



Programming and Software Development

COMP90041

Lecture 1

Introduction

NOTE: Some of the Material in these slides are adopted from the Textbook resources



- Who's teaching and how?
- What is this subject about?
- How much time will this take me?
- How and where do I get help?
- How does assessment work?
- What does academic conduct mean?

Learning Outcomes



Office hours, each **Monday, 10-11am**

- Via ZOOM
- [Email: udaya@unimelb.edu.au](mailto:udaya@unimelb.edu.au)
- Please include **COMP90041** in the subject line when sending an email.



Office hours, each **Thursday, 11am to 12 pm**

- Via ZOOM
- thuan.pham@unimelb.edu.au

Lecturers



- Start in week 2
- via ZOOM
- 6 Lab sheets + consultations

- Zhuohan Xie (zhuohan.xie@unimelb.edu.au)
- Rahul Sharma (sharma.r@unimelb.edu.au)
- Andrew Naughton (andrew.naughton@unimelb.edu.au)
- Zhe Wang (zoe.wang1@unimelb.edu.au)
- Dengke Sha (dengke.sha@unimelb.edu.au))
- Gaunli Liu (guanli.liu1@unimelb.edu.au)

Tutorials



- **Professor,**
- School of Computing and Information Systems, Melbourne School of Engineering, University of Melbourne.
- Web: <https://people.eng.unimelb.edu.au/udaya/>
- **Leader - Quantum Computing Research,**
- Senior Member, [IEEE](#).

- Research Interests:
 - Trust and Privacy in Networks
 - Sequences for communication and security
 - Cryptography
 - Combinatorics
 - Theory of error correcting codes

Who's Udaya

- **Lecturer in Cybersecurity**

- Software Security — building automated security testing techniques (e.g., Fuzzing) to discovery software vulnerabilities

- <https://thuanpv.github.io>

- Twitter: @thuanpv_

- **Past experience**

- Lecturer, Software architect, Research Fellow, Start-up (co-)founders

- My hometown: Hanoi, Vietnam
- Father of two kids :)



Hanoi (Vietnam), The City for Peace

Who's Thuan?



- Object-Oriented (OO) software development
 - Program design, implementation and testing
 - OO concept
 - classes
 - objects
 - encapsulation
 - inheritance
 - polymorphism
 - The Java programming language
 - Problem solving
 - data structures
 - algorithms

Subject Overview



- **Determinism**
- **Small actions have big effects**
- **Utterly logical**
- **Consistent, fair, and unequivocally unbiased**
- **A compiler is the most patient teacher you will ever meet**

Programming is Doing Magic



Week	Lecture	Tutorials	Assignments
1	Introduction		
2	Console I/O	Lab 1: Getting Started	
3	Flow Control	Lab 2: Basics of Java	
4	Classes I	Lab 3: User Input	
5	Classes II	Lab 4: Writing Java Classes	Timed Quiz: Sep 4
N/a	Teaching Break		Assignment 1 Release
6	Arrays and ArrayLists	Lab 5: Arrays	
7	Inheritance and Polymorphism	Assignment Consultation	Assignment 1 due
8	Interfaces and Exceptions	Lab 6: Inheritance	Assignment 2 Release
9	File I/O	Lab 7: Exception Handling	
10	Generics	Assignment Consultation	Assignment 2 due
			Final Project Release
11	Guest Lecture	Project Consultation	
12	Recap & Advanced Topics	Project Consultation	
Final Project due: TBC			

Semester Overview



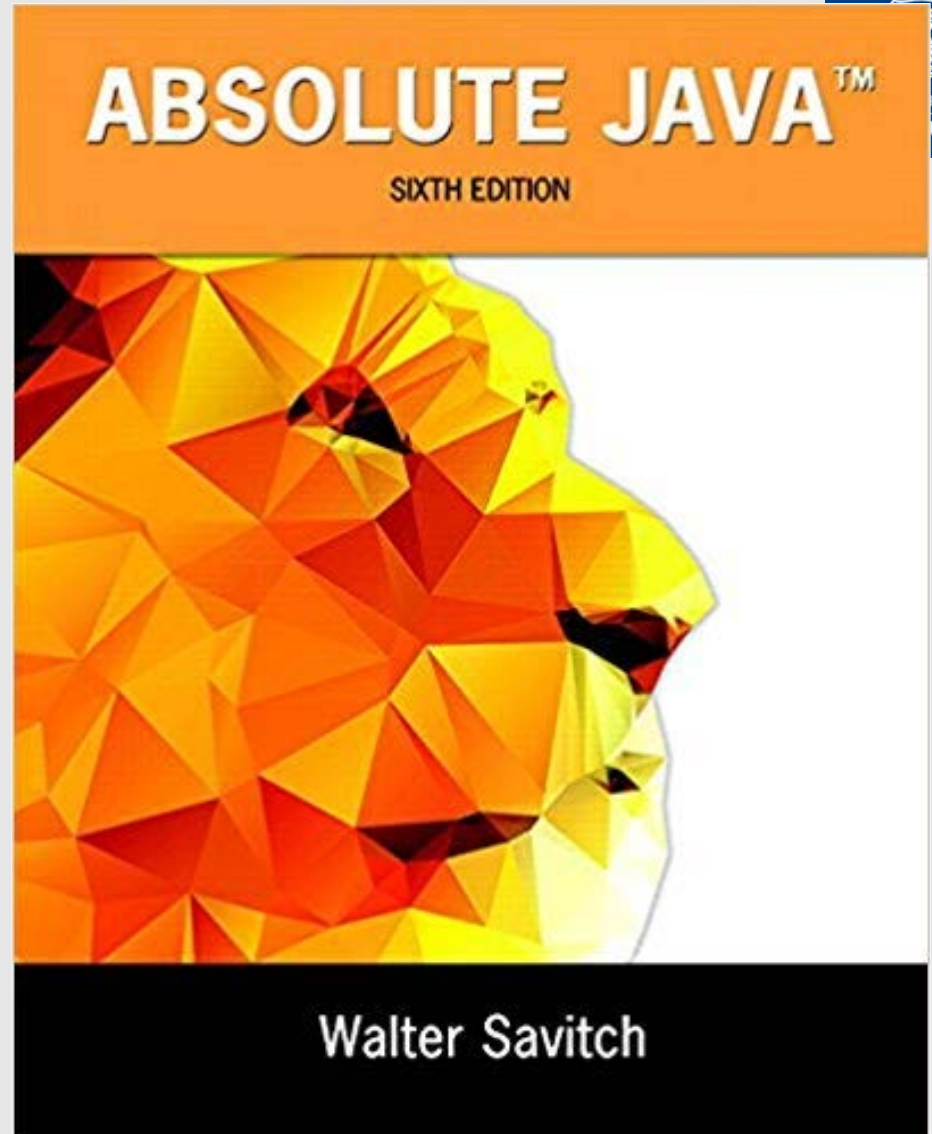
- Each Week we will have 2 Lectures,
- Lecture 1: Monday 5.15 pm.
- Lecture 2: Tuesday 5.15 pm.
- Please come prepared to Lectures by going through any material published for the week (slides, past presentations)
- I will be available for answering any questions after the lecture.
- We will have dedicated forum for discussing lecture contents.

Semester Overview

You can get it from the library

- Physically or

- [Online](#)



Textbook



- Two hours of lectures
- Two or three hours reading the textbook chapter
- One preparation hour for the tutorial
- Two hours attending the tutorial.
- An hour of general review, perhaps in a study group.
- In total, around **eight or nine hours per week** is required, **starting immediately**.
- Make a study timetable for all activities. Then start following it.

Time Commitment



- 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 the SSLC (Student Staff Liaison Committee) meeting 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

Student Representatives



1. **Think**, read, and google it.
2. **Check Canvas** for content and announcements
3. Ask your **fellow students** for help and advice.
4. Consult the **Discussion Board** online. If your question has not been answered already, create an entry.
5. If the question still has not been answered, bring it up in your next **tutorial** session.
6. If that does not help, **email** us (Email Subject must start with COMP90041). We receive a high amount of emails.
7. If nothing else works, come to our office hours.

Where do I get help?



1. One **online Quiz**, 30 minutes duration within a 60 minute time window, taken via Canvas, closed book assessment (worth 10%).
2. **Two programming assignments**, worth 15% each.
3. A **final project** worth 60%.

To pass, a combined mark of 20/40 in the online quiz and programming assignments is required; a mark of 30/60 in the final exam; and an overall mark of at least 50/100 when all components are combined.

Assessment

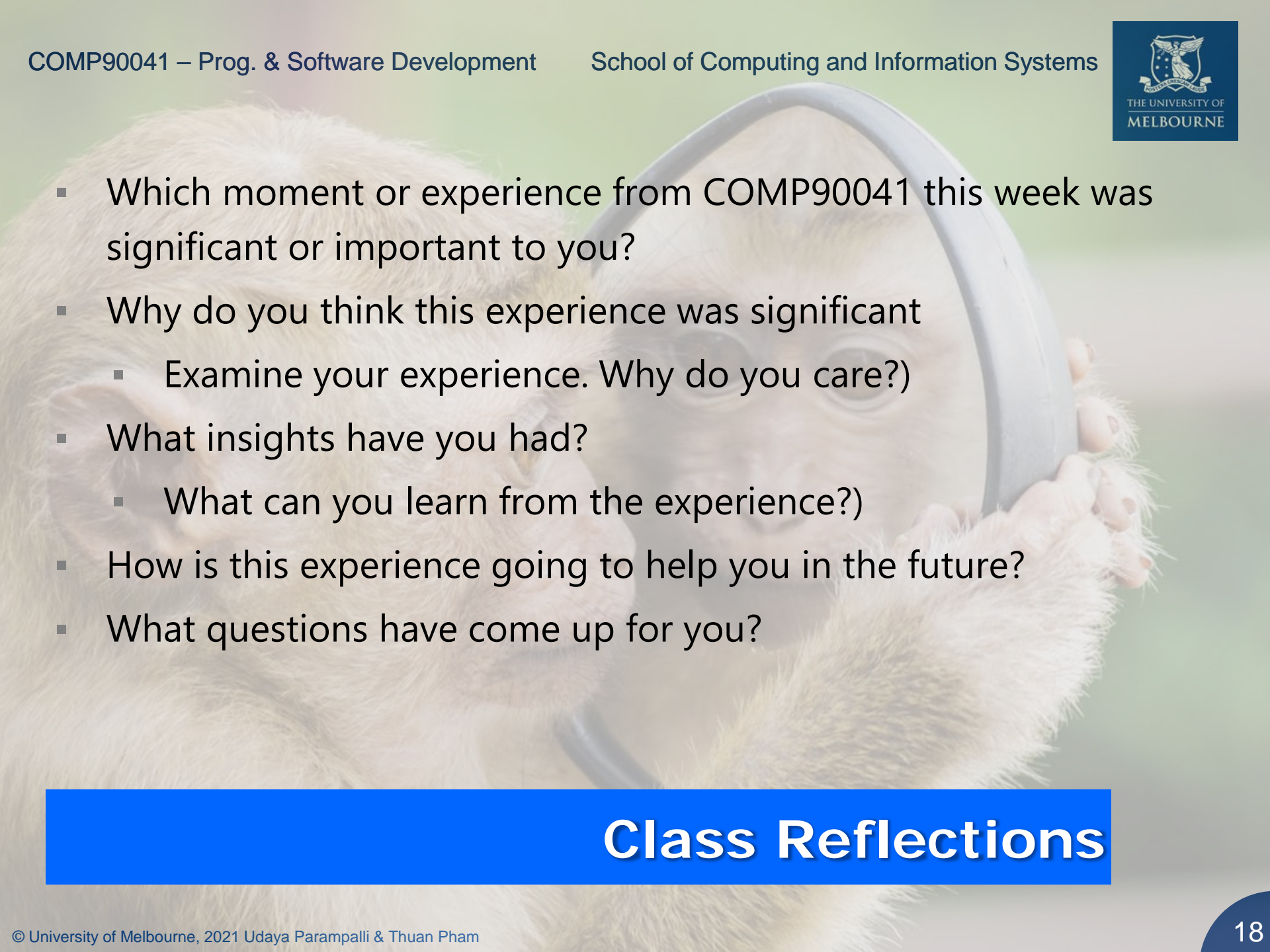


- Go to <https://academicintegrity.unimelb.edu.au/> and inform yourself about the responsibilities you carry in an academic environment
- Enroll in <https://catalog.lms.unimelb.edu.au/browse/communities/student-induction/courses/academic-integrity>
- Be your best and carry these values forward
 - Plagiarism is no joke and not worth it

Academic Integrity

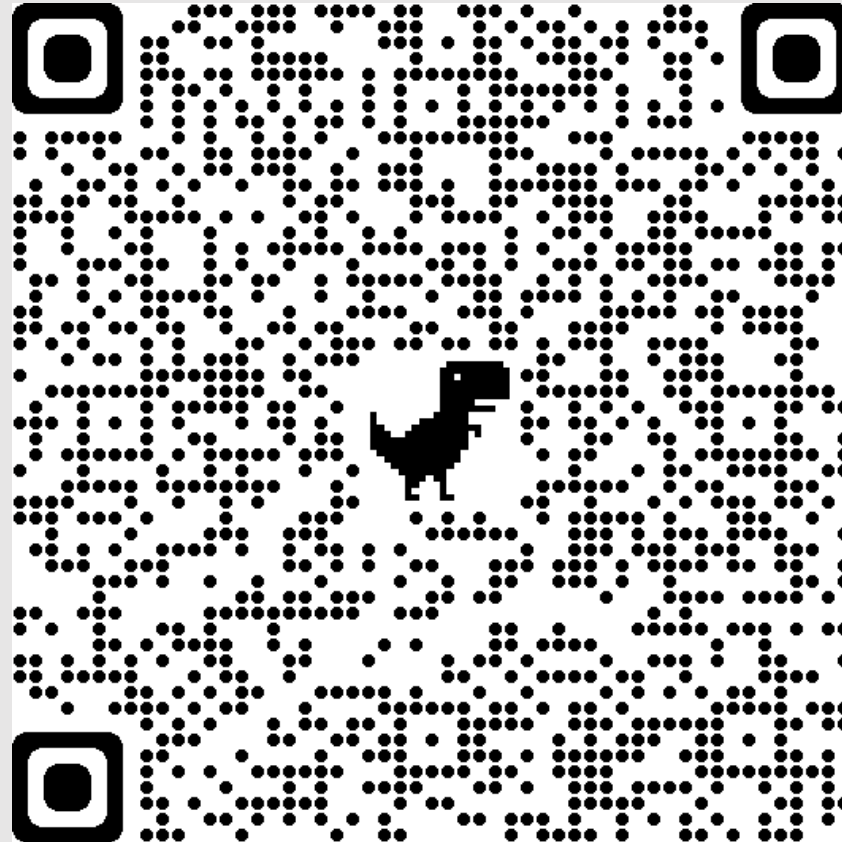
One more thing...



- 
- Which moment or experience from COMP90041 this week was significant or important to you?
 - Why do you think this experience was significant
 - Examine your experience. Why do you care?)
 - What insights have you had?
 - What can you learn from the experience?)
 - How is this experience going to help you in the future?
 - What questions have come up for you?

Class Reflections

Please fill in this microblog.



<http://go.unimelb.edu.au/5o8i>.



One more thing...





- Make sure you have friends in class
- Every time you enter a ZOOM over the next two weeks, turn on your video and mic, and introduce yourself to someone new, while you wait for the start of the class.
 - (we are starting the lecture 15min past the hour)

Make Friends!



- Things were challenging last year. The same situation may continue this year too!
- **Engagement and connectivity** are going to be problems to overcome. Please make a habit of actively contributing to tutorials, by leaving your video on, interacting with the tutors, and building networks with your classmates that extend beyond the formal sessions.
- It won't be as good as being on campus, but it should still be possible for you to **establish and maintain relationships** with a set of other students in the class. Please make the effort!

Make Friends!

■ ...

Take-Aways



Java Programming - Getting Started



- What is the structure of a Java program?
- How to write, compile, and run a Java program?
 - Using simple text editor (e.g., Vim)
 - Using an Integrated Development Environment (IDE) (e.g., Eclipse)
- Variable declaration & assignment

Outline

// 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** (Weeks 4, 5, 7, 8)
- Java class is made up of zero or more **methods** and instance **variables** (Weeks 1, 2)
- Java method is made up of zero or more **statements**
- There are a few other things, like **comments**

First Java Program



System.out.println

- Prints something out to the console
- The “ln” part means “new line”
 - Next output will start a new line
- To print something without moving to another line, use
 - System.out.print

Standard output



- Every line of Java code must be in some text file. The filename must match the class name, including upper and lower casing, and always ends with `.java`
- Therefore the following class should be in a file called `Hello.java`

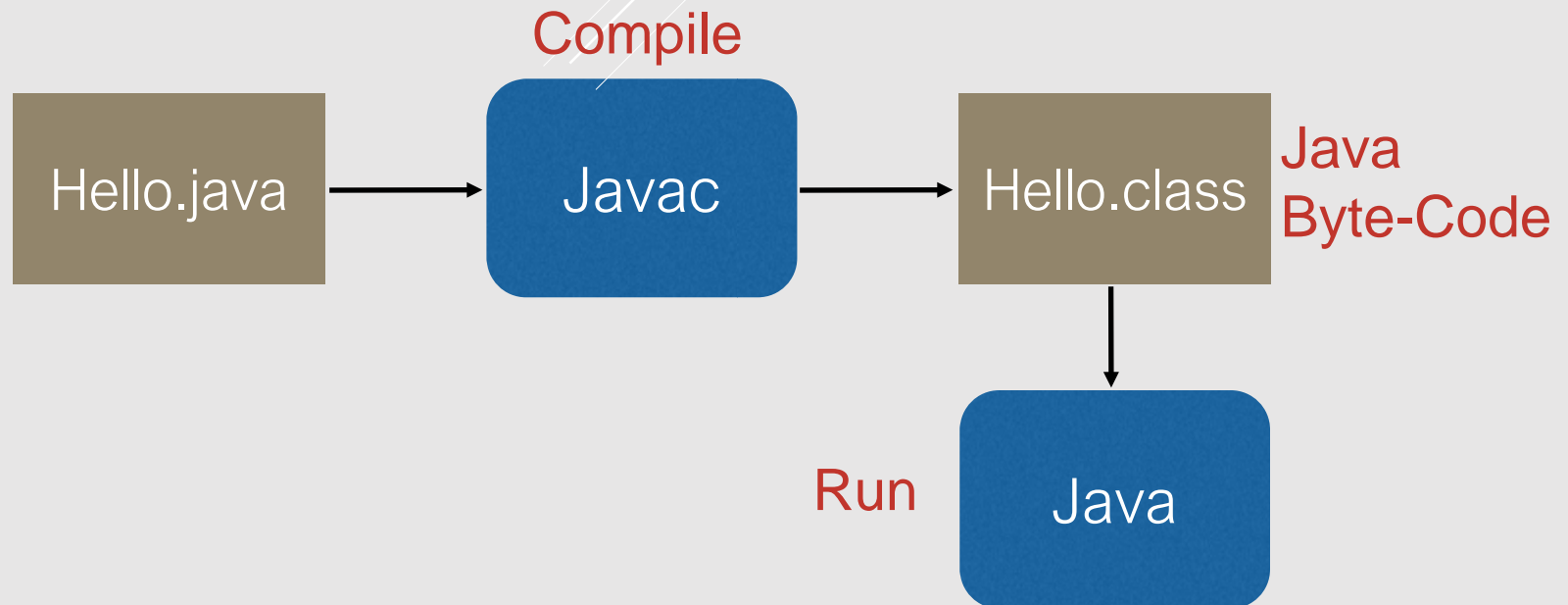
```
public class Hello
{
    // your code goes here
}
```



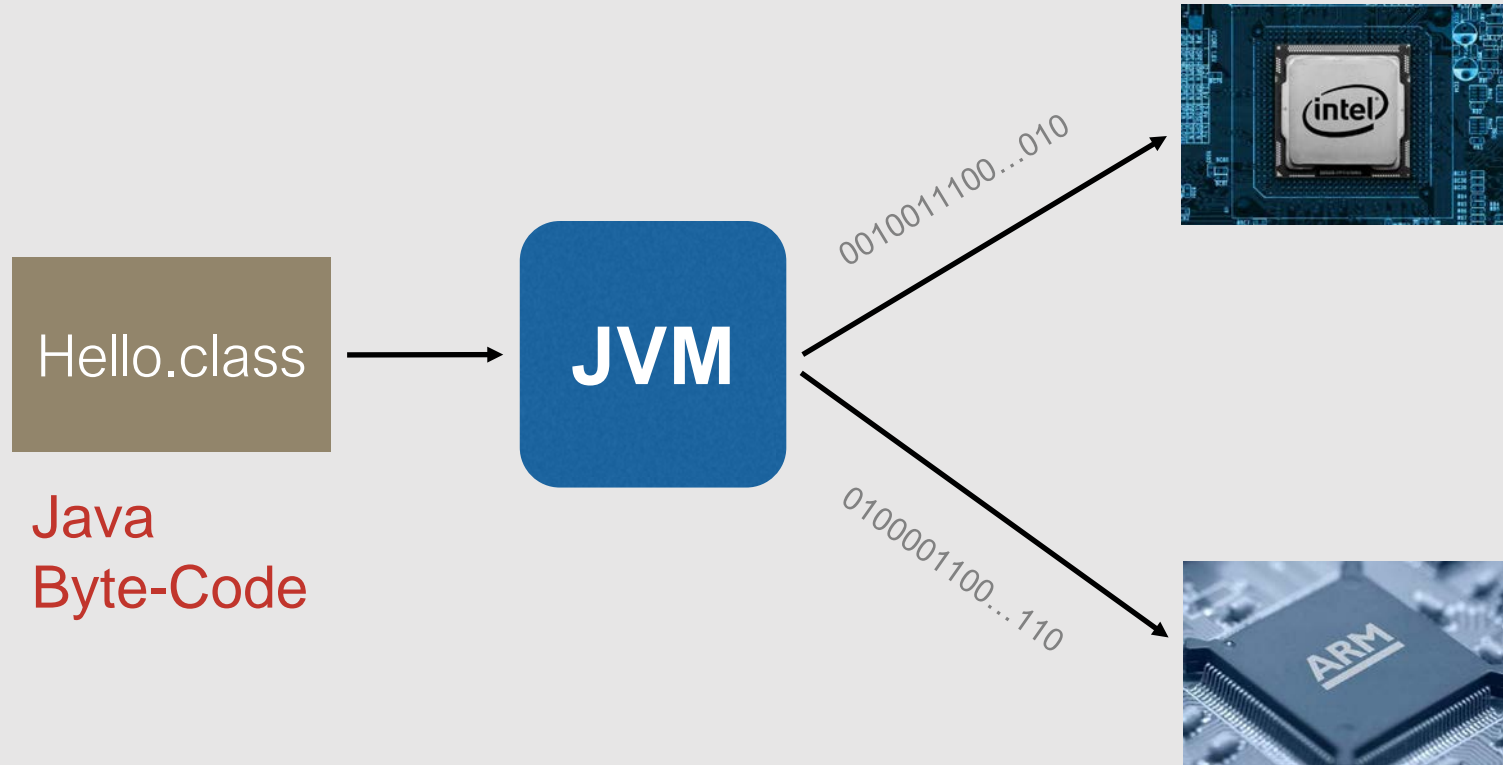
- Most people use an Integrated Development Environment (IDE) to create Java code
 - Popular IDEs include Eclipse and Netbeans
 - Both are free to download and use
 - Use whatever tools you like, even Notepad
- IDEs hide some boring details
 - But you still need to understand the details

Developing Code

- 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



Compilation



Java Virtual Machine (JVM)

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



- This subject will focus on the console text input/output — No graphical user interfaces
- IDEs simulate console input/output

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

Command Line Execution



- What is the structure of a Java program?
- How to write, compile, and a Java program?
 - Using simple text editor (e.g., Vim)
 - Using an Integrated Development Environment (IDE) (e.g., Eclipse)
- Variable declaration & assignment

Outline



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

Data



- 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	-2^{15} to $2^{15} - 1$ (-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



- 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 naming convention for variable names is:
 - Begin with lower case letter
 - Follow with lower case, except
 - Capitalise first letter of each word in phrase
- E.g. height, windowHeight, tallestWindowHeight
- Best practice: make them descriptive, but not too long (clear abbreviations are OK)

Variable names



- Each variable *must be* declared, specifying its type
 - 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 = true;`
- You can combine declaration with initial assignment
 - E.g., `int count = 1;` or `boolean done = true;`

Variable Declarations and Assignment



- Write Java classes in file named `classname.java`
- Compile and run the code using `javac` and `java` commands
- Variables hold values, can be assigned and reassigned
- Variables must be declared, with their types
- Variables must be initialised before being used

Summary