The philosophy of OO programming

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming Concepts using Java

Week 3

Algorithms + Data Structures = Programs

- Title of Niklaus Wirth's introduction to Pascal

Algorithms + Data Structures = Programs

- Title of Niklaus Wirth's introduction to Pascal
- Traditionally, algorithms come first

Algorithms + Data Structures = Programs

- Title of Niklaus Wirth's introduction to Pascal
- Traditionally, algorithms come first
- Structured programming
  - Design a set of procedures for specific tasks
  - Combine them to build complex systems

Algorithms + Data Structures = Programs

- Title of Niklaus Wirth's introduction to Pascal
- Traditionally, algorithms come first
- Structured programming
  - Design a set of procedures for specific tasks
  - Combine them to build complex systems
- Data representation comes later
  - Design data structures to suit procedural manipulations

Object Oriented design

- Reverse the focus

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate
- Then identify algorithms to operate on the data

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate
- Then identify algorithms to operate on the data
- Claim: works better for large systems

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate
- Then identify algorithms to operate on the data
- Claim: works better for large systems
- Example: simple web browser
  - 2000 procedures manipulating global data

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate
- Then identify algorithms to operate on the data
- Claim: works better for large systems
- Example: simple web browser
  - 2000 procedures manipulating global data
  - . . . vs 100 classes, each with about 20 methods

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate
- Then identify algorithms to operate on the data
- Claim: works better for large systems
- Example: simple web browser
  - 2000 procedures manipulating global data
  - . . . vs 100 classes, each with about 20 methods
  - Much easier to grasp the design

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate
- Then identify algorithms to operate on the data
- Claim: works better for large systems
- Example: simple web browser
  - 2000 procedures manipulating global data
  - . . . vs 100 classes, each with about 20 methods
  - Much easier to grasp the design
  - Debugging: an object is in an incorrect state

Object Oriented design

- Reverse the focus
- First identify the data we want to maintain and manipulate
- Then identify algorithms to operate on the data
- Claim: works better for large systems
- Example: simple web browser
  - 2000 procedures manipulating global data
  - . . . vs 100 classes, each with about 20 methods
  - Much easier to grasp the design
  - Debugging: an object is in an incorrect state
  - Search among 20 methods rather than 2000 procedures

Object Oriented design: Example

- An order processing system typically involves
  - Items
  - Orders
  - Shipping addresses
  - Payments
  - Accounts

Object Oriented design: Example

- An order processing system typically involves
  - Items
  - Orders
  - Shipping addresses
  - Payments
  - Accounts
- What happens to these objects?
  - Items are added to orders
  - Orders are shipped, cancelled
  - Payments are accepted, rejected

Object Oriented design: Example

- An order processing system typically involves
  - Items
  - Orders
  - Shipping addresses
  - Payments
  - Accounts
- What happens to these objects?
  - Items are added to orders
  - Orders are shipped, cancelled
  - Payments are accepted, rejected
- Nouns signify objects, verbs denote methods that operate on objects
  - Associate with each order, a method to add an item

Designing objects

- Behaviour — what methods do we need to operate on objects?

Designing objects

- Behaviour — what methods do we need to operate on objects?
- State — how does the object react when methods are invoked?
  - State is the information in the instance variables
  - Encapsulation — should not change unless a method operates on it

Designing objects

- Behaviour — what methods do we need to operate on objects?
- State — how does the object react when methods are invoked?
  - State is the information in the instance variables
  - Encapsulation — should not change unless a method operates on it
- Identity — distinguish between different objects of the same class
  - State may be the same — two orders may contain the same item

Designing objects

- Behaviour — what methods do we need to operate on objects?
- State — how does the object react when methods are invoked?
  - State is the information in the instance variables
  - Encapsulation — should not change unless a method operates on it
- Identity — distinguish between different objects of the same class
  - State may be the same — two orders may contain the same item
- These features interact
  - State will typically affect behaviour
  - Cannot add an item to an order that has been shipped
  - Cannot ship an empty order

Relationship between classes

- Dependence
  - `Order` needs `Account` to check credit status
  - `Item` does not depend on `Account`
  - Robust design minimizes dependencies, or coupling between classes

Relationship between classes

- Dependence
  - `Order` needs `Account` to check credit status
  - `Item` does not depend on `Account`
  - Robust design minimizes dependencies, or coupling between classes
- Aggregation
  - `Order` contains `Item` objects

Relationship between classes

- Dependence
  - `Order` needs `Account` to check credit status
  - `Item` does not depend on `Account`
  - Robust design minimizes dependencies, or coupling between classes
- Aggregation
  - `Order` contains `Item` objects
- Inheritance
  - One object is a specialized versions of another
  - `ExpressOrder` inherits from `Order`
  - Extra methods to compute shipping charges, priority handling

Summary

- An object-oriented approach can help organize code in large projects
- This course is not about software engineering
- Nevertheless, useful to know the motivation underlying OO programming to understand design choices in a programming language like Java