Object Oriented Paradigms

College Requirements

CSCR2205

Overview

- Principles of Object Oriented Programming
- What is OOP?
- Why is it important?
- Basic principles and advantages

Algorithm & Computer Program

- An **algorithm** is a step-by-step process.
- A computer program is a step-by-step set of instructions for a computer.
  - Every computer program is an algorithm.
- Computer programs implement algorithms that manipulate the data.

Problem Solving

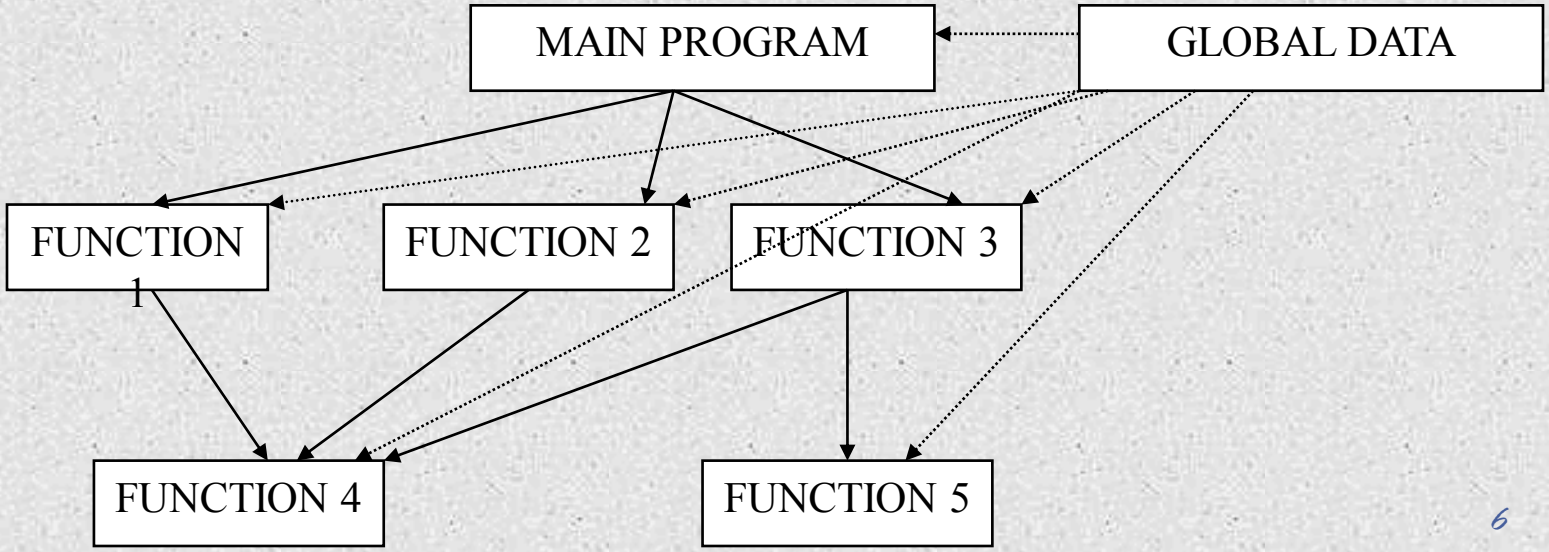
- The purpose of writing a program is to solve a problem
- Solving a problem consists of multiple activities:
  - Understand the problem
  - Design a solution
  - Consider alternatives and refine the solution
  - Implement the solution
  - Test the solution
- These activities are not purely linear – they overlap and interact

Problem Solving

- The key to designing a solution is breaking it down into manageable pieces
- When writing software, we design separate pieces that are responsible for certain parts of the solution
- An *object-oriented approach* lends itself to this kind of solution decomposition
- We will dissect our solutions into pieces called objects and classes

Structured Programming

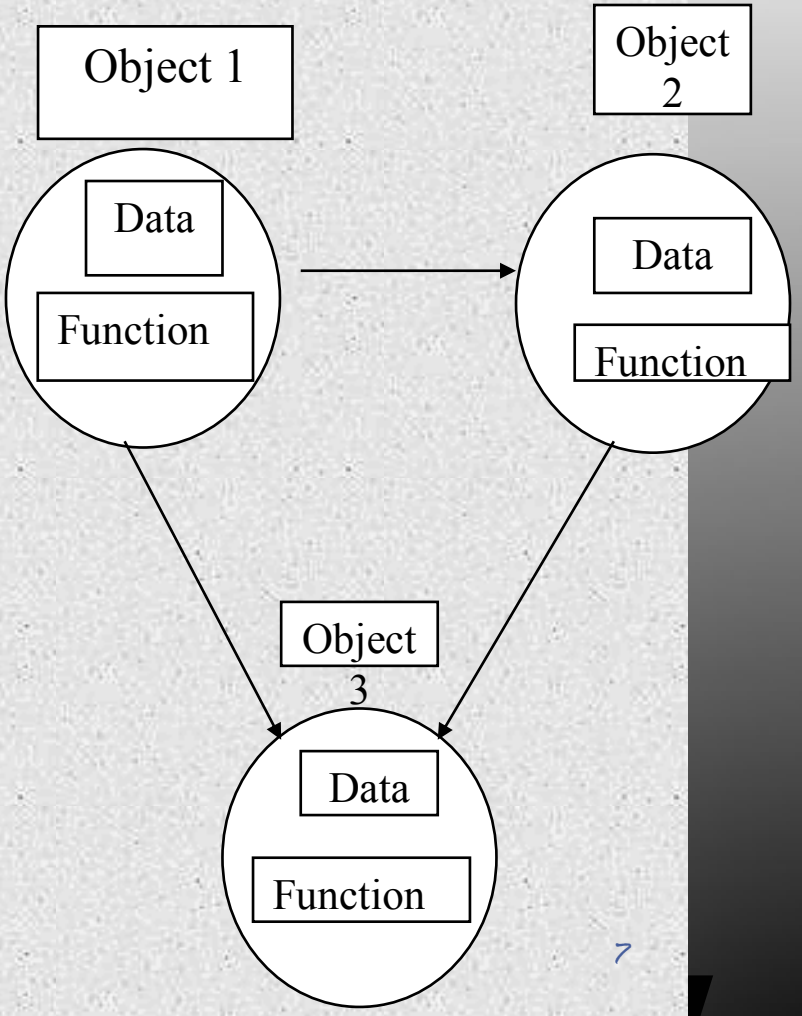
- Using function
- Function & program is divided into modules
- Every module has its own data and function which can be called by other modules.



OBJECT ORIENTED PROGRAMMING

- **Objects** have both **data** and **methods**
- Objects of the same class have the same data elements and methods
- Objects send and receive *messages* to invoke actions.
- **Key idea in object-oriented:**

The real world can be accurately described as a collection of objects that interact.



OBJECT ORIENTED PROGRAMMING

- Computer scientists have introduced the notion of ***objects*** and ***object-oriented programming*** to help manage the growing complexity of modern computers.
- An ***object*** is anything that can be represented by data in a computer's memory and manipulated by a computer program.
- An object can be something in the physical world or even just an abstract idea.
  - An airplane, for example, is a physical object that can be manipulated by a computer.
  - A bank transaction is an example of an object that is not physical.

Object-oriented programming

Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs. C++, Objective-C, Smalltalk, Java and C# are examples of object-oriented programming languages.

Concept: An object has behaviors

- In old style programming, you had:
  - data, which was completely passive
  - functions, which could manipulate any data
- An object contains both **data** and **methods** that manipulate that data.
  - An object is active, not passive; it does things
  - An object is responsible for its own data
    - But: it can expose that data to other objects
- In object-oriented programming, the programs that manipulate the properties of an object are the object's methods.

What is OOP?

- Modelling real-world objects in software
- Why design applications in this way?
  - We naturally classify objects into different types.
  - By attempting to do this with software aim to make it more maintainable, understandable and easier to reuse
- 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.
- Similar to database modelling, only the types have both behaviour and state (data)

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.

OOP Features

- Emphasis on data rather than procedure
- Programs are divided into entities known as objects.
- Data Structures are designed such that they characterize objects.
- Functions that operate on data of an object are tied together in data structures.
- Data is hidden and cannot be accessed by external functions.
- Objects communicate with each other through functions.
- New data and functions can be easily added whenever necessary.
- Follows bottom up design in program design

Object-Oriented Programming

- Understanding OOP is fundamental to writing good Java applications
  - Improves design of your code
  - Improves understanding of the Java APIs
- There are several concepts underlying OOP:
  - Object.
  - Abstract Types (Classes)
  - Encapsulation (or Information Hiding)
  - Inheritance
  - Polymorphism

Why OO-Programming?

- Reduces conceptual load
  - By reducing amount of detail
- Provides fault containment
  - Can't use components (e.g., a class) in inappropriate ways
- Provides independence between components
  - Design/development can be done by more than one person
- Save development time (and cost) by reusing code
  - Once an object class is created it can be used in other applications
- Easier debugging
  - Classes can be tested independently

Design Principles of OOP

- Four main design principles of Object-Oriented Programming(OOP):
  - Encapsulation
  - Abstraction
  - Polymorphism
  - Inheritance

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
  - An E-Commerce application may need to know about Users, Shopping Carts, Products, etc
- 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

Encapsulation

- Encapsulation means
  - 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
- 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

Inheritance

- Inheritance is the ability to define a new class in terms of an existing class.
  - The existing class is the parent, base or superclass
  - The new class is the child, derived or subclass
- The child class inherits all of the attributes and behaviour of its parent class
  - It can then add new attributes or behaviour
  - Or even alter the implementation of existing behaviour
- Inheritance is therefore another form of code reuse.

Polymorphism

- Means ‘many forms’
- Difficult to describe, easier to show, so we’ll look at this one in a later lesson
- In brief though, polymorphism allows two different classes to respond to the same message in different ways
- E.g. both a Plane and a Car could respond to a ‘turnLeft’ message,
  - however the means of responding to that message (turning wheels, or banking wings) is very different for each.
- Allows objects to be treated as if they’re identical

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)
  - Good level of code reuse (aggregation and inheritance)

NOW:

Waiting for your questions and comments

Lecture 2.1

Objects and Classes.