Introduction to Object-Oriented Programming

KSU

College of Sciences

Department Statistics & OR

Preview

Why objects?

Object-oriented programming (OOP)

- classes
- methods
- objects
- representation invariant

Computers just manipulate 0's and 1's
But binary is hard (for humans) to work with

Towards a higher level of abstraction

?

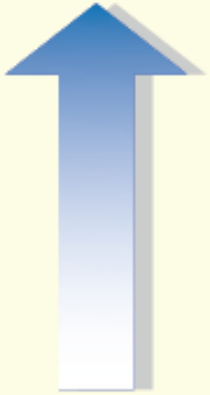
declarative languages (Haskell, ML, Prolog...)

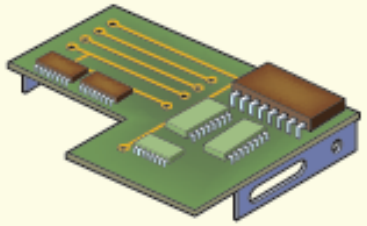
OO languages (C++, Java, Python...)

procedural languages (C, Fortran, COBOL...)

assembly languages

binary code





There are always trade-offs

High-level languages

- simpler to write & understand
- more library support, data structures

Low-level languages

- closer to hardware
- more efficient (but also dangerous!)

What about C++?

What is Object Oriented Programming?

An object is like a black box.

The internal details are hidden.

- Identifying *objects* and assigning *responsibilities* to these objects.
- Objects communicate to other objects by sending *messages*.
- Messages are received by the *methods* of an object

What are objects?

Objects model elements of the problem context

Each object has:

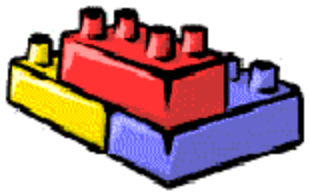
characteristics

responsibilities (or required behaviors)

- Tangible Things as a car, printer, ...
- Roles as employee, boss, ...
- Incidents as flight, overflow, ...
- Interactions as contract, sale, ...
- Specifications as colour, shape, ...

Why do we care about objects?

- Modularity - large software projects can be split up in smaller pieces.
- Reuseability - Programs can be assembled from pre-written software components.
- Extensibility - New software components can be written or developed from existing ones.



Basics in OOP

Classes

A class is like a cookie cutter; it defines the shape of objects

Objects are like cookies; they are instances of the class

Basic Terminology

- *Abstraction* is the representation of the essential features of an object. These are 'encapsulated' into an *abstract data type.*
- *Encapsulation* is the practice of including in an object everything it needs hidden from other objects. The internal state is usually not accessible by other objects.

Basic Terminology: Inheritance

- Inheritance means that one class inherits the characteristics of another class.
  This is also called a “is a” relationship:

A car *is a* vehicle

A dog *is an* animal

A teacher *is a* person

Basic Terminology: Polymorphism

- Polymorphism means “having many forms”. It allows different objects to respond to the same message in different ways, the response specific to the type of the object.

E.g. the message *displayDetails()* of the Person class should give different results when send to a Student object (e.g. the enrolment number).

Basic Terminology: Aggregation

- Aggregation describes a "has *a*" relationship. One object is a part of another object.

A car has wheels.

- We distinguish between *composite* aggregation (the composite "owns" the part) and *shared* aggregation (the part is shared by more then one composite).

Basic Terminology: Behaviour and Messages

- The most important aspect of an object is its *behaviour* (the things it can do). A behaviour is initiated by sending a *message* to the object (usually by calling a method).

The two steps of Object Oriented Programming

- *Making Classes:* Creating, extending or reusing abstract data types.

- *Making Objects interact:* Creating objects from abstract data types and defining their relationships.