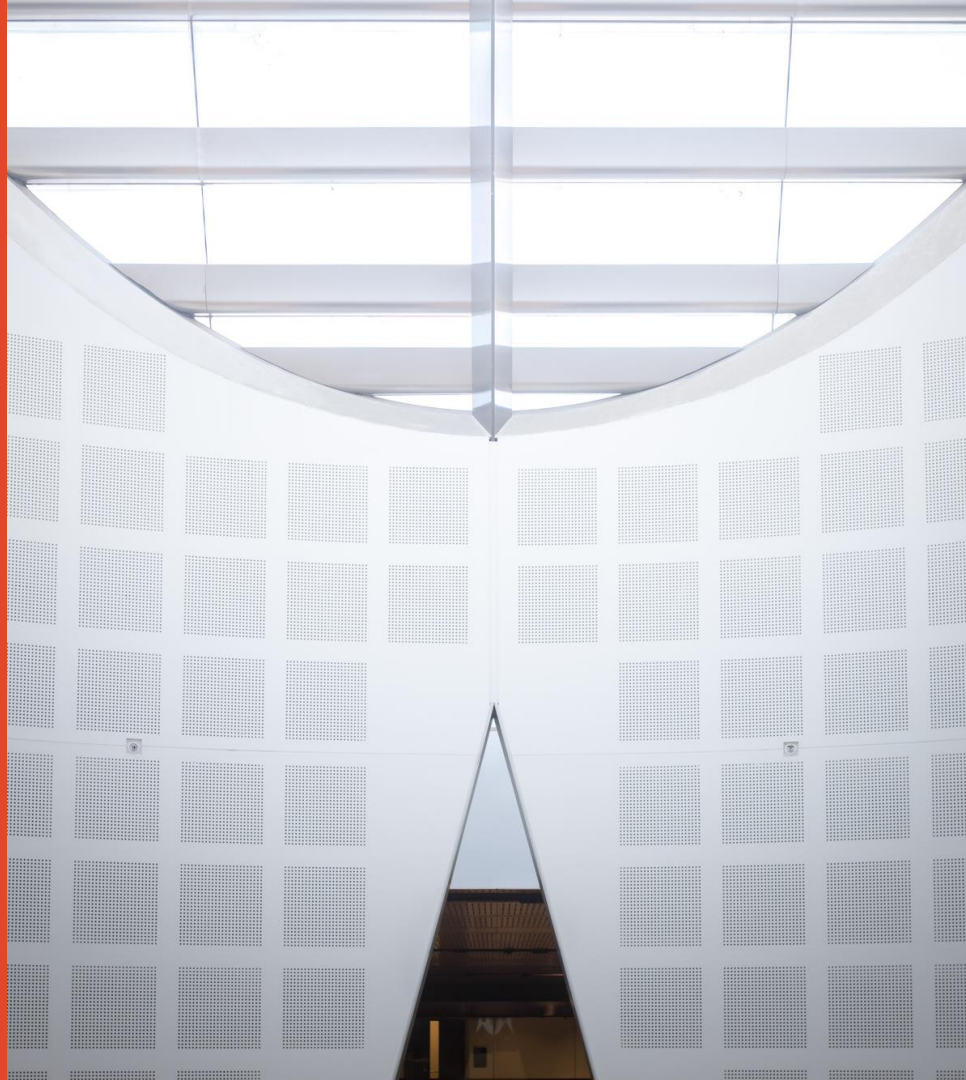
COMP9103: Software Development in Java

W1: Introduction

Presented by
Dr Ali Anaissi
School of Computer Science

THE UNIVERSITY OF SYDNEY



Course Overview

THE UNIVERSITY OF SYDNEY

Curriculum at a glance

Whirlwind tour of:

- **Java programming elements** including data types, variables, control flow primitives, loops, simple I/O
- **OOP fundamentals** including classes, objects, methods …
- **Essential OOP concepts** including inheritance, interfaces, encapsulation, and polymorphism
- **Software development methodology and procedure** such as UML and unit testing

Learning and Teaching

- The *"fundamentals-first & objects-later"* strategy is used:
  1. First the programming building blocks: data types, decisions, iteration, arrays etc (week1-week4)
  2. Later the OO approach: classes and objects, collections, interfaces, polymorphism, inheritance, OO design (from week5)
- The unit is strictly "hands-on":
  - **Learn by doing**
  - Attendance at lab classes is compulsory
  - Skills such as computational problem solving and software development are learned by programming and you are expected to **spend extra hours on PROGRAMMING**

Expected Learning Outcomes

- Programming in Java
  - ✓ Ability to read, understand, and interpret Java code
  - ✓ Ability to understand, modify, and add functionality to Java programs
  - ✓ Ability to convert pseudo-code / algorithms to Java code
  - ✓ Ability to write correct Java programs to manipulate data
- Understanding and programming with the essential concepts of OOP
  - ✓ Class
  - ✓ Object
  - ✓ Encapsulation
  - ✓ Inheritance
  - ✓ Polymorphism
- Problem-solving & Basic Software Development Skills with Java
  - ✓ Identify, define, and analyze problems
  - ✓ Handle problems by designing software solutions with Java
- Testing and debugging
  - ✓ Experience and skills of tracing, testing and debugging Java programs

UNIT ARRANGEMENTS

THE UNIVERSITY OF SYDNEY

Introducing Team

Lecturer

Dr Ali Anaissi

Tutors

Hossein Moeinzadeh

Mohammad Hossein Chinaei

Unit Coordinator

Dr Ali Anaissi

SIT Building J12, Level 2

ali.anaissi@sydney.edu.au

Textbooks and readings

Cay Horstmann,
"Big Java: programming and practice", PJohn Wiley & Sons, Inc.

Rober Sedgewick and Kevin Wayne,
"Introduction to Programming in Java--An Interdisciplinary Approach", Addison Wesley

John Lewis and William Loftus
"JavaSoftware Solution--Foundations of Program Design", Addison Wesley.

Find everything on Canvas

- The web site for this unit is on Canvas
- Use it to access contacts, schedule, readings, slides, etc
- Participate in Q&A with instructors and classmates

https://canvas.sydney.edu.au

**** Check Canvas regularly for all the important information and contents**

ASSESSMENTS

THE UNIVERSITY OF SYDNEY

Assessment

- 10%: Participation
- 20%: Individual assignment project
- 10%: Two quizzes
- 60%: Final exam

To pass the course, you MUST:

*** Score at least 40% in the Final Examination; and**

*** Score at least 50% overall.**

Participation

Objective

Ensure everybody is keeping up.

Output

Code from exercises

Requirements

Submit code at end of each exercise

Marking

10% of overall mark

Individual assignment project

Objective

Developing a Java software application

Activities

Design a class diagram

Implementation and testing

Demonstration

Output

Class diagram

Code

Demonstration

Marking

20% of overall mark

Two Quizzes

Objective

Assess understanding of unit material, ability to trace java code, identify syntax errors and writing java classes and methods

Activities

Answer questions about lecture materials

And Practical excises.

Format

Written examination

Marking

10% of overall mark

Final exam

Objective

Assess understanding of unit material, ability to trace java code, identify syntax errors and writing java classes and methods

Activities

Answer questions about lecture materials

And Practical excises.

Format

Written examination

Must get 40% on exam to pass unit per SIT policy

Marking

60% of overall mark

Lecture plan

- W1: Introductions
- W2: Java Basics & Primitive data types
- W3: Decisions and Iterations
- W4: Arrays
- W5: Introduction to OOP

  Quiz 1
- W6: Define Classes
- W7: Java collections
- W8: Exceptions and File I/O
- W9: Case study

  Quiz 2
- W10: Inheritance
- W11: Abstract classes and Interfaces
- W12: Review

LATENESS AND PLAGIARISM

THE UNIVERSITY OF SYDNEY

Recipe for success

- Attend scheduled classes except for illness, emergency, etc
- Plan 6-9 hours per week for preparation, practice, project, etc
- Participate in classes and forums with respect and humility
- Submit assessments on time

- Let us know if any concerns, e.g., if you are falling behind

Special consideration (University policy)

- If your performance on assessments is affected by illness or misadventure
- Follow proper bureaucratic procedures
  - Have professional practitioner sign special USyd form
  - Submit application for special consideration online, upload scans
  - Note you have only a quite short deadline for applying
  - http://sydney.edu.au/current_students/special_consideration/
- Notify us by email *as soon as anything begins to go wrong*
- There is a similar process if you need special arrangements for religious observance, military service, representative sports, etc

Penalty for lateness

- If you have not been granted special consideration
  - Penalty is 5% of awarded marks per day
  - Maximum 10 days late, then 0 points
- Examples:
  - Work would have scored 60% and is 1 hour late: 57%
  - Work would have scored 70% and is 28 hours late: 63%
- Recommendation: submit early; submit often

Academic integrity (University policy)

"The University of Sydney is unequivocally opposed to, and intolerant of, plagiarism and academic dishonesty.

Academic dishonesty means seeking to obtain or obtaining academic advantage for oneself or for others (including in the assessment or publication of work) by dishonest or unfair means.

Plagiarism means presenting another person's work as one's own work by presenting, copying or reproducing it without appropriate acknowledgement of the source."

http://sydney.edu.au/elearning/student/EI/index.shtml

Academic integrity (University policy)

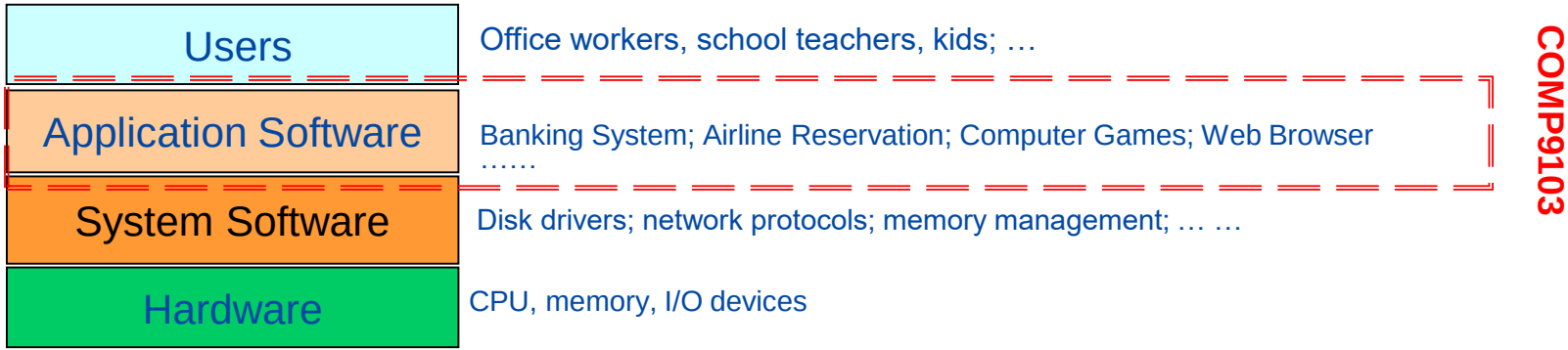
- Submitted work is compared against other work
  - Turnitin for textual tasks (through eLearning)
  - other systems for code
- Penalties for academic dishonesty or plagiarism can be severe
- Complete required self-education AHEM1001

INTRODUCTIONS AND BACKGROUNDS

THE UNIVERSITY OF SYDNEY

Computer Software

A computer system



- **System software**
  - Allocates and manages computer resources
  - Provides a high-level environment (hides details of underlying hardware) for applications
- **Application software**
  - Handles specific requests from users

COMP9103

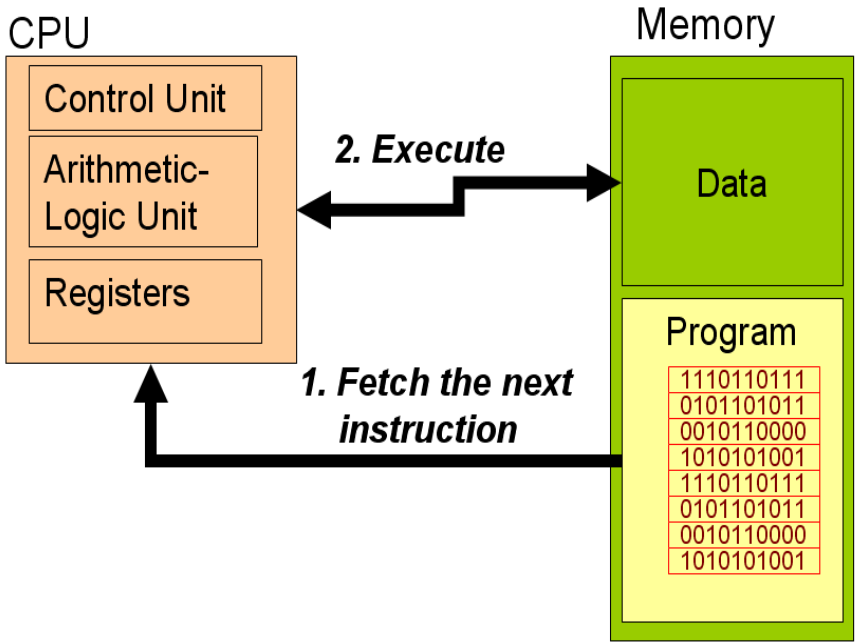
Computer Programs and Programming Languages

- Computer Programs: definite **computational methods** that are expressed in a programming language and implemented in a computer medium.
  - control the computer
  - operate data

- Programming language
  - specifies the words and symbols that we use to write a program
  - employs a set of rules that dictate how the words and symbols can be put together to form valid program statements

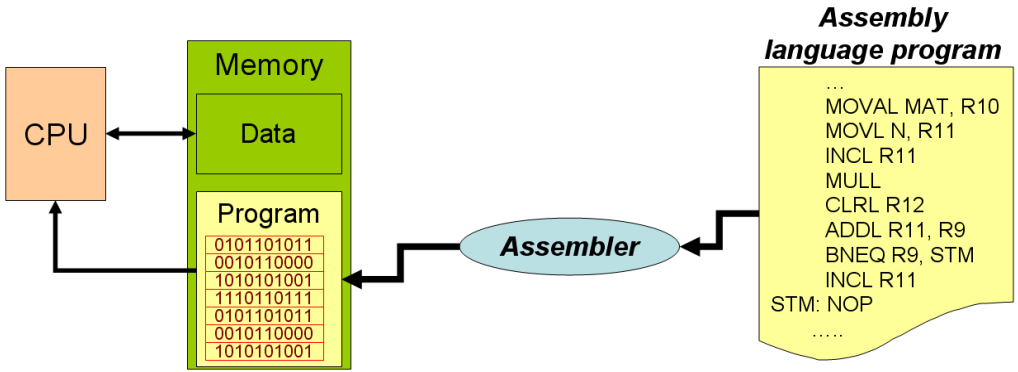
Programming Languages

- First generation language
- Machine languages
  - Instructions are strings of 0s & 1s
  - Problems:
    - No human can remember what 0100011001001010 means
    - As the software became bigger and bigger, it is too costly to write and maintain the machine code



Programming Languages

- Second generation language
  - "Assembly language"
    - Instructions are written in text, using symbolic names
    - The assembly language program is converted into machine language by another program (called an "assembler")

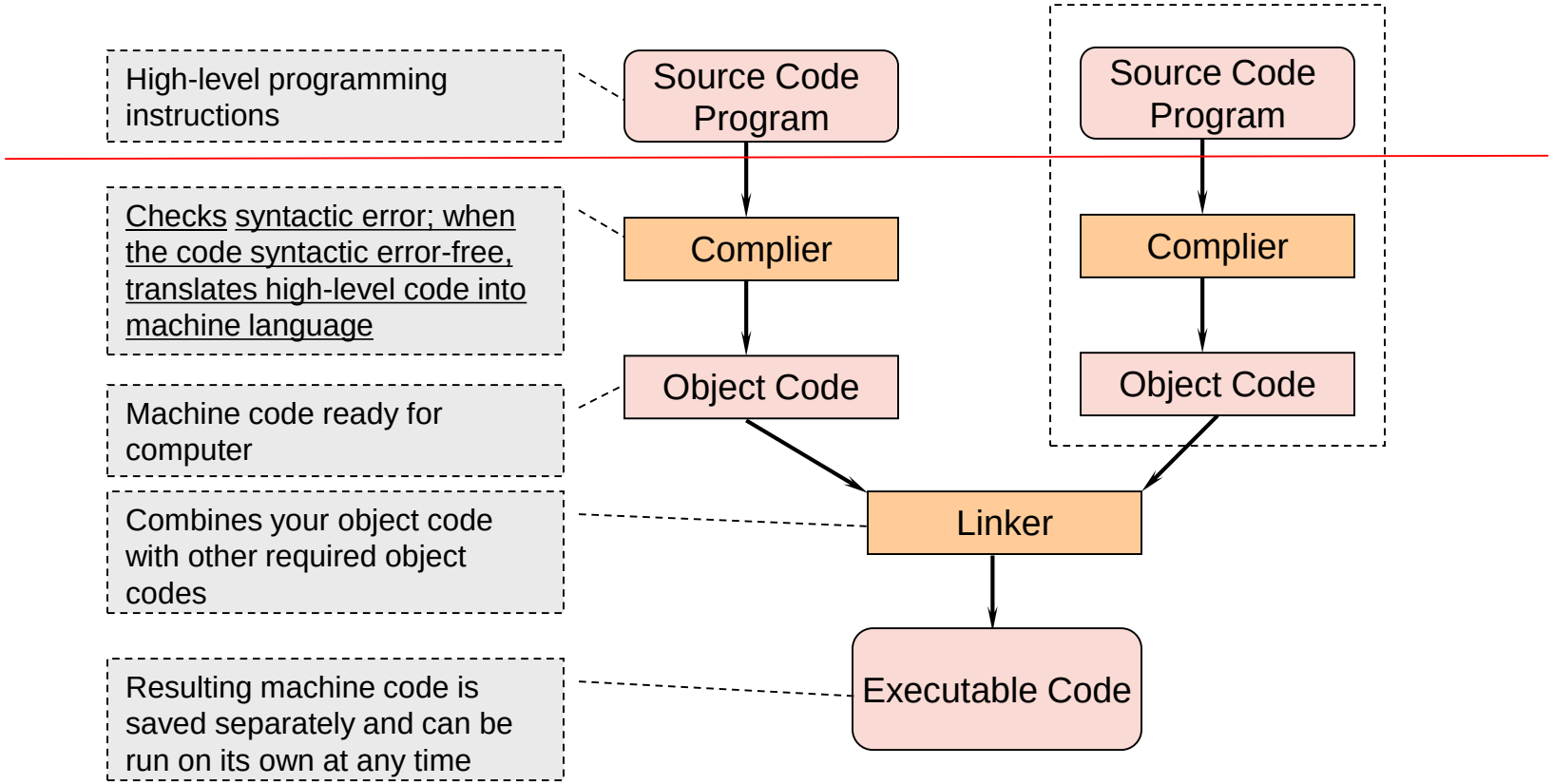


  - Problems:
    - Different hardware requires different machine languages, and this means a different assembly language

Programming Languages

- Third generation language
  - All third-generation languages aim to:
    - provide a framework for computational thinking
    - allow us to 'speak' to the computer at a level that is closer to how humans think than how computers 'think‘
  - Hide the details about the computer and the operating system
  - Hardware Platform independent
    - Same source code can be **compiled** on different platforms
  - “high-level” languages
    - There are many 3rd generation languages: Java, C, C++, Python,…
    - Many programming paradigms: procedural, ***object-oriented***, functional, …

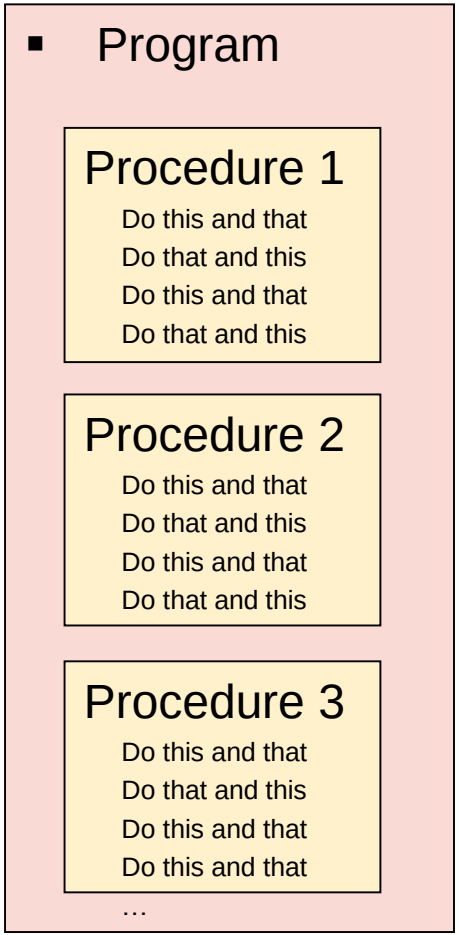
Translation of High-level Language to Machine Code



Programming Paradigms

Programming Paradigms

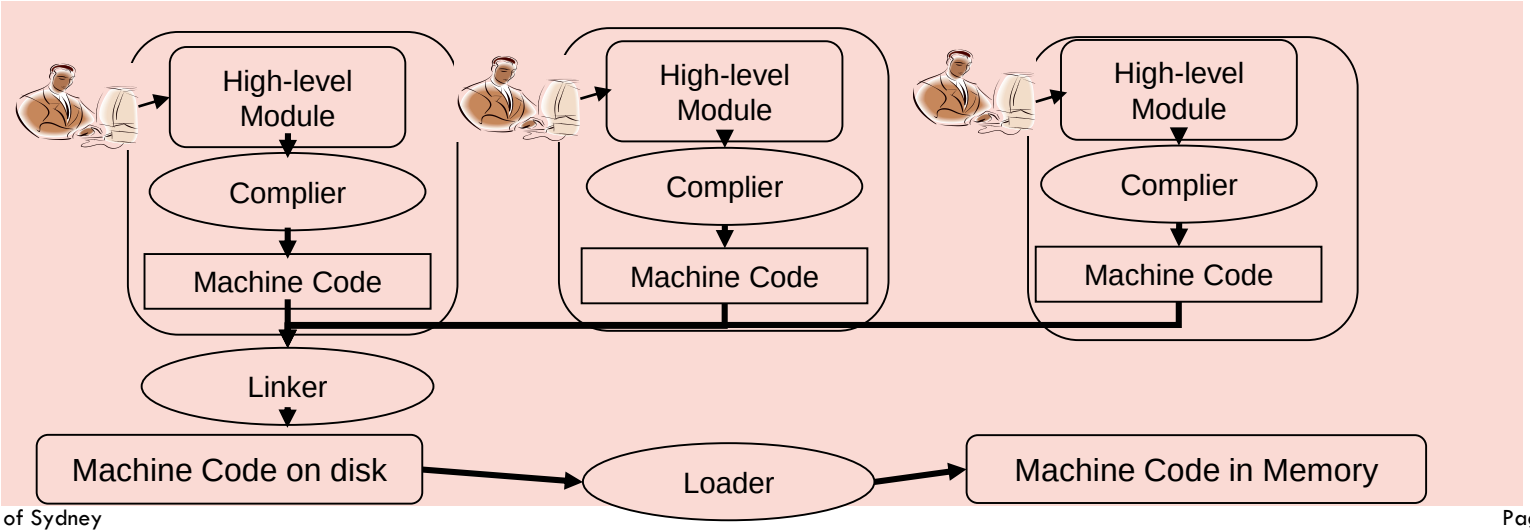
- **Procedural Programming**
  - A program consists of a set of procedures / methods / functions that call each other
  - Procedures/functions/methods/subroutines are small sections of code that perform a list of computations
  - By splitting the tasks into functions, procedural programming allows ***re-use*** of the methods



Programming Paradigms

Modular Programming

- Common functionalities are grouped together into separate *modules*
- Each module is to execute an individual aspect of functionality
- Program modules can be developed and maintained independently (perhaps by different programmers from different organizations in different locations at different times).
- The modules can be linked together to complete a whole task.



Programming Paradigms

- **Object-Oriented Programming (OOP)**
  - One of the most powerful and popular paradigms
  - Includes facilities for modular programming, and (in some sense) procedural programming.

  - **A program** consists of **a collection of cooperating objects** rather than a list of instructions (actions)
  - **An object** combines data structure and its related operations together

Programming Paradigms

- **Major concepts of OOP**
  - **Class & Objects**: a template or prototype from which objects are created; and it includes a collection of variables and methods
  - **Encapsulation**: the implementation details of a class will be hidden from all objects outside of the class
  - **Inheritance**: new class can be derived from a parent class; the derived classes inherit the attributes and behavior of the parent, and can also extend new data structures and methods
  - **Polymorphism**: objects of different types can receive the same massage and respond in different ways

Java

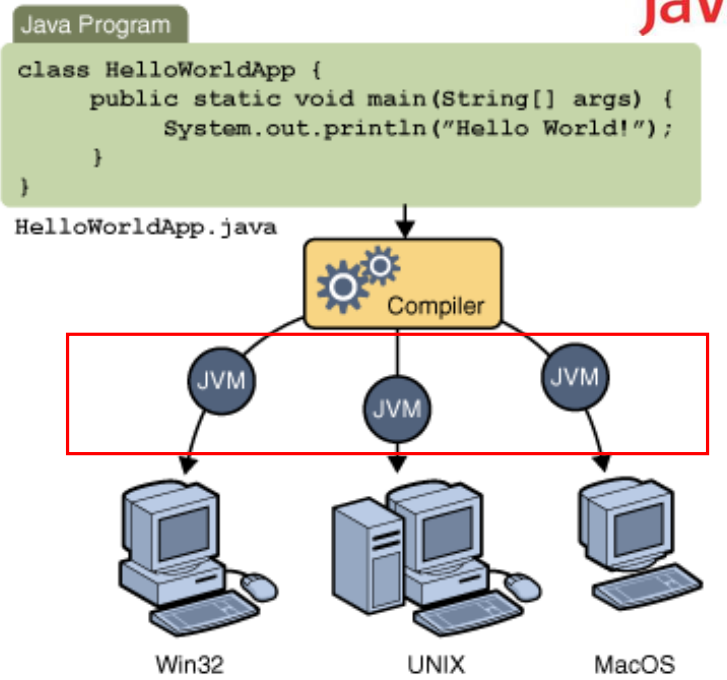
THE UNIVERSITY OF SYDNEY

Why Java?

- Created in 1991, developed by SUN Microsystems in 1995 (which is now a subsidiary of Oracle Corporation), and it's popularity has grown quickly since its development
- Simple (simplified from C++) and elegant
- Safe (no pointers!)
- Rich library (packages/API)
- **Platform-independent** (**"write once, run anywhere"**)

Why Java?

- **Platform-independent** (“**write once, run anywhere**”) Through the **Java Virtual Machine (JVM).**
- the same application can run on multiple platforms.



Your First Java Program

THE UNIVERSITY OF SYDNEY

Your First Java Program—HelloWorld.java

```
/*-------------------------------------------------------------------------------------------
 First java program
  A program to display the message "Hello World!" on standard display
-------------------------------------------------------------------------------------------*/


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

//   end of class HelloWorld
```

comments

comments

- **Comments**
  - Used to explain program, processing steps
  - Can be put any where in your program
  - Will not be compiled

```
// use this when the comment extends to the end of a line;
/*  use these when the comment
    across several lines
*/
```

Java Program Structure

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

- **Classes**
  - Classes are the fundamental building blocks for Java programs
  - A program can be made up of one or more **classes**
  - Every source file contain **at most ONE *public* class**
  - **The name of the file containing the public class MUST match the name of the public class.**
    - e.g. the public class ***HelloWorld*** must be saved in the file named ***HelloWorld***.*java*

Java Program Structure

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

Statement(s)

Each statement ends with a semicolon (;)

Java is case sensitive

- **Methods**
  - A class contains one or more **methods**
  - A method contains **statements** or **instructions** to define how to handle a given task
  - A Java application contains a method called **main() which is starting point of the running program**

Java Program Structure

- **Notes**
- Java is **case sensitive**
  - Be careful about upper- and lower-case letters;

    E.g., System.out.println("Hello World!") ;
- **Layout your program in a readable format (using indentation to reflect your logic & program structure)!**

indentation

```
//*******************************************************
// First java program
//      A program to display the message "Hello World!" on standard display
//*******************************************************

public class HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hello World!");
        }
}
//      end of class HelloWorld
```

Java Translation

Creates a program in Java;

file name ends with **.java**

Editor

e.g.

notepad

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

Java Source code

Java compiler translates Java program into **Byte Code**

Java compiler

Output of compiler: a file name ends with **.class**

Java Byte Code

Byte code is **NOT** a machine language for any particular computer!
It is a language similar to the machine language of most common computers and is easy to translate into machine language for any specific computer.
Same for all platforms and machine-independent!

Java Library files

Java interpreter

Machine language

Machine instructions running on a specific computer

Translates and executes the instructions in Byte Code one after the other

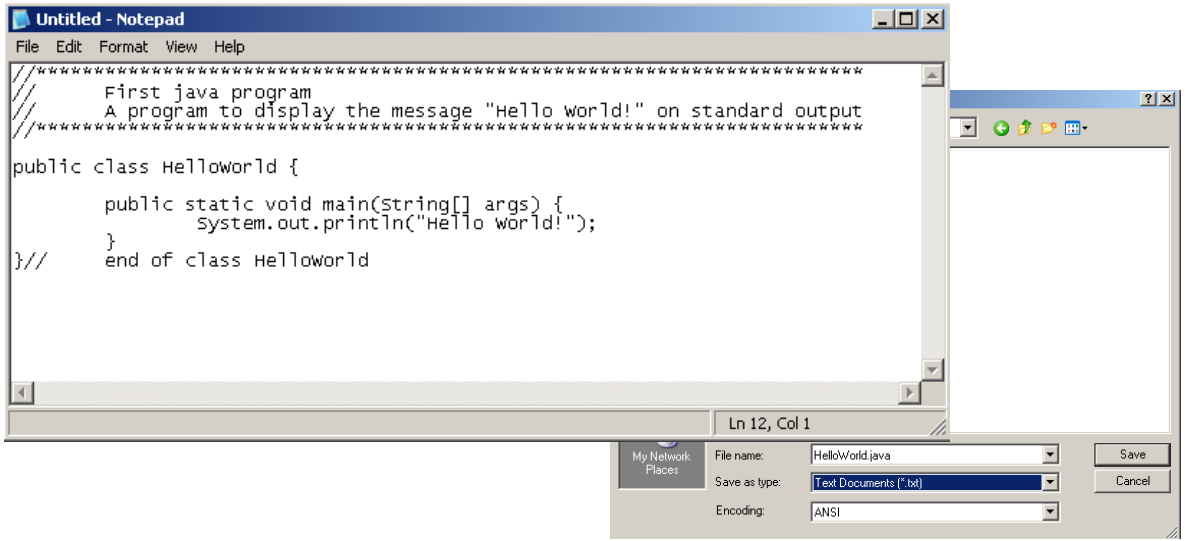
Java Virtual Machine (JVM)

A java program typically consists of different pieces, often written by different people. Each of these pieces is compiled separately, and thus is translated into a different piece of byte code. In order to run your program, all these pieces of byte code need to be linked together. As this linking is usually done automatically, you need not worry about it.

Create, Compile and Run from the Command Line

1. Create

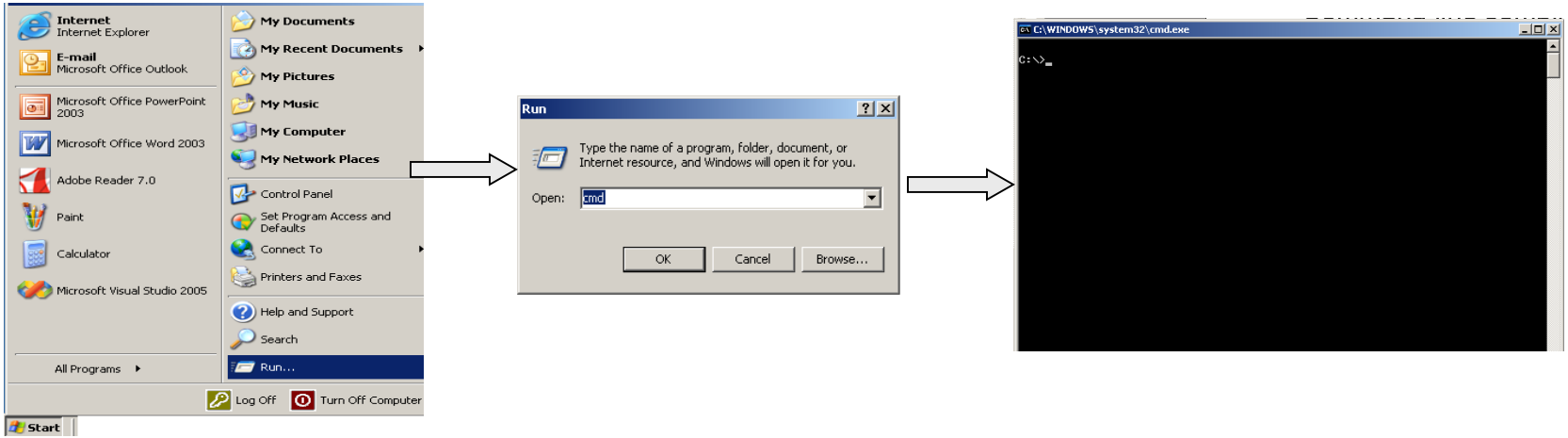
- Use any plain text editor, e.g., Notepad
- Type in the program
- Save to a file named "**HelloWorld.java**" (same as the public class)



Create, Compile and Run from the Command Line

2. Compile

- Run a command line
- Change into right folder
- Compile the program



Compiling and Running Your First Java Program

2. Compile

- Run a command line
- Change into right folder
  - Find the folder where your program resides in, e.g., d:\javaprograms\
  - Type:

    cd javaprograms ➔ ***cd <foldername>***
- Compile the program

Compiling and Running Your First Java Program

2. Compile

- Run a command line
- Change into right folder
- Compile the program
  - Type the following command to **compile** your program:

javac HelloWorld.java → Will generate **HelloWorld.class**

3. Run

- Type the following command to **execute** your program:

java HelloWorld

You will see the result: `Hello World!`

Using an Integrated Development Environment (IDE)

- There are many IDEs, that is, software packages that support the development of Java programs.
- Most are free.
- For example:
  - Sun NetBeans (www.netbeans.com)
  - JCreator (www.jcreator.com)
  - Borland JBuilder (www.borland.com/products/downloads/download_jbuilder.html)
  - BlueJ (www.bluej.org)
  - ***IBM Eclipse (www.eclipse.org) –Recommended. Please refer to the supplement document for how to use Eclipse***

Questions?