SWEN20003
Object Oriented Software Development

Revision - 1

Shanika Karunasekera
karus@unimelb.edu.au

University of Melbourne

Topics Covered So Far

- Subject Introduction
- A Quick Tour of Java (Java Introduction)
- Classes and Objects
- Arrays and Strings
- Input and Output
- Software Tools
- Inheritance and Polymorphism
- Interfaces and Polymorphism

Lecture Objectives

- Respond to any mid-semester test related questions
- Review the subject learning outcomes so far

Mid-semester Test

This information is the same as what was given in the announcement on Canvas a few weeks ago.

- Will be a 40-minute test, which includes multiple choice and short answer questions, administered via the Canvas Quiz facility.
- The test has been designed to test your knowledge on the contents taught up to end of week 6, as we would test it in a non-online setting (under normal exam conditions), so you are expected to know the content without having to look up subject material or external resources during the exam.

Mid-semester Test

Cont...

- Recognizing the practical constraints in imposing exam conditions in the online context, to be fair to everybody, we have decided to make the exam open-resources (which means you are allowed to look up anything including searching the Internet), but if you do not understand/revise the material and choose to rely on referring to material during the exam, most probably you will run out of time to complete the exam, because the timing is set assuming you do not have to spend time referring to material.
- Even if you refer to material from other sources, you are not allowed do directly copy anything including from subject material the answers must be your own.
- A practice mid-semester exam will be available.

A Quick Tour of Java

Learning Outcomes:

- Identify some of the key Java features
- Understand the following in context of Java:
  - Identifiers, Data Types, Variables and Constants
  - Operators and Expressions
  - Flow of control
- Write simple Java programs

Classes and Objects

Learning Outcomes:

- Explain the difference between a *class* and an **object**
- Define classes, and give them **properties** and **behaviours**
- Implement and use classes
- Identify a series of well-defined classes from a **specification**
- Explain object oriented concepts: **abstraction, encapsulation, information hiding** and **delegation**
- Understand the role of **Wrapper** classes

Arrays and Strings

Learning Outcomes:

- Understand how to use **Arrays**
- Understand how to use **Strings**

Input and Output

Learning Outcomes:

- Accept input to your programs through:
  - Command line arguments
  - User input
  - Files
- Write output from your programs through:
  - Standard output (terminal)
  - Files
- Use files to store and retrieve data during program execution
- Manipulate data in files (i.e. for computation)

Software Tools

Learning Outcomes:

- Use **git** for **software version control**
- Use **Maven** to **manage software builds**
- Use the **IntelliJ Debugger** to find program bugs
- Use the software package **Bagel** (Basic Academic Graphical Engine Library)
- Contribute to **Open Source Software** (OSS) projects

Inheritance and Polymorphism

Learning Outcomes:

- Use **inheritance** to abstract common properties of classes
- Explain the relationship between a **superclass** and a **subclass**
- Make better use of **privacy** and **information hiding**
- Identify errors caused by **shadowing** and **privacy leaks**, and avoid them
- Describe and use method **overriding**
- Describe the **Object** class, and the properties inherited from it
- Describe what **upcasting** and **downcasting** are, and when they would be used
- Explain **polymorphism**, and how it is used in Java
- Describe the purpose and meaning of an **abstract** class

Interfaces and Polymorphism

Learning Outcomes:

- Describe the purpose and use of an **interface**
- Describe what it means for a class to **use** an interface
- Describe when it is appropriate to use inheritance vs. interfaces
- Use interfaces and inheritance to achieve powerful abstractions
- Make any class "sortable"