Programming and Software Development
COMP90041

Semester 2, 2018

Dr. Thomas Christy
thomas.christy@unimelb.edu.au
Level: 08 Room: 11, Doug McDonell, Parkville.

Slides and materials were developed by Prof Peter Schachte

Object-Oriented (OO) software development:

- The Java programming language
- OO concepts
  - Classes
  - Objects
  - Encapsulation
  - Inheritance
  - Polymorphism
- Problem Solving
- Small-scale program design, implementation and testing.

Focus of the subject 2

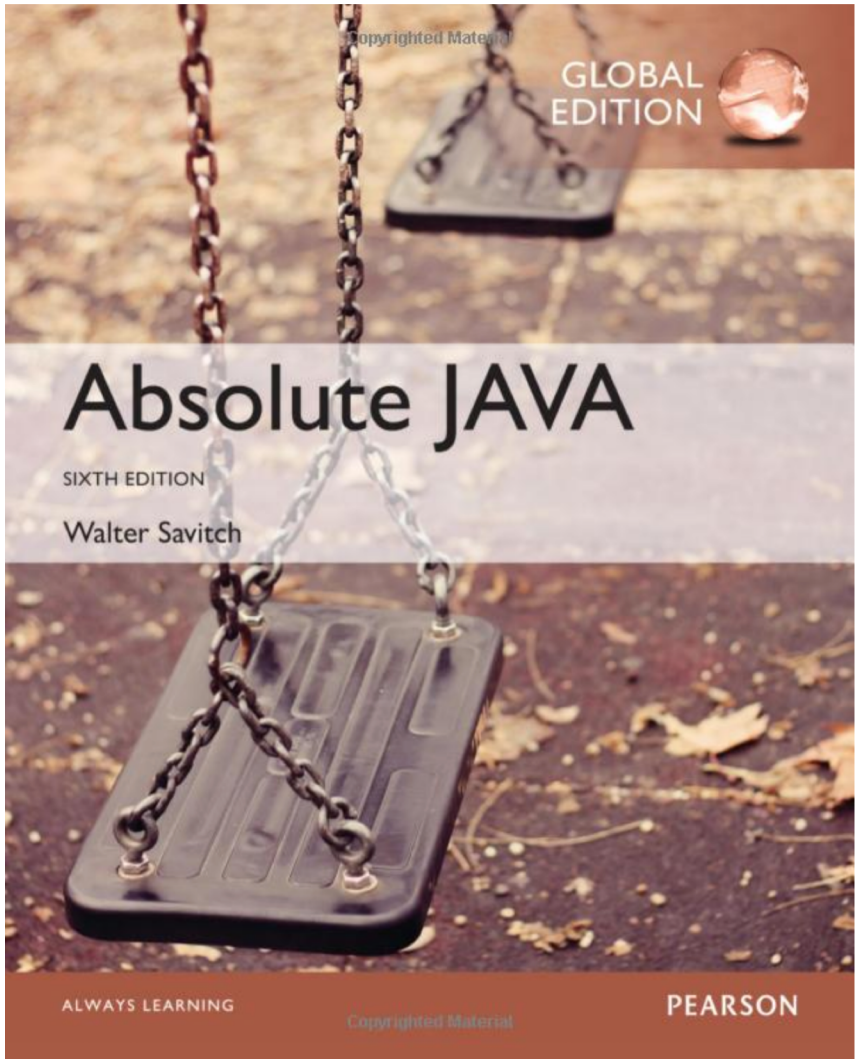
- Best practices in software development:
  - Good programming style
  - Good documentation habits
  - Following specifications

[A computer is] like and Old Testament god, with a lot of rules and no mercy. – Joseph Campbell

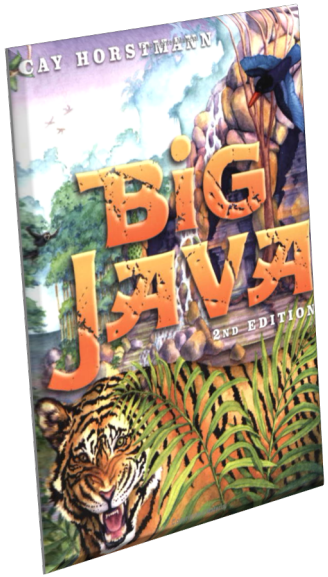
Subject Structure

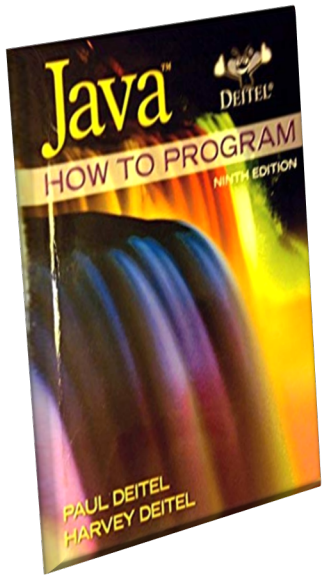
- Twelve 2-hour lectures
  - Short break in each lecture
- Fourteen 1-hour workshop
  - Beginning in week 2
  - Select any one workshop session
- Assessment
  - 40/100 project work
  - 60/100 Final exam
  - You must pass both components to pass the subject
  - All assessments will be individual

Textbook



Absolute Java, 6th Edition (Global), Walter Savitch, Addison Westley.
3rd, 4th, or 5th Edition also fine





Alternative textbooks

Assessment

- 5 assessed exercises
  - Due Friday at 5pm in weeks 3, 5, 7, 9 and 11
  - Only correctness matters
  - Assessed online; submit as often as you like
  - Each exercise is worth 5 points
  - Drop the lowest mark (20 points in total)
- 1 larger project
  - Due after the semester break (around week 10)
  - Worth 15 points
  - Code quality matters, as well as correctness
- You will be asked to critique your classmates' code
  - Worth 5 points
- Final exam worth 60 points

Subject Resources

All available from the Learning Management System (LMS)

http://www.lms.unimelb.edu.au

Student Representatives

- This subject needs 1 or 2 student representatives
- Student reps act as a conduit for anonymous student feedback by email or in time set aside in one lecture
- Reps also report back to department staff on how the class is going (and get a free lunch!)
- Help your classmates and get something to put on your CV
- Email me if you want to volunteer

Academic Misconduct

- Work with friends if you like on **unassessed** labs
- All **assessed** work (project and assessed labs) is to be done by you alone
- You can discuss overall approach to solving problems with peers or others
- **Do not** show your code to peers, in person or electronically, or ask peers for code.
- When in doubt, ask lecturer or demonstrator
- Sophisticated software will be used to identify cheating.
- See http://academichonesty.unimelb.edu.au/

Quick Poll

- We will use the Quick Poll system to check your understanding
- Go to:
  - http://qp.unimelb.edu.au/tchristy
- This is anonymous and NOT assessed
- ...but a good way to check your progress
- ...and you learn more if you tackle these problems

Quick Poll: Transport

How did you get to uni today?

A. Foot

B. Bicycle

C. Car/Motorcycle

D. Train/Tram/Bus

E. Helicopter

Quick Poll – Background

Quick Poll:

How much programming experience do I have?

A. I know Java well

B. I know some Java

C. I know C++ or C, but not Java

D. I know another programming language

E. I don't know any programming language?

This subject

- Is a Java introduction for students with some programming experience
- If you've never programmed, it may be tough
- If you know Java well, you'll be bored
- Only Covers Java language, not frameworks, or GUI, or advanced usage
- If you've never programmed, **ISYS90088 Introduction to Application Development** or **COMP90059 Introduction to Python Programming** would be more appropriate
- Don't suffer in silence: If I'm going too fast or too slow, please let me know

How to succeed in this class

What people think is important:

- Be a "computer person"
- Be good at math
- Know logic

How to succeed in this class

What people think is important:

- ~~Be a "computer person"~~
- ~~Be good at math~~
- ~~Know logic~~

How to succeed in this class

What is important:

- It's like learning a new human language
- Rewiring your brain
- Be patient with yourself
- Be persistent
- **Practice!**

Things to do next:

- Buy Textbook
- Read Chapters 1 and 2
- Download and install a Java IDE
- Attempts first workshop exercise
- Try to compile and run a Java program
- Try to access a student server (nutmeg or dimefox) and copy files

Operating Systems

- The operating system (OS) controls the computer's hardware devices
  - Such as hard disk and DVD drives, mice, keyboards, display screens etc.
  - Saves programs from having to directly control each device
- The OS controls which programs to run at once
- OS shares resources among programs
- Many OSes have a graphical interface and a text-based command line (shell) for control
- Shell is less friendly, but much more powerful

Operating Systems

- Commonly user operating systems:
  - Windows, Max OS X, iOS, Android, GNU/Linux, Solaris, NetBSD, ...
  - All of these, except Windows, are based on Unix
- Most desktop and laptop computers run Windows
- Most mobile, server, mainframe, and supercomputers run some Unix variety
- Becoming comfortable with Unix/Linux will assist you in perusing a career in computing.
  - Good idea to get comfortable with it

OS for this subject

- We will be using Windows in the lab
- You can use any OS to do the exercises and the project
- You can bring your laptop to labs and use the OS you prefer
- **You must use the Unix shell** to submit your work
- In the first workshop, you will learn how to access the Linux servers
- To access the servers from off campus, you will need to use University's VPN

5 mins

Hello, World!

- Our first Java program:

```
// display a friendly greeting

public class Hello
{
      public static void main(String [] args)
      {
            System.out.println("Hello, World!");
      }

}
```

- For now, treat parts you don't understand as boilerplate

- All will be explained....

Java Code

```
// display a friendly greeting

public class Hello
{
      public static void main(String [] args)
      {
            System.out.println("Hello, World!");
      }

}
```

- Java program is made up of one or more classes
- Java class is made up of zero or more methods and instance variables
- Java method is made up of zero or more statements.
- There are a few other things, like comments

Developing Code

- Most people use an Integrated Development Environment (IDE) to create Java code.
  - Popular IDEs include Eclipse and Netbeans
  - Both are free to download and use
  - Use whatever tools you like
    - Even notepad
- IDE's hide some boring details
  - But you still need to understand the details

Files

Every line of Java code must be in some text file

The filename must match the class name, including upper or lower casing, and always ends with

.java

Therefore the following class should be in a file called

Hello.java

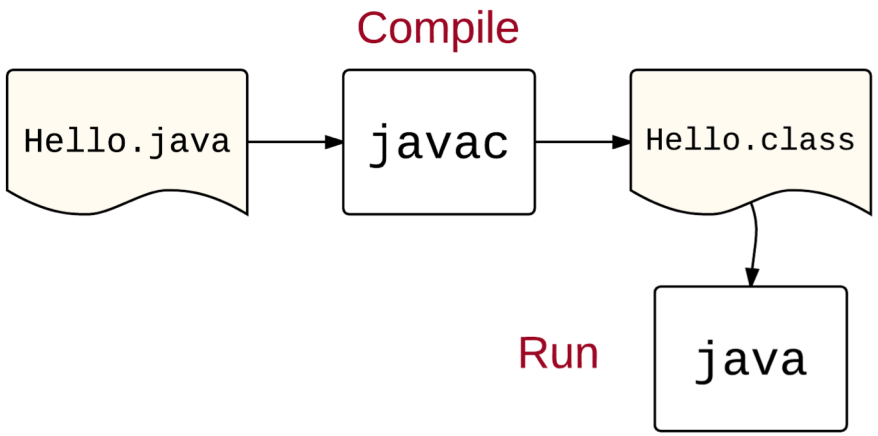
```
public class Hello
{
      // your code goes here
}
```

IDE's try to keep these consistent, but be careful

Compilation

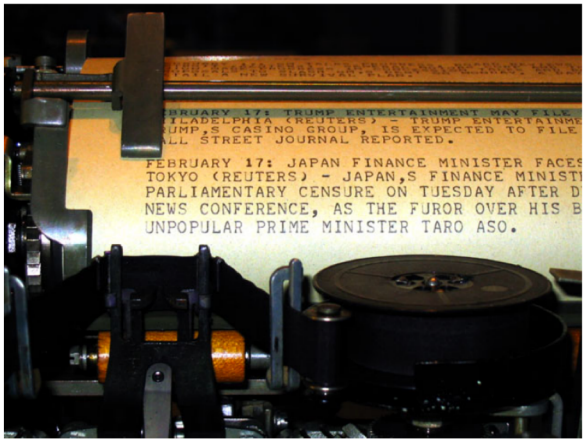
- Before a java program can be run, it must first be compiled with the `javac` command
  - Checks the program obeys the rules of Java
  - Produces a .class file (if compiler passes)

- Once compiled, program is run using the `java` command



Execution

- Java applications run in a console
- Computer consoles originally looked like this



- This subject will focus on the console text input/output.
- IDEs simulate console input/output

Command Line Execution

```
user$ javac Hello.java
user$ java Hello
Hello, World!
user$
```

- **user$** represents my OS prompt; yours will be different
- Compile using **javac <filename.java>**
- If no errors detected, returns to prompt
- Run program with: **java <classname>** (not filename)
- Program output is shown, followed by next OS prompt; keyboard input may be needed

Data

- Two parts of any program: **code** and **data**
- **Code** is the text of the program, what operations the program performs, the verbs of the program
- **Data** is what the code operates on, the nouns of the program
- Each datum (singular of data) has a type
- Three kinds of types: primitive, class, and array
  - We will cover class and array types later

Primitive Types

- Building blocks: all data are built from primitives
- Primitives can't be broken into smaller parts

Type	Bytes	Values
boolean	1	True, false
char	2	All Unicode chars (e.g., 'a', 'b', etc.)
byte	1	-2^7 to 2^7 -1 (-128 to 127)
short	2	-215 to 215 -1(-32768 to 32767)
int	4	-2^{31} to $2^{31} - 1 (\approx \pm 2 \times 10^9)$
long	8	-2^{63} to $2^{63} - 1 (\approx \pm 10^{19})$
float	4	$\approx \pm 3 \times 10^{38}$ (limited precision)
double	8	$\approx \pm 10^{308}$ (limited precision)

Primitive Types

- Which one of these is not a primitive type?
    - bool
    - byte
    - char
    - int
    - double

Variables

- Variables have names and hold data
- Different values at different times
- Variable names begin with a letter, and follow with letters, digits, and underscores (_)
- Java convention is:
  - Begin with lower case letter
  - Follow with lower case, except:
  - Capitalise first letter of each word in phrase
  - Run words together
- E.g. **`height, windowHeight, tallestWindowHeight`**
- Best practice: make them descriptive, but not too long (clear abbreviations OK)

Variable Declaration and Assignment

- Each variable must be declared, specifying its type
  - Specify type first, then variable name, then semicolon
  - E.g., `int count;` or `Boolean done;`
- Variable must be assigned a value before being used
  - Specify variable first, then equal sign (=), then value followed by a semicolon
  - E.g., `count = 1;` or `done = false;`
- You can combine declaration with initial assignment
  - Specify type, variable, equal sign (=), then value and semicolon
  - E.g., `int count = 1;` or `Boolean done = false;`

System.out.println

`System.out.println`

- Prints something out to the console
- The "ln" part means "new line"
  - Next out put will start a new line
- To print something without moving to another line, use

```
System.out.print(something);
```

- Always end each line with a new line. E.g. `System.out.println`, and remember to put whitespace where needed.

Example

println vs print

```
String who = "World";
System.out.print("Hello");
System.out.println(who);
```

Output:

```
HelloWorld
```

This example needed a space between the words

Summary

- Write Java classes in file named ***`classname.java`***
- Variables hold values, can be assigned and reassigned
- Variables must be declared, with their types
- Use **`System.out.print`** or **`System.out.println`** to output values.

THE UNIVERSITY OF
MELBOURNE