BACS2063 Data Structures and Algorithms

Abstract Data Types (ADTs)

Chapter 3

Learning Outcomes

At the end of this lecture, you should be able to

- Explain the benefits of encapsulation and data abstraction.
- Write abstract data type (ADT) specifications.
- Implement ADTs using Java interfaces and classes.

Winning Strategies in Programming

Increase Productivity	• Projects can be finished on time • More projects can be handled
Assure Quality	• Reliable: bug-free

How to … ***increase productivity*** and ***assure quality***?

Code / component reuse

Code maintainability

To achieve *reuse*

Abstraction	• Abstract data type (ADT) specification
Encapsulation	• ADT implementation

To achieve *maintainability*

Encapsulation / Information Hiding

- ADT implementation

Benefits of

Abstraction

- Can focus on the general (abstract) properties without worrying about how it is going to be implemented.

Encapsulation

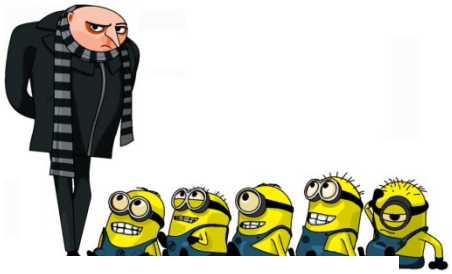
- Can use the components without knowing the implementation details.

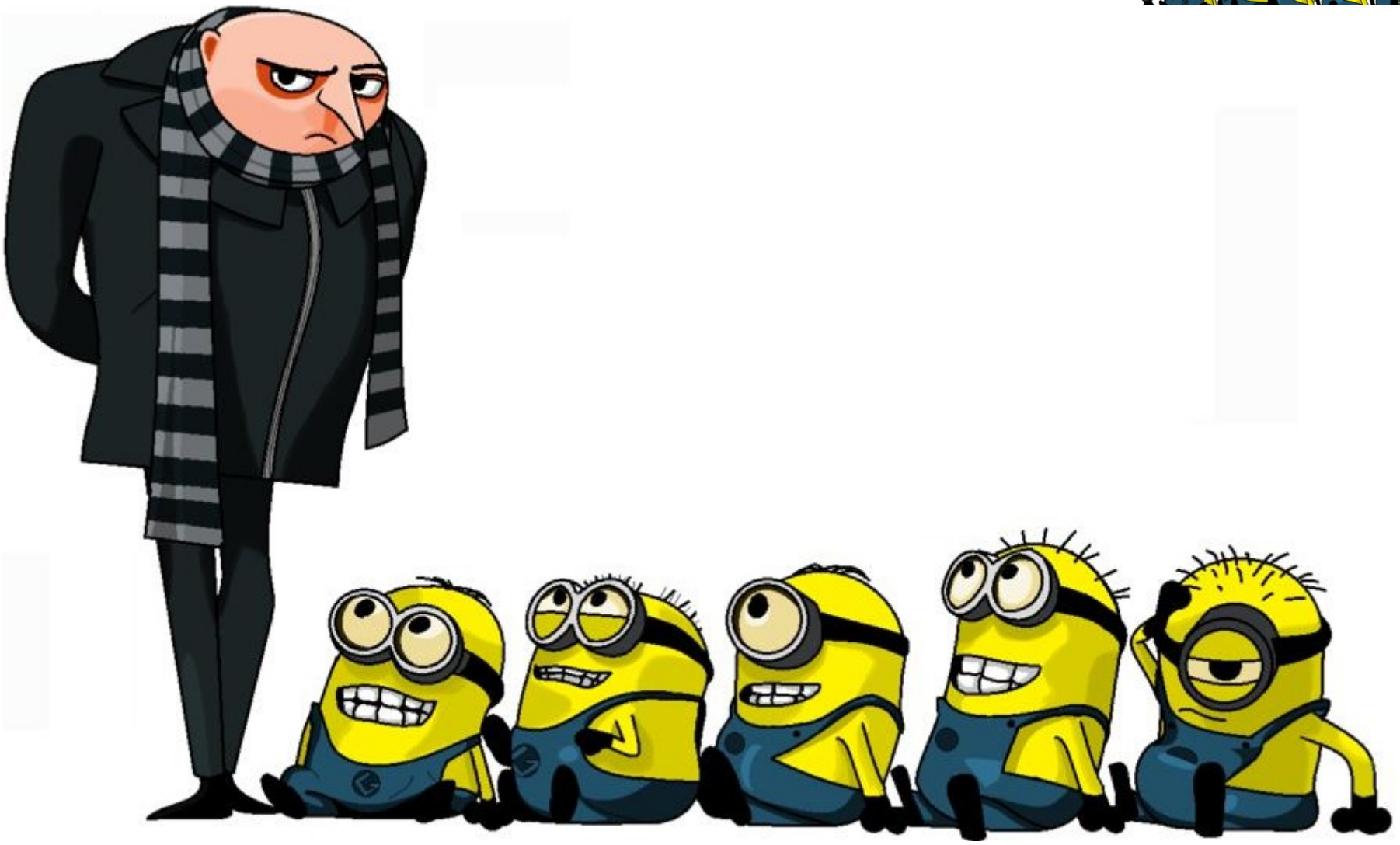
Abstraction & Encapsulation

Abstraction	• Specifying the data type
Encapsulation	• Using the data type

2 sides of the same coin!

Case Study





Gru's observation

- Code duplication – reinventing the wheel
- Not-so-maintainable code

MinionSoft's code duplication

- The list is a "thing" that appears in all of the software applications.
- It has specific characteristics
  - Data in the list is organized in a certain way
  - There are certain operations that are performed on the list
- The declaration for the array to hold the list elements and the coding for the operations were repeated in all the software applications!!

MinionSoft's not-so-maintainable code

- If a bug is discovered, the changes to the code would need to be applied to every single software application component which used a list.

How to…

- Enable reuse?
- Increase maintainability?

Solution Steps

1. **Abstraction**
   - Identify the common/general/abstract properties of the list.
     - ➢ **An abstract data type!**
   - Produce a specification of the list
     - What are the characteristics of the data?
     - What are the operations for manipulating the data?

Solution Steps (cont'd)

2. **Encapsulation**
   - Implement the list in such a way that a programmer can use the list without knowing how it is implemented.
     - ➢ **Information hiding!**

❖ An *ADT* is

"An abstraction of a commonly appearing data structure along with a set of defined operations on the data structure. It specifies the logical properties without the implementation details. In Java, ADT is achieved through the use of classes."

Malik & Nair (2003)

ADT Specifications are

- Written in a natural language (e.g. English) and are independent of any programming language.
- Used as specifications for concrete data types (i.e. the actual data types used in programs).

What to include in an ADT specification?

- ADT title
- Description of the characteristics (logical properties) of the data type
- Description of each operation:
  - Brief description of what the operation does
  - Precondition (if any)
  - Postcondition (if any)
  - What is returned by the operation (if any)

Preconditions and Postconditions

Precondition

- A statement specifying the condition(s) that must be true before the operation is invoked.

Postcondition

- A statement specifying what is true after the operation is completed.

– Malik & Nair (2003)

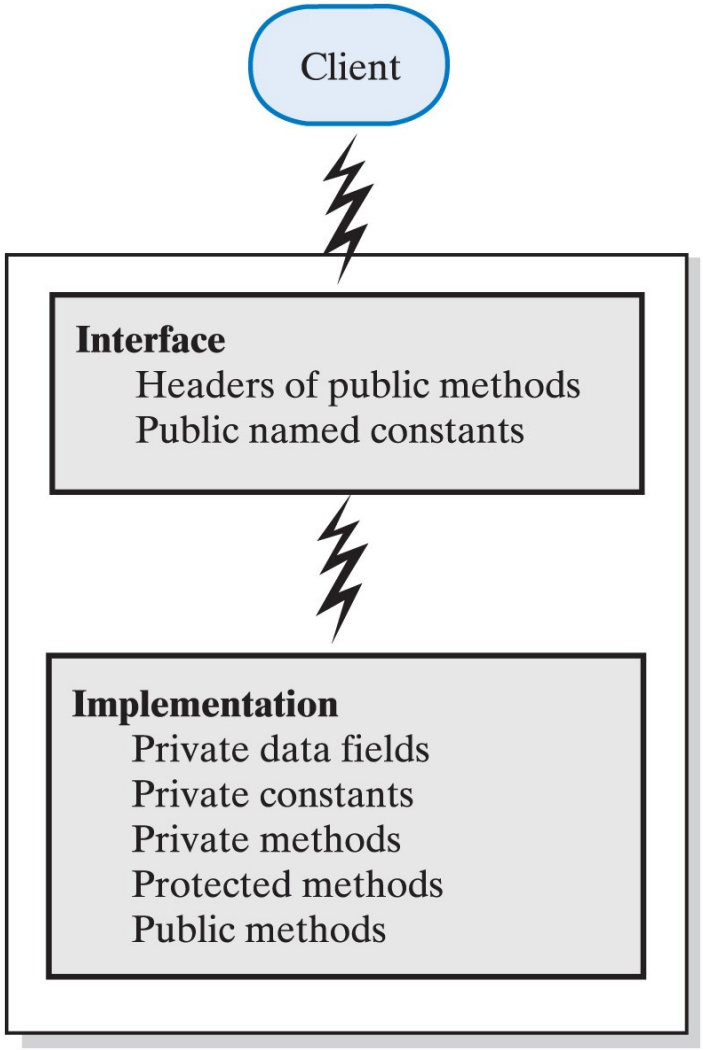
Problem: A Counter ADT

- Counter devices are used for counting things such as cars entering a parking lot, people taking numbers at the post-office, etc. A counter object would have a non-negative integer value representing the current count. It can be incremented, decremented, reset to zero and have its value read at any time.
- Specify a Counter ADT whose instances would represent counter objects.

Implementing Encapsulation

- In object-oriented programming, encapsulation is achieved in a class by
  - Making its data fields private, and
  - Providing public methods for controlled access to and manipulation of the data fields

Data Abstraction



An interface provides well-regulated communication between a hidden implementation and a client.

Carrano (2011)

To implement an ADT in Java

1. Translate the ADT specification into a Java interface
2. Write a class which implements the Java interface

Sample Code: Counter ADT

In **`Chapter3\carpark\`**

- **`CounterInterface.java`**
- **`Counter.java`**
- **`CarParkSystem.java`**
  - A GUI application which simulates the use of the counter in a car park with 2 wings

Problem: A collection that does not allow duplicate values - the **Set ADT**

1. Specify the ADT
   - Describe the characteristic
   - Identify the operations
   - Write descriptions for the operations
2. Implement the ADT

Design Principles: Recall

- Encapsulation
- Modularity
- Abstraction

A data structure is an abstract data type

Abstract Data Type (ADT)

- Defines the structure, behavior and operations of a data structure without specifying how those structure, behavior and operations are actually implemented [5]

ADT: Rationale

- Separation of Concerns
  - By separating the definition of the data structure from the implementation, we can use the new data structure in programs without regard for its implementation.
  - Hence, the implementation can be changed (improved, updated, etc) and the client programs (i.e. the programs that use the data structure) will only have to change the name of the class that implements the new data structure.

Exercise: Representing a Purse

- Write the specification for a `Purse ADT` whose instances would represent coin purses that hold 5-sen, 10-sen, 20-sen and 50-sen coins. Include operations for getting the number of x-sen coins in the purse, adding n number of x-sen coins to the purse and sum up the number of all the coins in the purse. You should include an ADT statement as well as precondition, postcondition and return statement for each of the operation in the `Purse ADT` .

- Translate your ADT into a Java interface class named purse.

Exercise: Representing Time

a) Define a `Time ADT` to represent the time of day (hour, minute and second) using the 24-hour clock format.

Specify appropriate operations for the ADT by writing

- A brief description of each operation
- The operation's preconditions and postconditions (if any)
- What will be returned by the operation

b) Translate your ADT into a Java interface named **TimeInterface.**

c) Implement the interface **TimeInterface** in a class named **Time.**

Learning Outcomes

You should now be able to

- Explain the benefits of encapsulation and data abstraction.
- Write abstract data type (ADT) specifications.
- Implement ADTs using Java interfaces and classes.