UNSW Business School
Information Systems and Technology Management

INFS1609/2609 Fundamentals of Business Programming

Lecture 1
Introduction to Programming

yenni.tim@unsw.edu.au

INFS1609/2609 Fundamentals of Business Programming

Objectives:

1. Introduction to programming
2. Introduction to the course

Main references

- *Textbook: Chapter 1*
- *Reference book: Starting Out with Programming Logic and Design (4th Edition) - Tony Gaddis*
- *Other online references posted on Ed*

Why Programming?

Programming as the new literacy

https://www.youtube.com/watch?v=nKIu9yen5nc

INFS1609/2609 Fundamentals of Business Programming

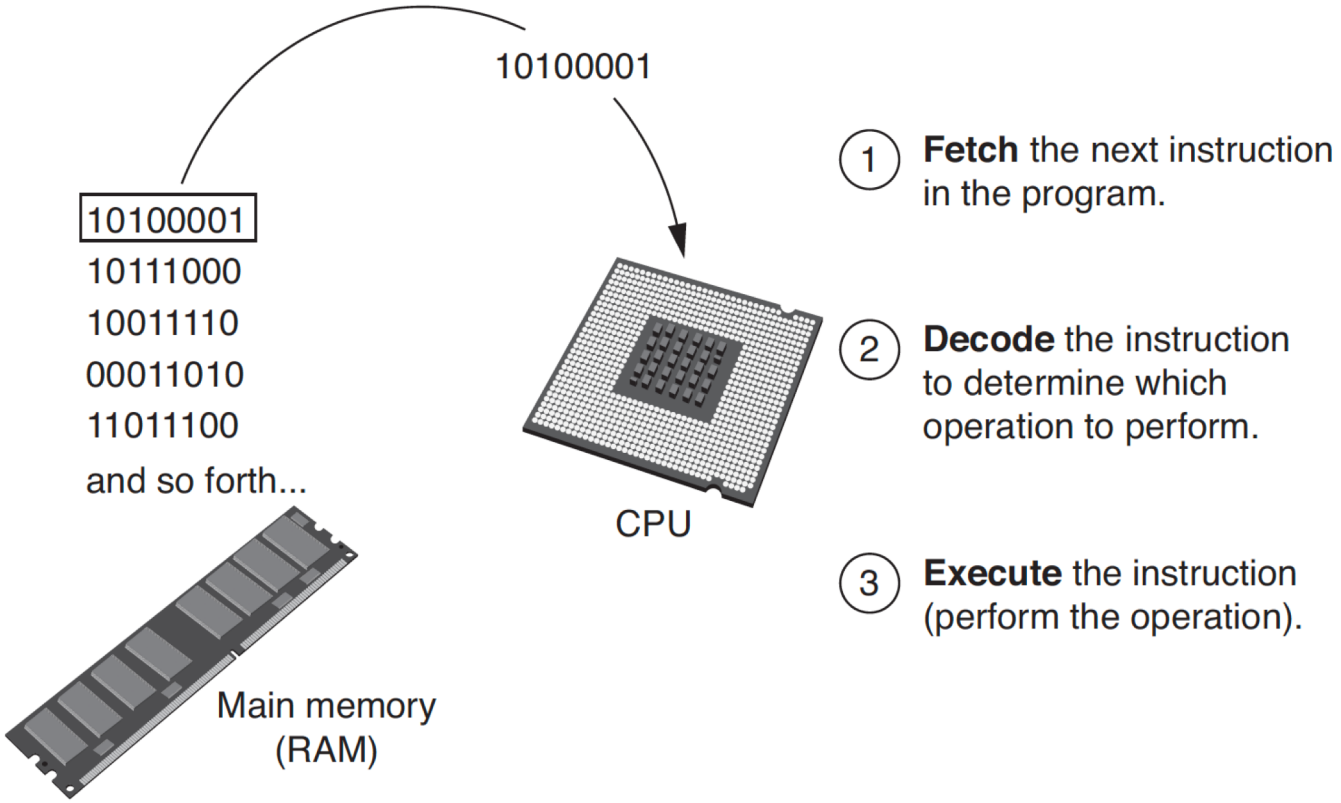
Computers can do a wide variety of things because they can be **programmed**

A **program** is **a set of instructions** that a computer follows to perform a task

To understand programming, we have to first understand how computers work

How computers store data

- All data that is stored in a computer is converted to sequences of 0s and 1s (i.e. binary)

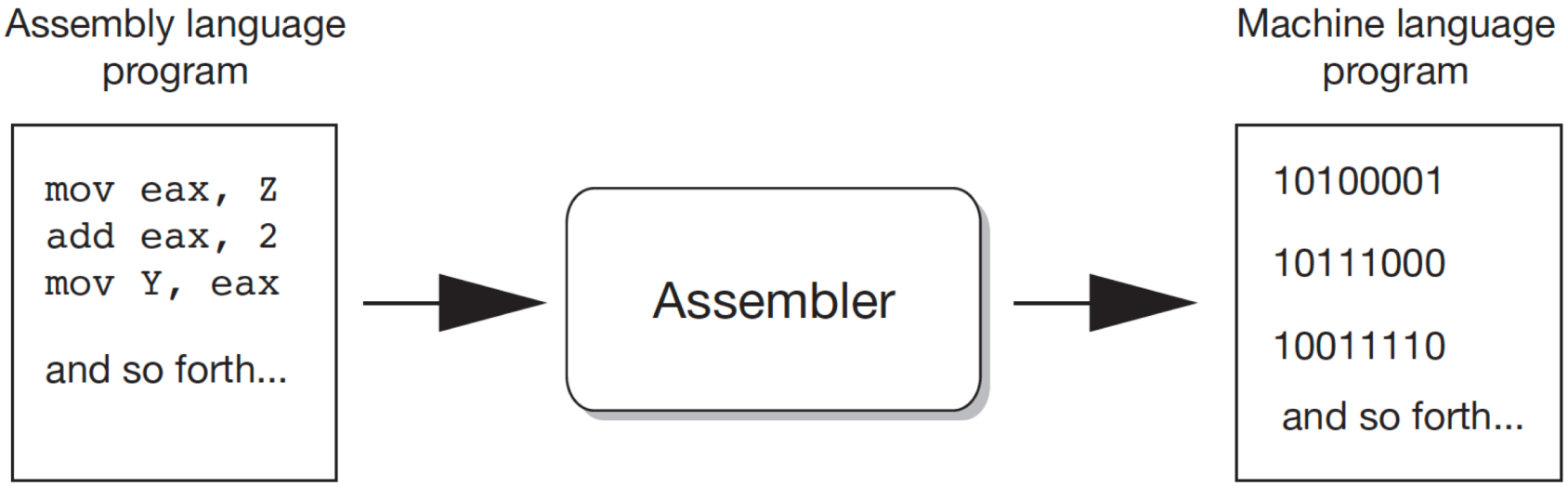


How computers store data

- All data that is stored in a computer is converted to sequences of 0s and 1s (i.e. binary)

A computer's CPU can only understand instructions that are written in machine language. Because people find it very difficult to write entire programs in machine language, other languages have been invented

From Machine language to Assembly language



AL is low-level language, need to write a large number of instructions for even the simplest program

High-Level Languages

- In the 1950s, high-level languages began to appear
- Most high-level languages use words that are easy to understand (for human)
- **Java is one of the high-level languages**
  - Java was created by Sun Microsystems (a company that is now owned by Oracle) in the early 1990s
  - It can be used to develop programs that run on a single computer or over the Internet from a Web server

High-Level Languages

- Each high-level language has its own **set of rules** (keywords, operators, and syntax) that the programmer must learn in order to use the language
- **Syntax: a set of rules that must be strictly followed when writing a program**. The syntax rules dictate how key words, operators, and various punctuation characters must be used in a program

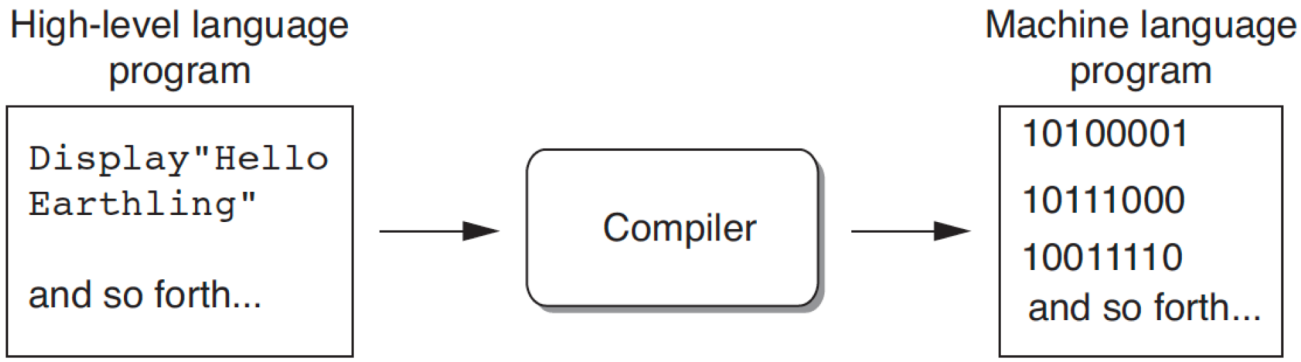
Think and share: How human languages are similar and differ to high-level programming languages?

Compilers and Interpreters

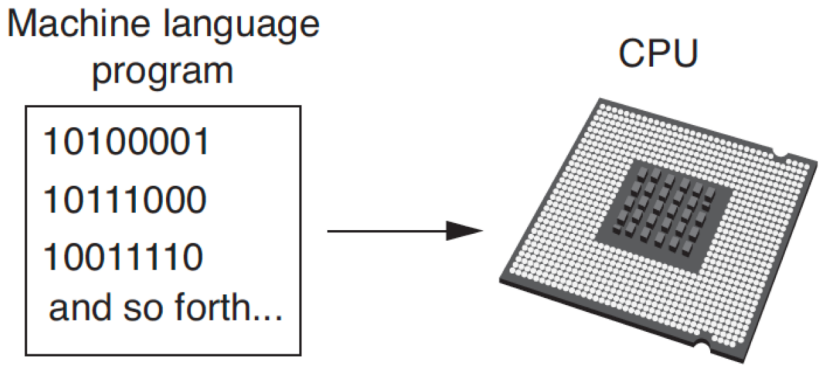
- Because CPU (i.e. the brain of a computer) only understands machine language, programs that are written in a high-level language must be translated into machine language
- A **compiler** is a program that **translates a high-level language program into a separate machine language program**, and that file is then executed

INFS1609/2609 Fundamentals of Business Programming

1. The compiler is used to translate the high-level language program to a machine language program.



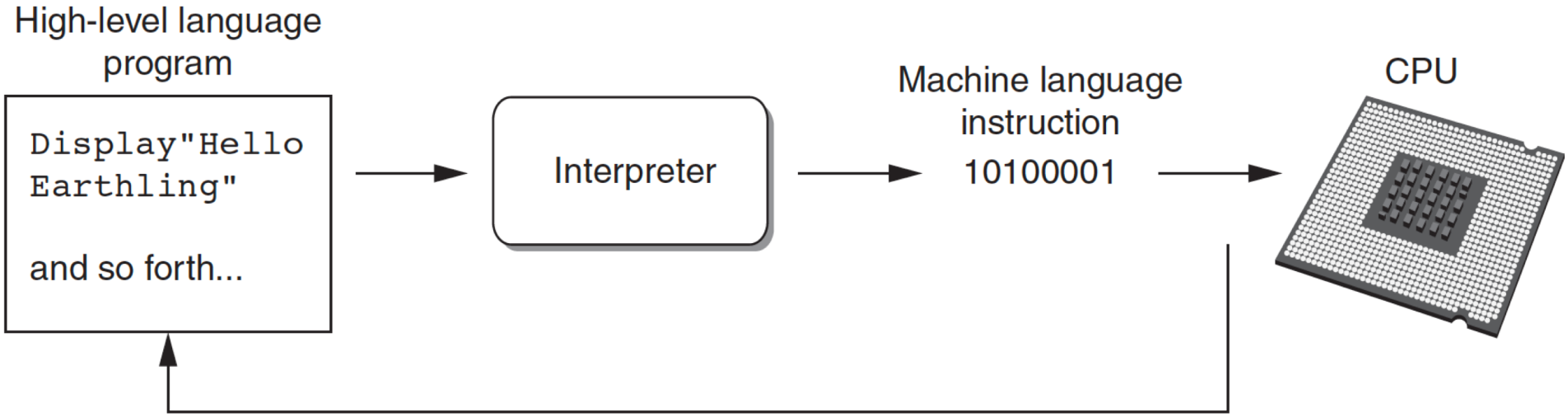
2. The machine language program can be executed at any time, without using the compiler.



Compilers and Interpreters

- An **interpreter** is a program that both **translates and executes** instructions written in a high-level programming language
- Because interpreters combine translation and execution, they typically do not create separate machine language programs

INFS1609/2609 Fundamentals of Business Programming



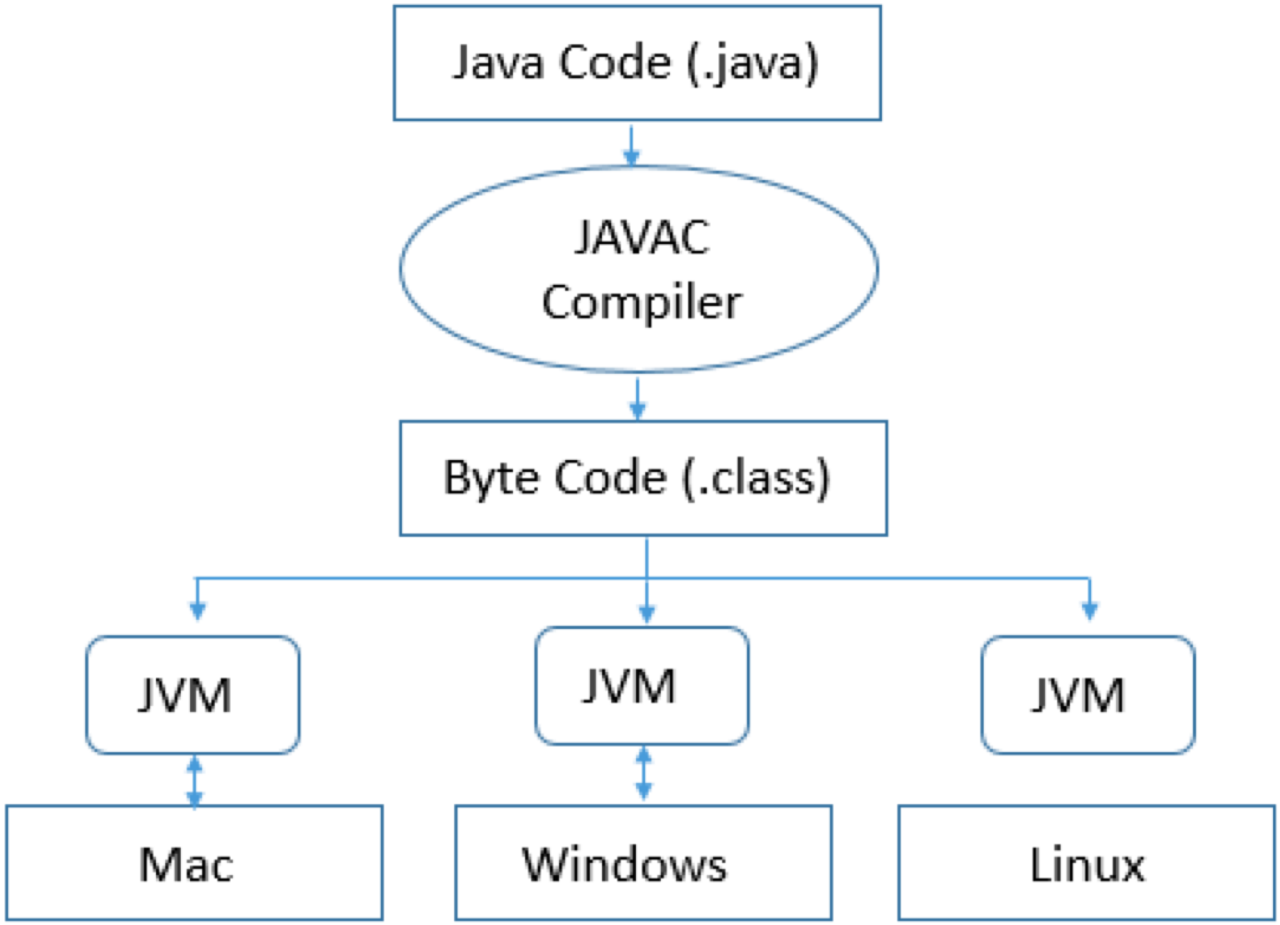
The interpreter translates each high-level instruction to its equivalent machine language instruction and immediately executes it.

This process is repeated for each high-level instruction.

How Java works

- You write a program in Java (Java source code, ***.java* file**)
- A Java compiler (***javac***) is used to compile the source code into a Java bytecode (***.class* file**) (a middle language between Java and the machine language)
- The JVM (**Java Virtual Machine**) executes the class file
- The class file is represented using a hardware- and operating system-independent binary format, it is thus executable in different machines with different OS (as long as JVM is installed)

INFS1609/2609 Fundamentals of Business Programming



How Java works

- Good news: you don't have to do all these manually
- Compiling and executing Java program made easy by **Integrated Development Environments (IDE)**
- Most IDEs combine the following programs into one software package:
  - A text editor that has specialized features for writing statements in a high-level programming language
  - A compiler and/or interpreter
  - Useful tools for testing programs and locating errors

INFS1609/2609 Fundamentals of Business Programming

IDEs we use in this course: **Netbeans**

Installation:
http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html

INFS1609/2609 Fundamentals of Business Programming

Why Java?

- **Widely use** “blue collar” programming language
- Well established **coding conventions** – good for learning
- Scalability and compatibility (platform independent)
- Java is used in **Android apps, server applications in financial services industry, web applications** etc.

INFS1609/2609 Fundamentals of Business Programming

Why this course?

For most of you:

INFS1609: Fundamentals coding capability

INFS2605: Programming independence

INFS3634/3605: Applied Programming

INFS1609/2609 Fundamentals of Business Programming

Lecturer-in-charge: Dr. Yenni Tim

E: yenni.tim@unsw.edu.au

Consultation Time – **Monday, 4 - 6pm** (by appointment)

- **For emails, use only your UNSW email with subject INFS1609/2609**
- **Do not use Moodle messaging function**
- **Feel free to post questions on Ed**

INFS1609/2609 Fundamentals of Business Programming

Course Overview

Assessment Task	Weighting	Length	Due Date
Tutorial Preparation and Participation	20%	See below	Tutorials, Week 2 to Week 13
Assignment	30%	See below	Three sub-parts (Week 4, Week 9 and Week 12)
Final Examination	50%	2 hours	University Exam Period

INFS1609/2609 Fundamentals of Business Programming

Assignment Timeline

Assignment	Release Date	Due Date	Scope
Part 1 (5%)	Start of Week 03	End of Week 06	Week 01 – 05 (Elementary Programming 2)
Part 2 (10%)	Start of Week 06	End of Week 09	Week 06 – 08 (Arrays)
Part 3 (15%)	Start of Week 09	End of Week 13	Week 09 – 12 (OOP)

INFS1609/2609 Fundamentals of Business Programming

Tutorial Schedule and Tutors Contacts

F09A	Blair
F11A	Blair
F13A	Blair
H15A	Ben
H18A	Wilbert
M09A	Kendrick
M11A	Kathy
M13A	Kathy
M15A	Ben
M18A	CT
T13A	Kathy
T15A	Ben
T18A	CT
T18B	Rafiya
W18A	Wilbert

- Blair: blair.wang@unsw.edu.au
- Ben: benjamin.xing@unsw.edu.au
- Wilbert: z5014879@zmail.unsw.edu.au
- Kendrick: kendrickluong@gmail.com
- Kathy: kathy.xu@unsw.edu.au
- CT: z5017884@ad.unsw.edu.au
- Rafiya: rafiya.harun@gmail.com

INFS1609/2609 Fundamentals of Business Programming

PASS class will start in Week 03 (to Week 13)

Wednesday 11.00am – 12.00pm (Quad 1049)

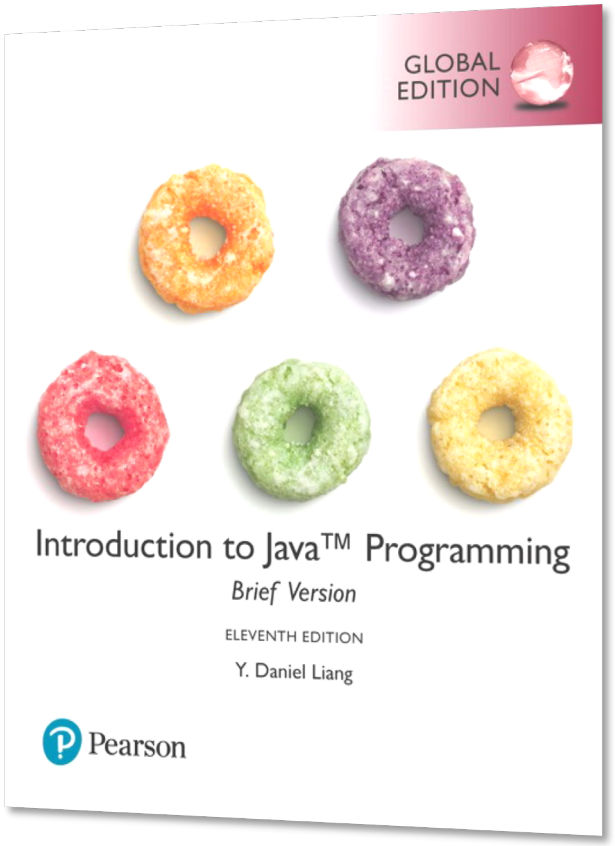
Friday 2.00pm – 3.00pm (Quad 2082)

PASS leader: Mo Kyaw Tun (Mo)

INFS1609/2609 Fundamentals of Business Programming

Required textbook:

Introduction to Java Programming
Brief Version 11e Global Edition
by Liang Y Daniel



INFS1609/2609 Fundamentals of Business Programming

The **Ed platform** (access from Moodle)

- Coding **Challenge** with instant feedback
- Interactive **Lessons** (integrated with coding challenge and short lecture videos)
- Discussion forum (give us your **feedback** anytime)

INFS1609/2609 Fundamentals of Business Programming

Questions?

INFS1609/2609 Fundamentals of Business Programming

Topics for today:

- A problem-solving mindset for programming
- Pseudocode and flowcharting for program design
- A first look at a Java program

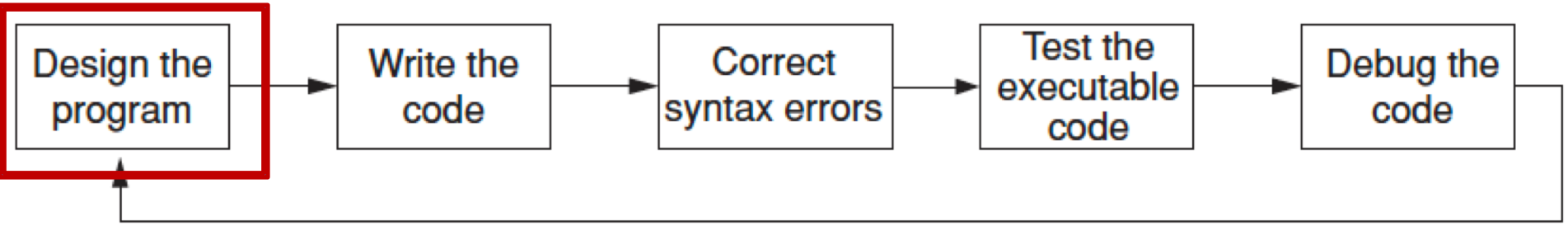
INFS1609/2609 Fundamentals of Business Programming

A problem-solving mindset for programming

https://www.youtube.com/watch?v=UaA1PbyS8BA

A problem-solving mindset for programming

- Programming is not just about “writing code”
- Having a solid program design and underlying logic are key to developing a good program



INFS1609/2609 Fundamentals of Business Programming

Design a program

Step 1: Understanding the tasks that the program needs to perform

- Do you want a program that prints out certain notification? a program that computes a calculation?

Step 2: Determine the steps needed to perform that task

Design a program

Once you understood what the program needs to do, you begin by breaking down the task into **a series of steps**

Think and share:

Suppose your little sister wanted to learn how to boil an egg. Break down the task into a series of steps for her

Design a program

You have just created an **algorithm - a set of well-defined logical steps** that must be taken to perform a task

Suppose you have been asked to write a program to calculate and display the gross pay for an hourly paid employee:

1. Get the number of hours worked.
2. Get the hourly pay rate.
3. Multiply the number of hours worked by the hourly pay rate.
4. Display the result of the calculation that was performed in Step 3.

Design a program

- Now, for a computer to understand and execute the steps, you need to **translate them into code**
- **Pseudocode** and **flowcharts** can be used in this process

**Note: You may be asked to show these steps in your tutorials, assignments and exam

Pseudocode

- Pseudo means “fake” – pseudocode is an **informal language that has no syntax rules**, and is not meant to be compiled or executed
  - Pseudocode allows programmers to focus on drafting logically-correct program instead of worrying about syntax details
  - Once a satisfactory design has been created with pseudocode, this “mock-ups” can be translated into actual code

Pseudocode

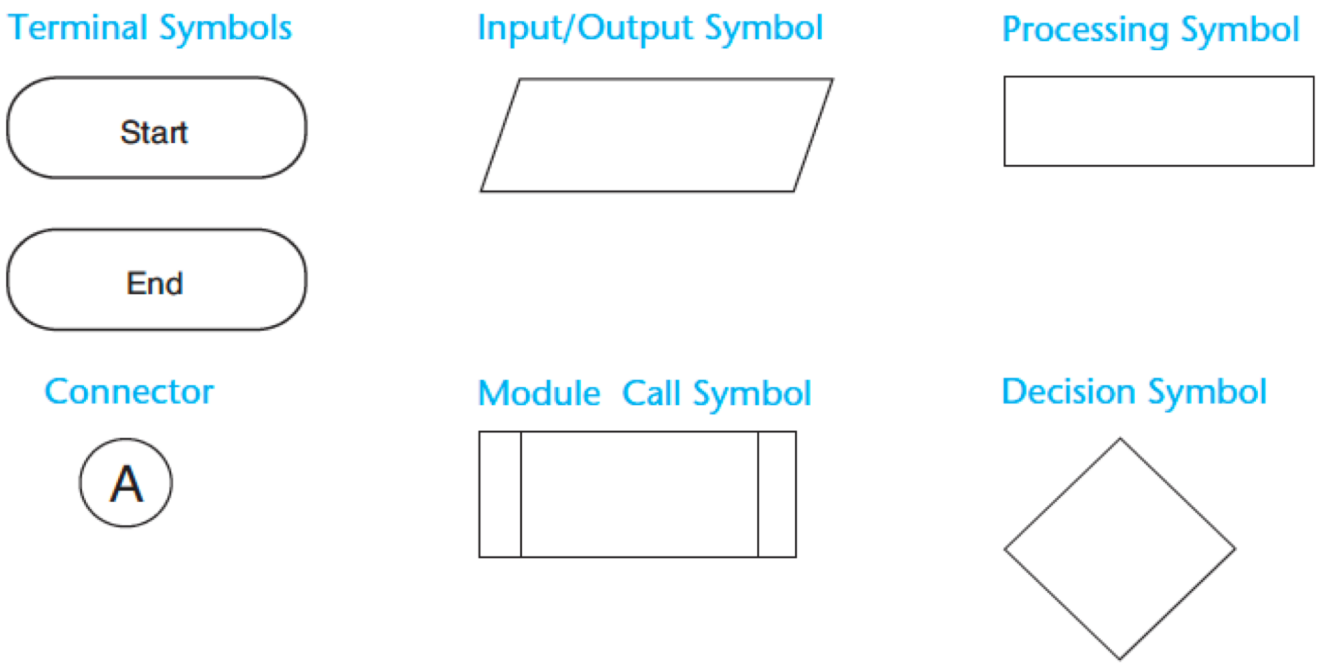
- Here is a sample pseudocode for the example we discussed earlier:

```
Display "Enter the number of hours the employee worked."
Input hours
Display "Enter the employee's hourly pay rate."
Input payRate
Set grossPay = hours * payRate
Display "The employee's gross pay is $", grossPay
```

- Each statement in the pseudocode represents an operation that can be performed in a high-level language

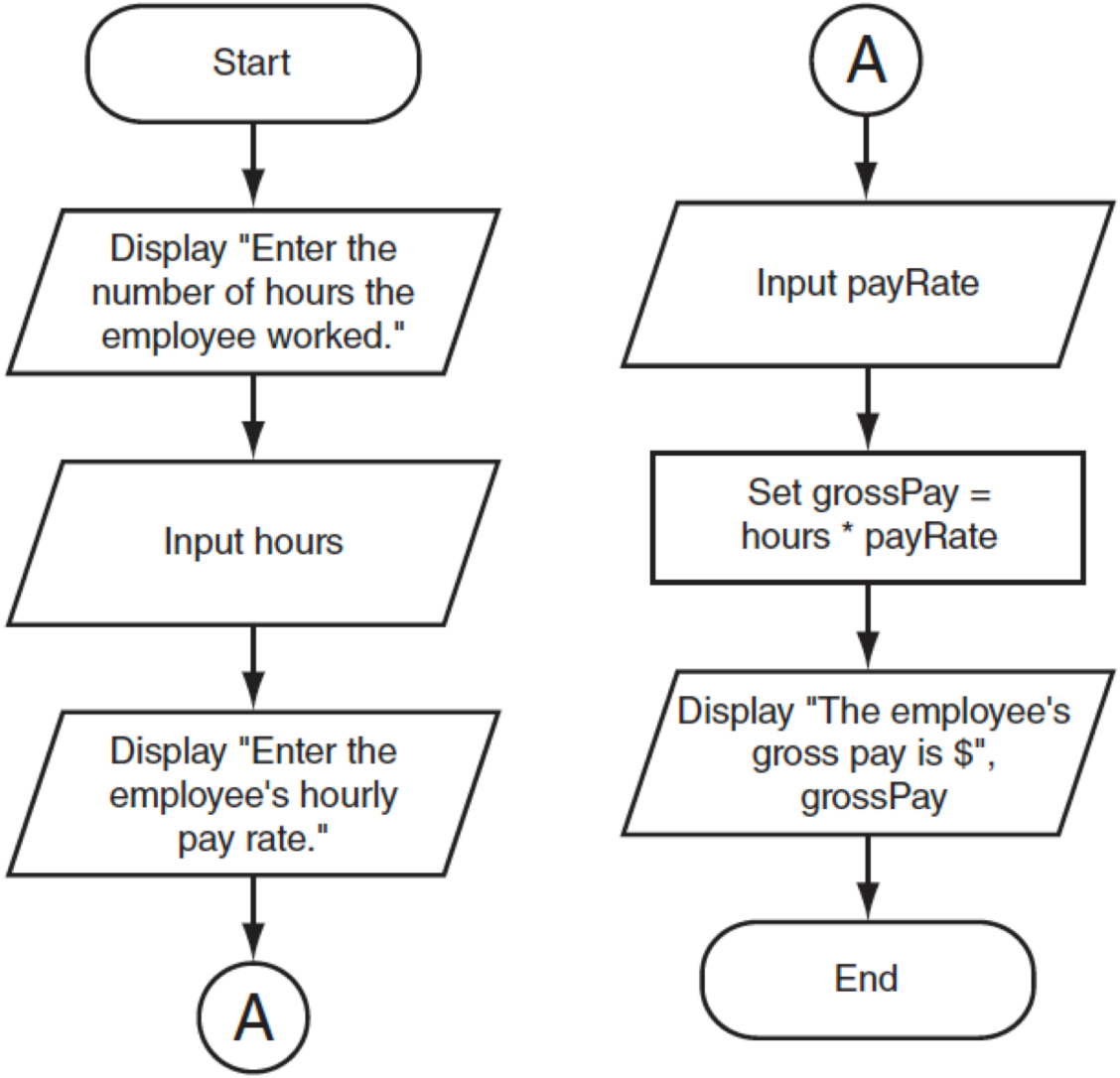
Flowcharts

- Flowcharting is another tool to design programs
- It is used to graphically depicts the steps and decision points that take place in a program
- We are not particular about the notation of a flowchart – these can be used as references:



Flowcharts

- Here is a sample flowchart for the example we discussed earlier:



Flowcharts

- Flowcharting is a good way to break down complicated program, to identify key decision points and map out the overall program flow

In solving your tutorial questions, assignments and exam questions, you should **first develop flowcharts and pseudocode before you type any code**. Syntax errors are easy to fix, but logic errors and poor program design are not

INFS1609/2609 Fundamentals of Business Programming

Now we have learned how a program works and how a program can be designed.

Let's look at a simple Java program!

A simple Java program:

```
public class Welcome {

  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }

}
```

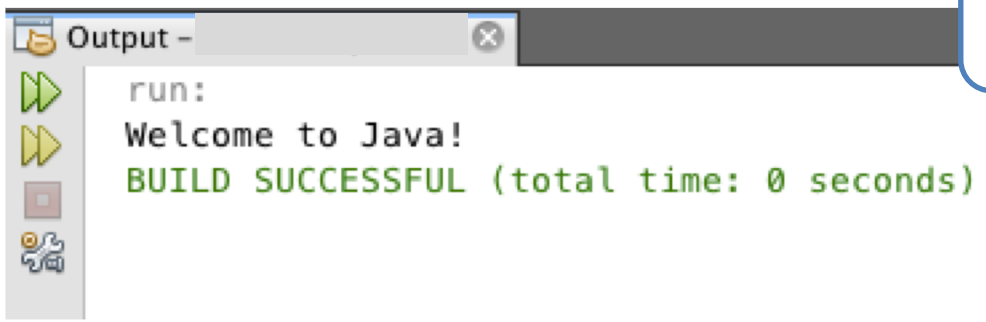
This program prints: Welcome to Java!

```
public class Welcome {

   public static void main(String[] args) {
      System.out.println("Welcome to Java!");
   }

}
```

Print a message to the console



INFS1609/2609 Fundamentals of Business Programming

Anatomy of a Java Program

- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks

Main Method:

- In order to run a class, the class must contain a method named _____
- A program _____ from the main method

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Statement:

- A statement represents an _____

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

- In this example, a statement is _____

A semicolon is a statement

_____

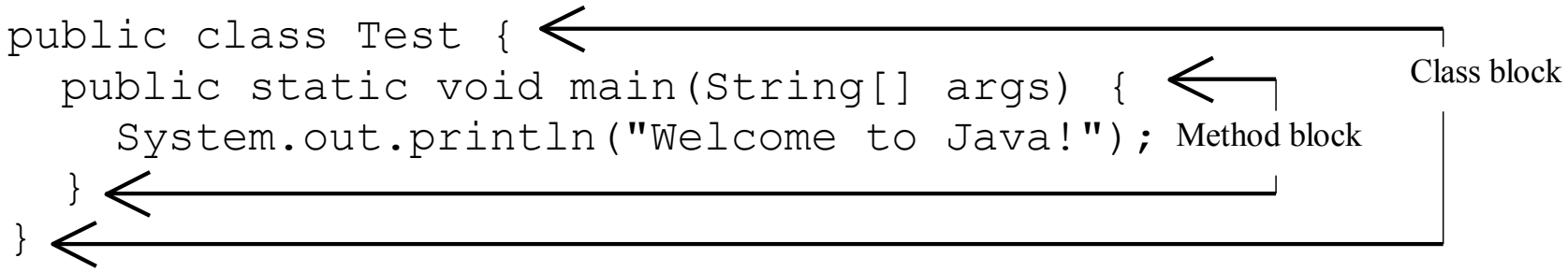
Reserved Words

- Reserved words or keywords are words that have a **specific meaning** to the compiler and cannot be used for other purposes in a program
- E.g., when the compiler sees the keyword _____, it understands that the word after the keyword is _____

```
public class Welcome {
      …
}
```

Blocks

- A pair of _____ in a program forms a block that groups components of a program



INFS1609/2609 Fundamentals of Business Programming

Special Symbols

Character	Name	Description
{}	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

// ...

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

{ … }

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

```
"    "
  ...  "
```

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Reflection

What have you learned today?

What's next?

- **Your first programming workshop**
  - Check out self-learning materials on Ed
  - Be there on time

- Before next lecture
  - Read "Lessons" on Ed

Thank you!