Austin Horner
Mr Jones
Computer Science
2023-9-16

Lesson 3 Notes: Procedural and Object-Oriented

Week 2: 9/11-9/15

Completing Unit 1 and Starting Unit 2: The Big Programming Picture

Learning targets

- I will understand the **concept of a program**.
- I will explain how information is stored in a **computer's memory**.
- I will describe the process by which programs are **translated to machine** language.
- I will comprehend the purpose of programming languages and give some **examples**.
- I will distinguish between **procedural and object-oriented programming**.

Paradigm shift

something happens that completely blows your mind and **gets you to look at something from a completely different perspective**. programming paradigms.

procedural programming.

tells the machine what to do in order, step by step.

subprograms

take care of **specific tasks**. These subprograms do their work and return to the main program, keeping things tidy and making it easier for **multiple programmers to work** on a program at the same time.

object-oriented programming (OOP)

came into existence in the 1950s and 60s at Massachusetts Institute of Technology.

1. Objects and methods

object

collection of data about an item, such as a ball, or the beach

properties

data about an object is called its properties.
ocean is blue and liquid.

methods

subtasks we mentioned regarding procedural programming.

functions

sequence of statements, or the functions, “belong” to the object.
ocean probably has a “tide” method, that can make the changes to be high or low tide, depending on the time of day.

2. Classes & Instances

class

-is a template that we can use for creating objects.

instance

-object is an instance of the class; it has been “instantiated.”
-Pacific Ocean object, the Atlantic Ocean object, and the Indian Ocean object.
-in the ocean class, but each with their own slightly different properties and methods
-Changes made to one object will not affect other objects

3. Inheritance

-class or object inherits all the properties and behaviors of another class or object.
-Inheritance is an amazing tool for developers and perhaps one of the reasons OOP is so popular!
-Programmers can build new classes or objects that are built upon previous work, which lets them reuse existing code, saving time and money.

4: Overriding

-overriding is the opposite of inheritance.
-replace a specific method of a class with your object’s own unique version.
-ocean class defines a tide method that causes the shoreline to recede in the day and expand at night, our object could do the exact opposite.

5: Encapsulation

-classes acting as a template from which we create objects.
- Another way, class is as a bundle full of all your favorite properties and methods.
-Once the bundle is all tied up, other people can’t see what’s inside, and they can only reach the items that you place at the top of your bundle, near the opening.
-encapsulation is the ability to hide information within a class and only make certain information accessible to the outside world.
-You **control how much access** other classes have to your class.
-Encapsulation results in simpler interactions between objects, allowing classes to interact in a noncomplicated way.
-encourages writing code with advantage: it’s reusable and easy to fix!


Polymorphism

Instead of separate code drawSquare() and drawCircle()
-In OOP, you could write one piece of code called draw()
-apply that code to a different object
-square.draw() , circle.draw() or triangle.draw() .
**-The draw method is polymorphic, it can take many different forms**.
-Polymorphism allows you to write simpler programs and it's a major strength of OOP.

No OOP in this class

Despite this discussion of OOP, we're not actually going to be doing any object-oriented programming in this course. You'll learn that down the line, of course, but for now we'll pick up the basics using that ever- lovable and straightforward paradigm that got computer programming off the ground—procedural programming.