PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

- Let's look at the properties of OOP.
- There are different interpretations of OOP but we shall examine the class based approach.

Defining Object-Oriented Design

ICHEC
Irish Centre for High-End Computing

- OO design is a vast concept, and various authors and languages each have a different idea of it.
- We will focus on *Class-based* OO design, which is a popular interpretation (C++, Java...)
- The OO approach is data-centered: it focuses on defining what your data *is* and what your data is allowed to *do*.

What is an Object?

- An Object is a combination of a **state**, a **behaviour** and an **identity**.
  - **Attributes** are data fields of various types, whose values define an object's **state**.
  - **Methods** are a number of things that the object can be told to do. These define an object's **behaviour**.
  - Finally an object's **identity** is what uniquely distinguishes an object from a similar one.
- A **Class** is a predefined set of attributes and methods. An object is a particular **instance** of a class.

What is an Object: example

- Example:
  - The *Car* class defines four attributes: make/model/colour/speed, and two methods: accelerate/decelerate.
  - A particular instance of that class could have ('citroen','C4','red',0) as its attribute values.
  - Another instance of that class could perfectly have the same attribute values, yet be a distinct *Object*, just like you could have two integer variables with the same value.
- Note that in addition to the methods described above, a class defines two additional methods:
  - a *constructor* whose function it is to create an instance (i.e. an object) of the class,
  - a *destructor* whose function it is to cleanly delete the object.

Encapsulation

- A key concept of OO design, *encapsulation* is the idea that the user isn't given full access to an object. Only a predefined subset of attributes and methods are made available.
- This allows better control of the state of an object and its transformations.
- Example: in the *Car* class, the *speed* attribute isn't accessible. The *accelerate/decelerate* methods are, though, and allow to modify it in controlled fashion.

Inheritance

- Inheritance is a kind of relationship between classes. It defines an *is-a* relationship. A child class inherits all (or sometimes part) of the parent class attributes and methods, and is allowed to add to these, or redefine particular methods.
  - Less code to write.
  - Less code to read for the next programmer: improved maintainability.
  - Code is easily extensible.
- Inheritance is *transitive*. A class inherits from its parent class, its 'grandparent' class, etc. In OO terms, we talk of a *subclass* inheriting from a *superclass*.
- Example: Person (name/contactdetails) $\rightarrow$ Employee (same as Person + salary) $\rightarrow$ Executive (same as Employee + title + bonus)

Polymorphism

- In OOP, *polymorphism* mainly describes the ability of identically-named methods to be defined in multiple ways.
- When such a method is called, its context will determine which definition is used.
- Across different classes, it allows a single common name to be used for the same kind of operation.
  - EmployeeList and CustomerList can both have a 'pop()' method. The class of the object calling it will determine which one is used, therefore whether the item is removed from the front or the back of the list.
- Within the same class, it allows a method to be defined in a way as to accept various numbers or types of arguments.
  - An 'Image' class could provide a 'scale()' method accepting one or two numeric arguments: one would scale uniformly on the x and y axis, while the two argument form would scale each axis individually.

The UML Class Diagram

- The Unified Modeling Language is a standardized, general-purpose modeling language.
- It is widely used to graphically represent OO Class models.
- The figure below shows the basics of such a diagram: