Principles of Object Oriented Programming

Mar. 28, 2017

Principles of Object Oriented Programming

- What is OOP?
- Why is it important?

Object-Oriented Programming

- Understanding OOP is fundamental to writing good Java applications
  - Improves design of your code

Object-Oriented Programming

- There are several concepts underlying OOP:
  - Abstract Types (Classes)
  - Encapsulation (or Information Hiding)
  - Polymorphism
  - Inheritance

What is OOP?

- Modelling real-world objects in software
- Why design applications in this way?
  - We naturally *class*ify objects into different *types*.
  - By attempting to do this with software aim to make it more maintainable, understandable and easier to reuse

What is OOP?

- In a conventional application we typically:
  - decompose it into a series of functions,
  - define data structures that those functions act upon
  - there is no relationship between the two other than the functions act on the data

What is OOP?

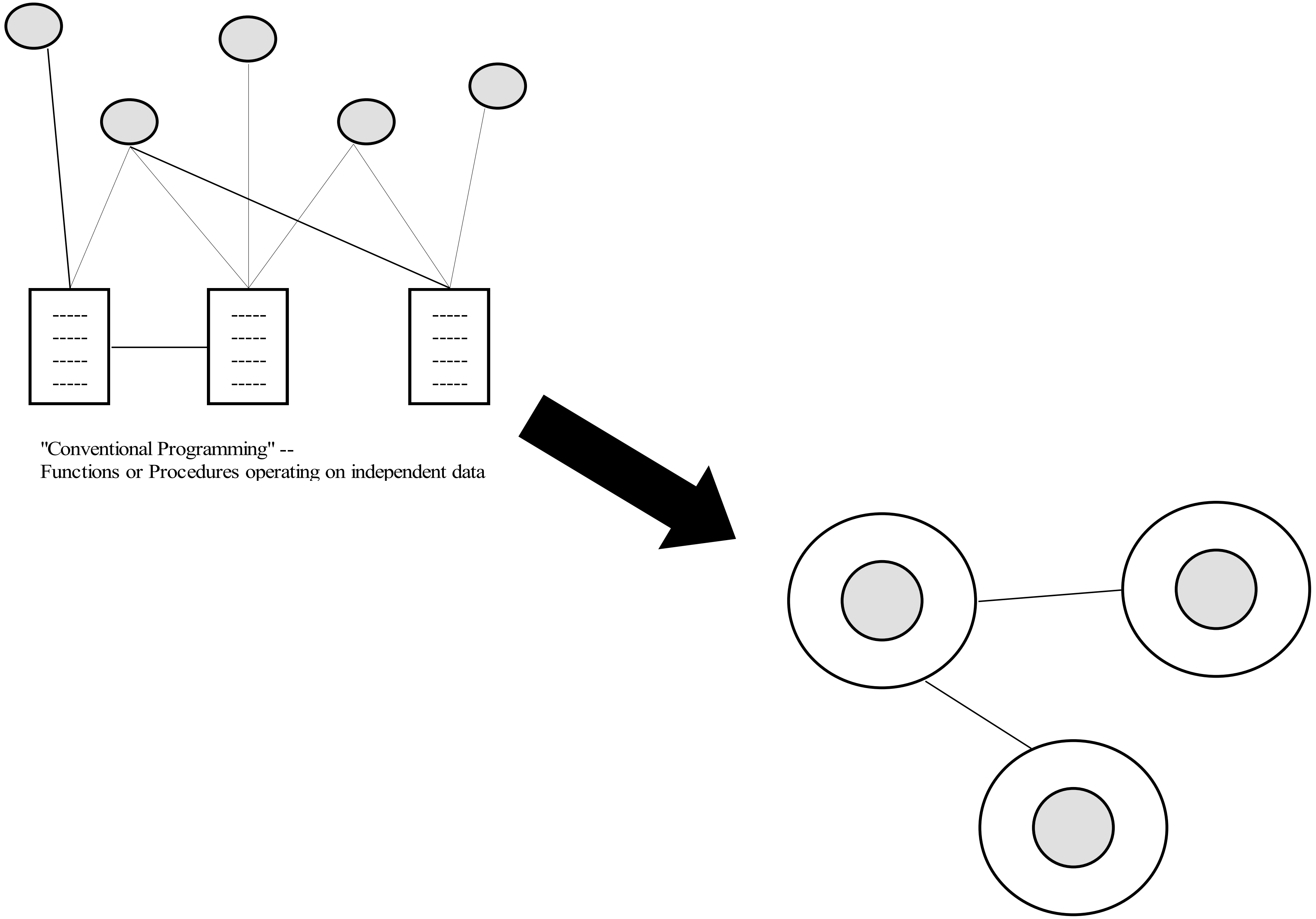
- How is OOP different to conventional programming?
  - Decompose the application into *abstract data types* by identifying some useful entities/abstractions
  - An abstract type is made up of a series of behaviours and the data that those behaviours use.

Abstract Data Types

- Identifying abstract types is part of the modelling/design process
  - The types that are useful to model may vary according to the individual application
- For example a payroll system might need to know about Departments, Employees, Managers, Salaries, etc
  - An E-Commerce application may need to know about Users, Shopping Carts, Products, etc

Abstract Data Types

- Object-oriented languages provide a way to define abstract data types, and then create *objects* from them
  - It’s a template (or ‘cookie cutter’) from which we can create new objects
  - For example, a Car class might have attributes of speed, colour, and behaviours of accelerate, brake, etc
  - An individual Car *object* will have the same behaviours but its own values assigned to the attributes (e.g. 30mph, Red, etc)



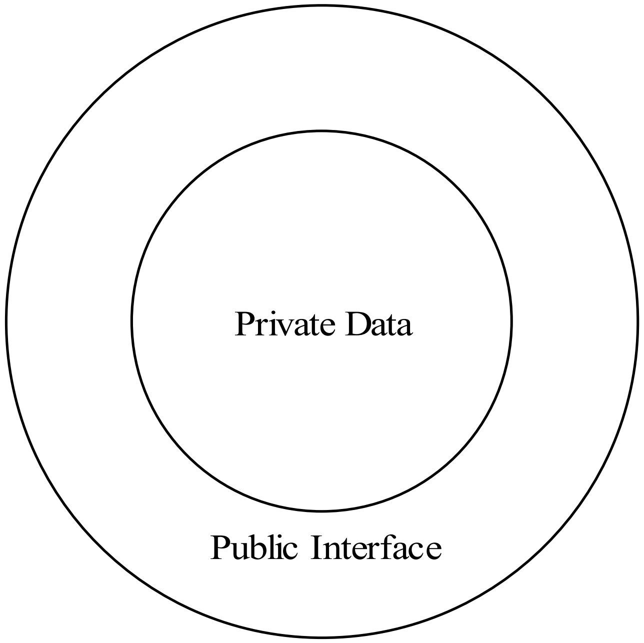
"Conventional Programming" --
Functions or Procedures operating on independent data

"OO Programming" --
Abstract Types combine data and behaviour

Encapsulation

- The data (state) of an object is private – it cannot be accessed directly.
- The state can only be changed through its behaviour, otherwise known as its public *interface* or *contract*
- This is called *encapsulation*

"The Doughnut Diagram" Showing that an object has private state and public behaviour. State can only be changed by invoking some behaviour



Encapsulation

- Main benefit of encapsulation
  - Internal state and processes can be changed independently of the public interface
  - Limits the amount of large-scale changes required to a system

What is an OO program?

- What does an OO program consist of?
  - A series of objects that use each others behaviours in order to carry out some desired functionality
  - When one object invokes some behaviour of another it sends it a *message*
  - In Java terms it invokes a *method* of the other object
  - A method is the implementation of a given behaviour.

What is an OO program?

- OO programs are intrinsically modular
  - Objects are only related by their public behaviour (methods)
  - Therefore objects can be swapped in and out as required (e.g. for a more efficient version)
  - This is another advantage of OO systems

Summary!

- In OO programming we
  - Define classes
  - Create objects from them
  - Combine those objects together to create an application
- Benefits of OO programming
  - Easier to understand (closer to how we view the world)
  - Easier to maintain (localised changes)
  - Modular (classes and objects)