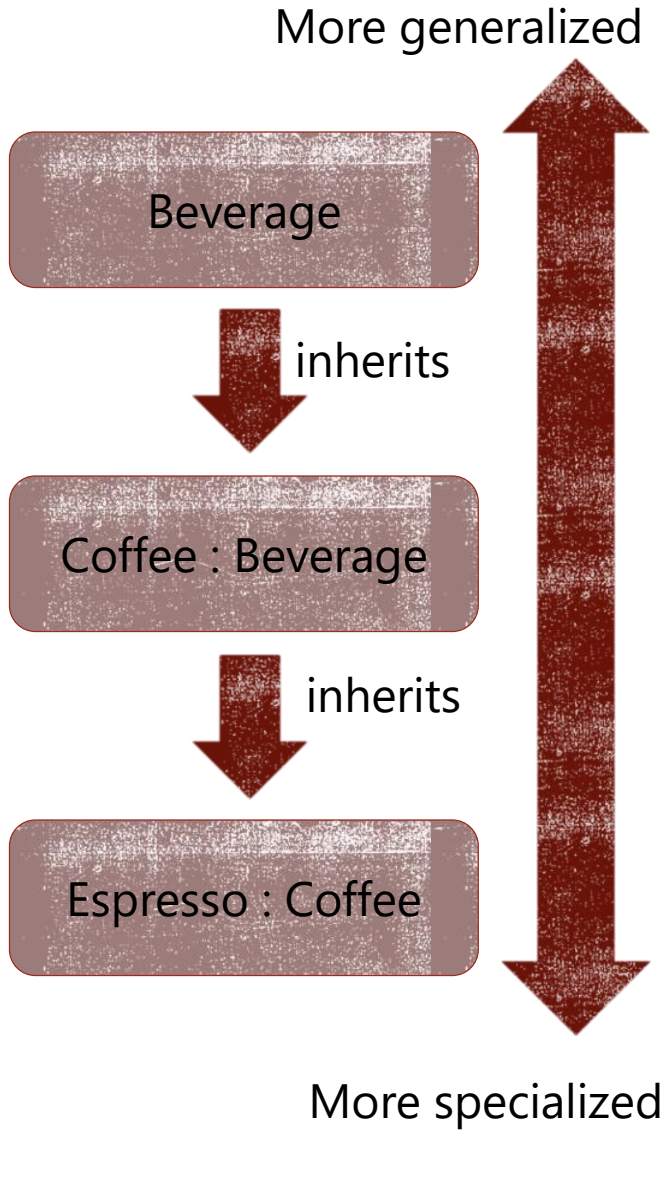
- **Example 1**: Override the **ToString** method in the **Student** class.

WHAT IS INHERITANCE?

- Inheritance enables you to create new classes (derived classes / child classes)

  by inheriting characteristics and behaviors from existing classes (base classes / parent classes)

- The **derived class** is a more specialized instance of the **base class**.
  - The derived class inherits all the members of the base class, including constructors, methods, properties, fields, and events.
  - Within your derived class, you can add new members to extend the functionality of the base type.



INHERITING FROM A BASE CLASS

- To inherit from a base class, add the name of the base class to the class declaration

```
public class Coffee : Beverage
```

- A class can only inherit from one base class. But it can implement one or more interfaces in addition to deriving from a base type.

```
class BaseClass
{
    public void Method1()
    {
        Console.WriteLine("Base - Method1");
    }
}

class DerivedClass : BaseClass
{
    public void Method2()
    {
        Console.WriteLine("Derived - Method2");
    }
}
```

```
static void Main(string[] args)
{
    BaseClass bc = new BaseClass();
    DerivedClass dc = new DerivedClass();
    BaseClass bcdc = new DerivedClass();

    bc.Method1();    //outputs: Base - Method1
    dc.Method1();    //outputs: Base - Method1
    dc.Method2();    //outputs: Derived - Method2
    //bcdc.Method2();  does not compile
}
```

ACCESS MODIFIERS (PUBLIC, PRIVATE, PROTECTED)

- Declaring a definition (class, method or field) private means that only methods in the same class are allowed to access the definition.
- Declaring a definition public, as we have done up until now, means that any method in any other class can use the definition.
- Protected means that only methods, in the same class, or children of the class, can use it.
  - The method is not available to classes outside of the family.
- All classes extend Object, including all of the classes we have written and all of the library classes, like Rectangle. Any class that does not explicitly name a parent inherits from **Object** by default.
  - The "family tree" of classes is called the class hierarchy.

EXAMPLE

- **Example 2**: create a base class of **User**, and two derived classes: **Student** and **Faculty**
  - Show examples of inheritance, access modifiers, overriding, this

- Use the **virtual** keyword (in the base class) to create members that you can override in derived classes
  - You can only override a base class member if the member is marked as **virtual** in the base class.
- To override virtual base class members, use the **override** keyword

```
public override int GetServingTemperature()
```

13.10 Glossary

instance method: A method that is invoked on an object, and that operates on that object. Instance methods do not have the keyword `static`.

static method: A method with the keyword `static`. Static methods are not invoked on objects and they do not have a current object (`this`).

current object: The object on which an instance method is invoked. Inside the method, the current object is referred to by `this`.

implicit: Anything that is left unsaid or implied. Within an instance method, you can refer to the fields implicitly (i.e., without naming the object).

explicit: Anything that is spelled out completely. Within a static method, all references to the non-static fields have to be explicit.

public definition: A method or field that can be accessed by a method in any class.

private definition: A definition (method or field) that can only be accessed by methods in the current class. No other classes (including sub-classes) can access the definition.

protected definition: A definition that can only be accessed by methods in the current class and in sub-classes, but not by methods in any other class.

HOMEWORK FOR CHAPTER 13

- Requirements: see moodle for details
- Deadline: see moodle

- **Reminder: If your code does not compile, crashes at start, or contains no meaningful comments, it will automatically be graded with 0!**