MODULE 1: INTRODUCTION TO PROGRAMMING

# Introduction to Classes and Encapsulation

## Yesterday

- What is a dictionary?
- How do we access elements in a dictionary?
- When should we use a dictionary?
- What is a HashSet?
- When should we use a HashSet?

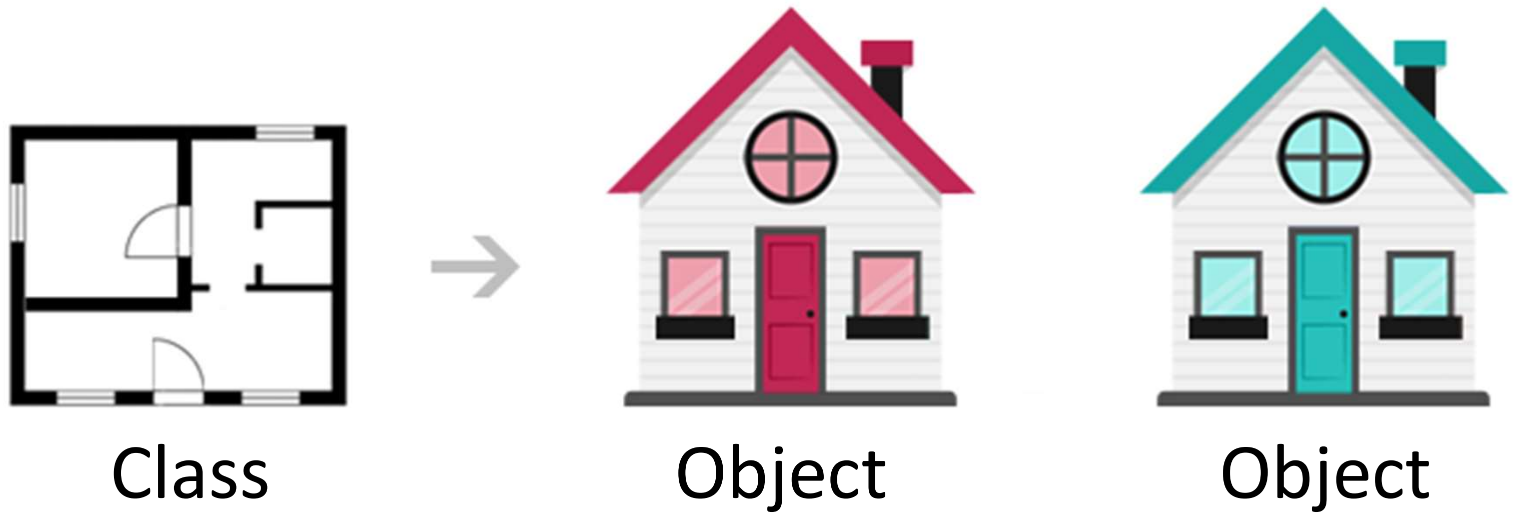
Today's information
← You
ShutterStock.com

## Object Oriented Programming

- A group of objects working together to get the job done

## Classes

- A **class** is a grouping of variables and methods in a source code file that from which we can generate objects.



## Three-ish Fundamental Principles of OOP

- **Encapsulation** - the idea of bundling data and methods that work on that data within one unit and of hiding values or state of data within a class, limiting the points of access.
- **Polymorphism** - the ability for our code to take on different forms.
- **Inheritance** - the practice of creating a hierarchy for classes in which descendants obtain the attributes and behaviors from other classes.
- **Abstraction** - is to handle complexity by hiding unnecessary details from the user.

## Benefits of OOP

- A natural way of expressing real-world objects in code
- Modular and reliable, allowing changes to be made in one part of the code without affecting another
- Discrete units of reusable code
- Units of code can communicate with each other by sending and receiving messages and processing data

## More Detail on Classes

- **Class** is a blueprint or model that defines state with fields (aka variables) and behavior with methods.
- **Instance** of a class that follow the blueprint but may have different property values.

## Class Naming

- Use *nouns or noun phrases*, not verbs
- Try to use the *singular form* of a class name
- Class name should match the file name
- Follow Pascal Casing

## LET'S CODE!

## What did you learn, Dorothy?

- **Fully Qualified Name** is an unambiguous name referencing a specific class type. It includes the namespace (or package) and the class name.
  - System.Collections.Generic.List<int> vs. List<int>
- **Private Modifier** – means accessible only from within class
- **Public Modifier** – means accessible by anything with access to that class
- **Getters and Setter** – how to store and retrieve values
- **Functions/Methods** – the actions of an object. They can have multiple inputs but **only return one value**.

## What did you learn, Dorothy?

- **Derived Property** -- is a getter that, instead of returning a member variable, returns a calculation taken from member variables.
- **Methods**
  - make the code base manageable with smaller chunks
  - reduces code into small units of work, making debugging simpler
  - introduces code reuse
- **Method Signature** – the definition of the method.
  - All methods have a name (that is usually a verb)
  - All methods have a return type (or output)
  - Methods can be parameterless or can include parameters (or inputs).

## What did you learn, Dorothy?

- **Overloaded Methods** -- The ability to create multiple functions with the same name and return type. The parameter signature must be different.
- **Constructors** – a special type of method that runs every time a new object is instantiated.
  - Help when member variables are required
  - Help when certain steps need to occur during initializing
  - Name must match the exact spelling and casing of the class.
  - No return type defined.

## Thirsty for more?

## Three-ish Fundamental Principles of OOP

- **Encapsulation** - the idea of bundling data and methods that work on that data within one unit and of hiding values or state of data within a class, limiting the points of access.
- **Polymorphism** - the ability for our code to take on different forms.
- **Inheritance** - the practice of creating a hierarchy for classes in which descendants obtain the attributes and behaviors from other classes.
- **Abstraction** - is to handle complexity by hiding unnecessary details from the user.

## Encapsulation

- Is the packaging of data and functions into a single component hiding the implementation details of a class to prevent other parties from setting the data to an invalid or inconsistent state and also reduce coupling.

## Goal of Encapsulation

- Makes code extendable
- Makes code maintainable
- Promotes "loose coupling"

# LET'S CODE!

ELEVATE YOURSELF

# What did you learn, Dorothy?

- Access Modifiers:
  - Public for all
  - Private for just that object
- Readonly
  - set can be private: read only
  - set can be left out: settable only through constructor
- Static
  - Makes that thing (method or property) a member of the **class** not the **object**.

## WHAT QUESTIONS DO YOU HAVE?

Reading for tonight:

**Encapsulation**