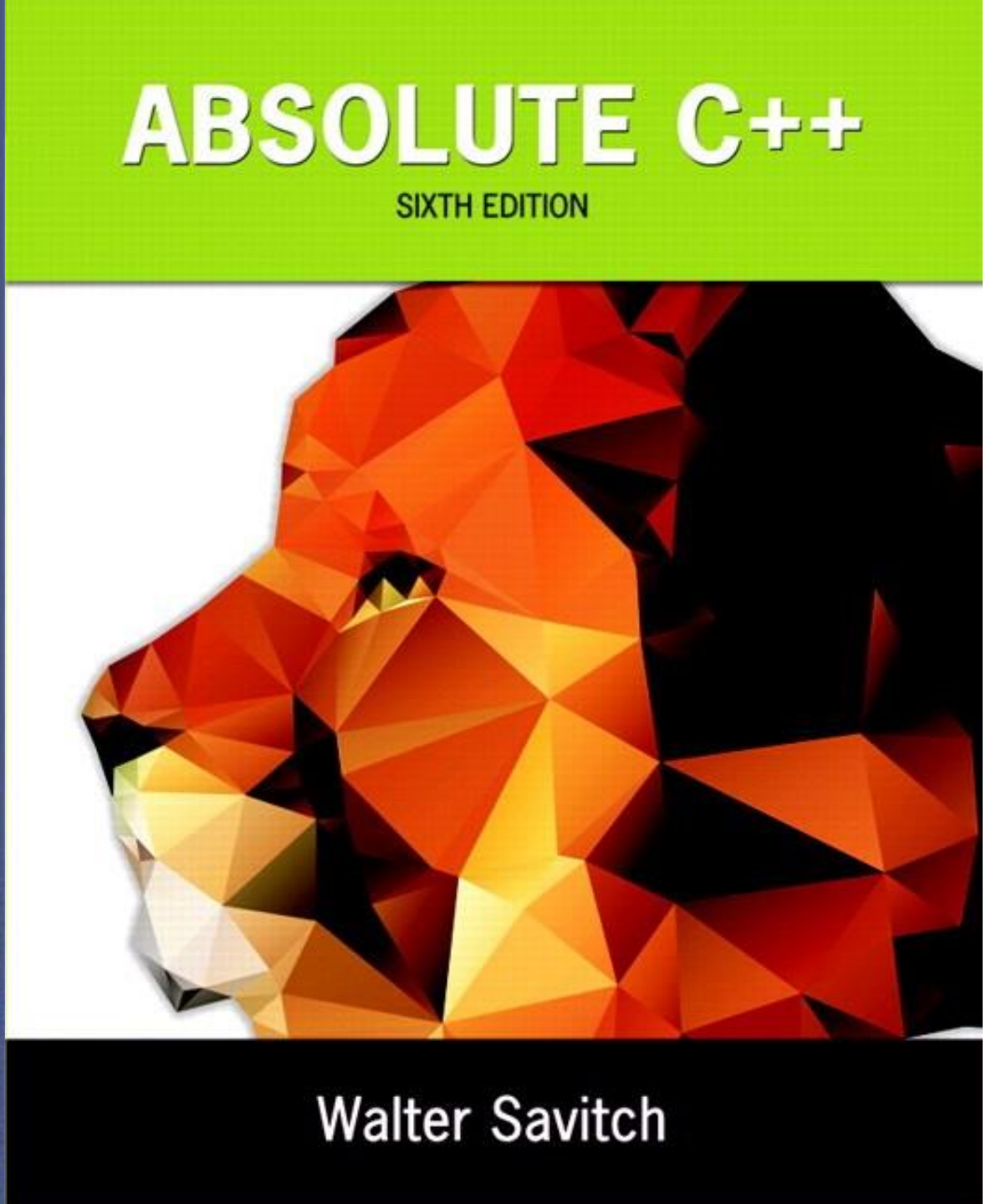

Chapter 11

Separate Compilation and Namespaces

Copyright © 2016 Pearson, Inc.
All rights reserved.

PEARSON

Learning Objectives

- Separate Compilation
  - Encapsulation reviewed
  - Header and implementation files

Copyright © 2016 Pearson Inc. All rights reserved.

Separate Compilation

- Program Parts
  - Kept in separate files
  - Compiled separately
  - Linked together before program runs
- Class definitions
  - Separate from "using" programs
  - Build library of classes
    - Re-used by many different programs
    - Just like predefined libraries

Copyright © 2016 Pearson Inc. All rights reserved.

Class Separation

- Class Independence
  - Separate class definition/specification
    - Called "interface"
  - Separate class implementation
  - Place in two files
- If implementation changes → only that file need be changed
    - Class specification need not change
    - "User" programs need not change

Copyright © 2016 Pearson Inc. All rights reserved.

Encapsulation Reviewed

- Encapsulation principle:
  - Separate how class is used by programmer from details of class's implementation
- "Complete" separation
  - Change to implementation → NO impact on any other programs
- Basic OOP principle

Copyright © 2016 Pearson Inc. All rights reserved.

Encapsulation Rules

- Rules to ensure separation:
  1. All member variables should be private
  2. Basic class operations should be:
     - Public member functions
     - Friend or ordinary functions
     - Overloaded operators

     Group class definition and prototypes together
     - Called "interface" for class
  3. Make class implementation unavailable to users of class

Copyright © 2016 Pearson Inc. All rights reserved.

More Class Separation

- Interface File
  - Contains class definition with function and operator declarations/prototypes
  - Users "see" this
  - Separate compilation unit
- Implementation File
  - Contains member function definitions
  - Separate compilation unit

Copyright © 2016 Pearson Inc. All rights reserved.

Class Header Files

- Class interface always in header file
  - Use .h naming convention
- Programs that use class will "include" it
  - #include "myclass.h"
  - Quotes indicate you wrote header
    - Find it in "your" working directory
  - Recall library includes, e.g., <iostream>
    - < > indicate predefined library header file
    - Find it in library directory

Copyright © 2016 Pearson Inc. All rights reserved.

Class Implementation Files

- Class implementation in .cpp file
  - Typically give interface file and implementation file same name
    - myclass.h and myclass.cpp
  - All class's member function defined here
  - Implementation file must #include class's header file
- .cpp files in general, typically contain executable code
  - e.g., Function definitions, including main()

Copyright © 2016 Pearson Inc. All rights reserved.

Class Files

- Class header file #included by:
  - Implementation file
  - Program file
    - Often called "application file" or "driver file"
- Organization of files is system dependent
  - Typical IDE has "project" or "workspace"
    - Implementation files "combined" here
    - Header files still "#included"

Copyright © 2016 Pearson Inc. All rights reserved.

Multiple Compiles of Header Files

- Header files
  - Typically included multiple times
    - e.g., class interface included by class implementation and program file
  - Must only be compiled once!
  - No guarantee "which #include" in which file, compiler might see first
- Use preprocessor
  - Tell compiler to include header only once

Copyright © 2016 Pearson Inc. All rights reserved.

Using #ifndef

- Header file structure:
  - #ifndef FNAME_H
    #define FNAME_H
    … //Contents of header file
    …
    #endif
- FNAME typically name of file for consistency, readability
- This syntax avoids multiple definitions of header file

Copyright © 2016 Pearson Inc. All rights reserved.

Other Library Files

- Libraries not just for classes
- Related functions
  - Prototypes → header file
  - Definitions → implementation file
- Other type definitions
  - structs, simple typedefs → header file
  - Constant declarations → header file

Copyright © 2016 Pearson Inc. All rights reserved.

Summary 1

- Can separate class definition and implementation → separate files
  - Separate compilation units

Copyright © 2016 Pearson Inc. All rights reserved.