Software Development

CS2500: Syllabus

M.R.C. van Dongen

September 23, 2013

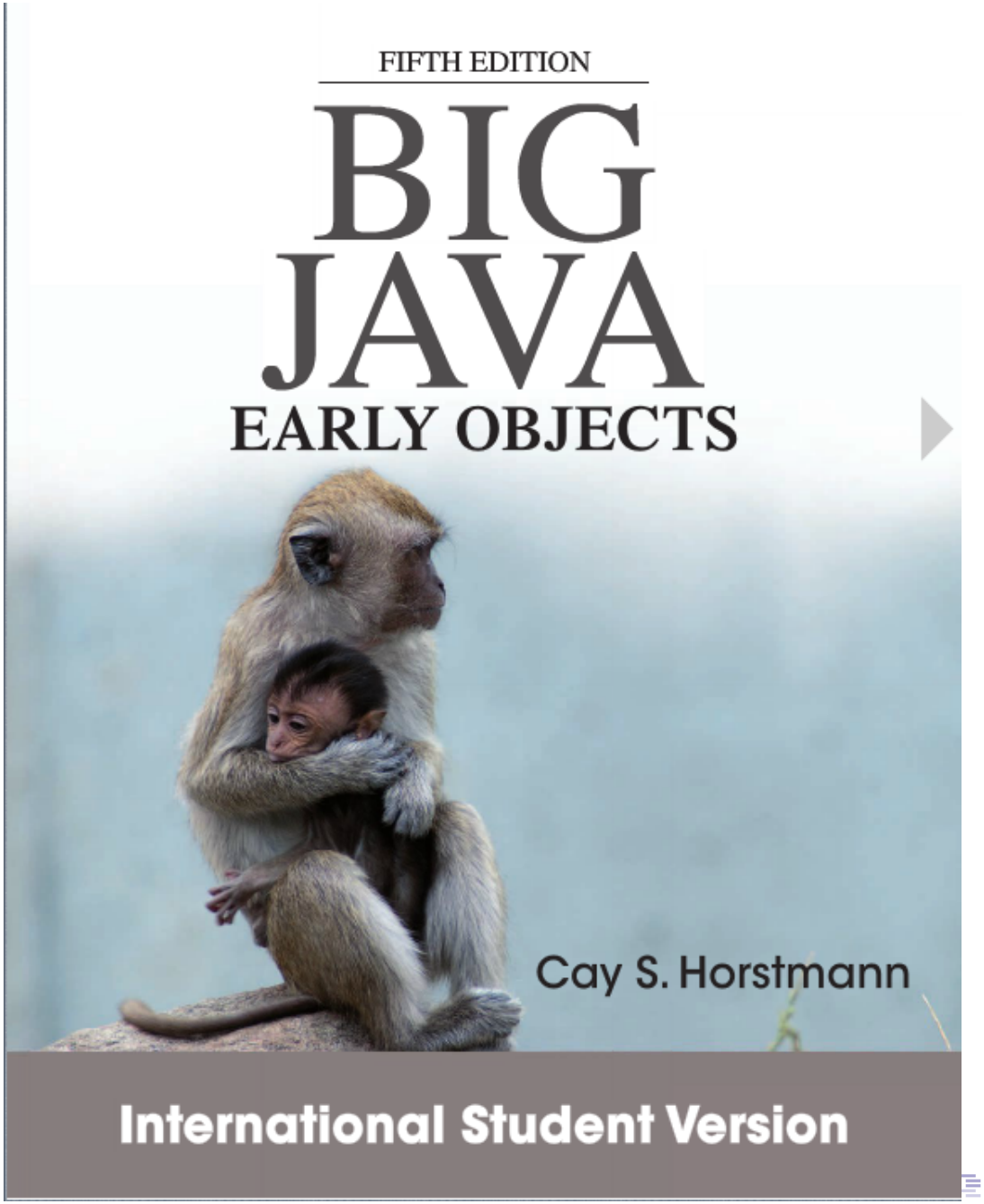
Main Topics

Basic Java: OO programming in Java.

Software development: Good OO software development.

Design: Design patterns and principles.

Literature


FIFTH EDITION
BIG
JAVA
EARLY OBJECTS
Cay S. Horstmann
International Student Version

Literature

- The course is based on the following book:

  Required: Big Java, *Early Objects, by Cay S. Horstmann, Wiley,* ISBN: *978-1-118-31877-5.*

- We will use some material from the following book:

  Optional: *Head First Design Patterns, by Eric and Elisabeth Freeman, O'Reilly,* ISBN: *978-0-596-00920-5.*

- The following are *great* books:

  Recommended: *Effective Java, by Joshua Bloch, Addison-Wesley,* ISBN: *978-0-321-35668-0.*

  Recommended: *Java Generics, by Maurice Naftalin and Philip Wadler, O'Reilly,* ISBN: *978-0-596-52775-4.*

Teaching Methods

- There will be three weekly lectures:

  Monday: 2 p.m.–3 p.m. in WGB G03 (P1)/ WGB 15 (P2);
  Wednesday: 2 p.m.–3 p.m. in WGB G01 (P1) / WGB 08 (P2); and
  Friday: 11 a.m.–12 m. in WGB G01.

- There will be weekly labs:

  Monday: 4pm–6pm in WGB G24.
  Tuesday: 4pm–6pm in WGB G24.

Assessment

Total marks: 300;

Written exam: 1×3 hour exam: 225 marks;

Continuous assessment: 75 marks:

- 10 programming assignments (3 marks each);
- End-of-Term Test in December (20 marks);
- Mid-Term Test in February (10 marks);
- End-of-Year Test in March (15 marks).

Officially

- Officially, students are allowed to repeat the Xmas test.
  - This is a lot of work;
  - If they fail, they lose the marks they had.
- Proposal:
  - Continuous assessment does not carry forward;
  - Repeat students pass if they pass the written exam in the Autumn.
  - For this to work, I need a signature of all student in the class.

Assignment Submission

- Using `Moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you too late for the official deadline,
  - You may still submit in the grace period.

Assignment Submission

- Using `Moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you too late for the official deadline,
  - You may still submit in the grace period.
- When you too late for the grace deadline

Assignment Submission

- Using `Moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you too late for the official deadline,
  - You may still submit in the grace period.
- When you too late for the grace deadline,
  - You’re too late.

Assignment Submission

- Using `Moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you too late for the official deadline,
  - You may still submit in the grace period.
- When you too late for the grace deadline,
  - You’re too late.
  - No submission allowed.

Assignment Submission

- Using `Moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you too late for the official deadline,
  - You may still submit in the grace period.
- When you too late for the grace deadline,
  - You’re too late.
  - No submission allowed.
  - Sorry about that.

Assignments

- You get a grade for each submitted assignment.
- You get feedback about what is good and bad.
- You get marks only if your submission compiles and is a *reasonable* attempt.

Learning Outcomes: Java

- Classes and objects;
- Variables, types, and declarations; Arrays; Loops and branching;
- Attributes and methods: state and behaviour;
- Object creation and garbage collection;
- Wrapper classes and autoboxing;
- Encapsulation and visibility modifiers;
- `Math` and `Java` libraries;
- Inheritance; polymorphism; Overriding versus overloading;
- Is-a versus has-a versus owns-a:
  - Extends versus aggregates versus composes;
- Constructor chaining;
- Delegation, aggregation, and composition;
- Abstraction; Abstract classes and interfaces; Static methods;
- Enumerated types (in-depth);
- Risky behaviour: exceptions; files and I/O;
- Threads;
- Graphical User Interfaces (GUIs) and GUI component classes;
- `Swing` library; Action events and event listeners; Inner classes;
- Generics & collections; Common collection types (sets/lists/maps/...).

Learning Outcomes: Software Development

- `JavaDoc` and guidelines for program layout and presentation;
- `JUnit` testing;
- Releasing your code;
- Version control.

Learning Outcomes: Design Principles

- A *design principle* is a basic tool or technique that can be applied to designing or writing code to make that code more maintainable, flexible, or extensible.
- You will learn useful design principles and how to use them.

Learning Outcomes: Design Patterns

Strategy: Defines a family of algorithms, encapsulates each one, and makes them interchangeable.

Observer: Defines a one-to-many dependency between a *subject* and *subscribers*, so that if the subject changes, all its subscribers are notified automatically.
(`Observer-Listener` in `Swing`.)

Factory: Separates object creation from the class that defines the object (avoids `new`) and hides the details of object creation behind a well-defined interface.

Singleton: Ensures that a class has (at most) one instance, and provides a global point of access to the instance.

Iterator: Provides a sequential access to the members of an aggregate object without exposing the object's underlying representation.

First Lectures

- We will cover the book from cover to cover.
  - No handouts for these lectures.
- We also study some things that are not in the book.
  - I shall provide handouts for these lectures.

Contract

- I want you to think about your implementation.
- You need to show me you know what you are doinging.
- Aim at *clean* programs.
  - Some programming styles discourage clean programming.
    - Multiple exit points: `break`;
    - “Overloading” `for`/`while` statements;
    - ...
  - Avoid these or you will lose marks.

Contact Details

Lecturer	Marc van Dongen
Room	WGB 1-74
Phone	+353 (0)21 4205903
Electronic mail	dongen@cs.ucc.ie
Snail mail	Computer Science Department University College Cork Western Road Cork

Almost Finally

- I'm teaching this module for the third time;
- I know it's not difficult:
  - Provided you attend the lectures and *study*.
- If you find programming difficult or if you're repeating:
  - Make an appointment with me.

Acknowledgements

The picture in the presentation and some of the text in the literature section are from a screen shot from the publisher's website, O'Reilly, and Wikipedia.

Bibliography

- Bloch, Joshua [2008]. *Effective Java*. Addison–Wesley. ISBN: 978-0-321-35668-0.
- Freeman, Eric, and Elisabeth Freeman [2005]. *Head First Design Patterns*. O'Reilly. ISBN: 978-0-596-00920-5.
- Horstfmann, Cay S. *Big Java, Early Objects*. International Student Version. Wiley. ISBN: 978-1-118-31877-5.
- McLaughlin, Brett D., Gary Pollice, and David West [2007]. *Head First Object-Oriented Analysis & Design*. O'Reilly. ISBN: 978-0-596-00867-3.
- Naftalin, Maurice, and Philip Wadler [2009]. *Java Generics*. O'Reilly. ISBN: 978-0-596-52775-4.
- Sierra, Kathy, and Bert Bates [2004]. *Head First Java*. O'Reilly. ISBN: 978-0-596-00712-6.

About this Document

- This document was created with `pdflatex`.
- The LaTeX document class is `beamer`.