UNSW Business School
Information Systems and Technology Management

INFS1609/2609 Fundamentals of Business Programming

Lecture 2
Thinking Programmatically

yenni.tim@unsw.edu.au

INFS1609/2609 Fundamentals of Business Programming

Topics for this week:

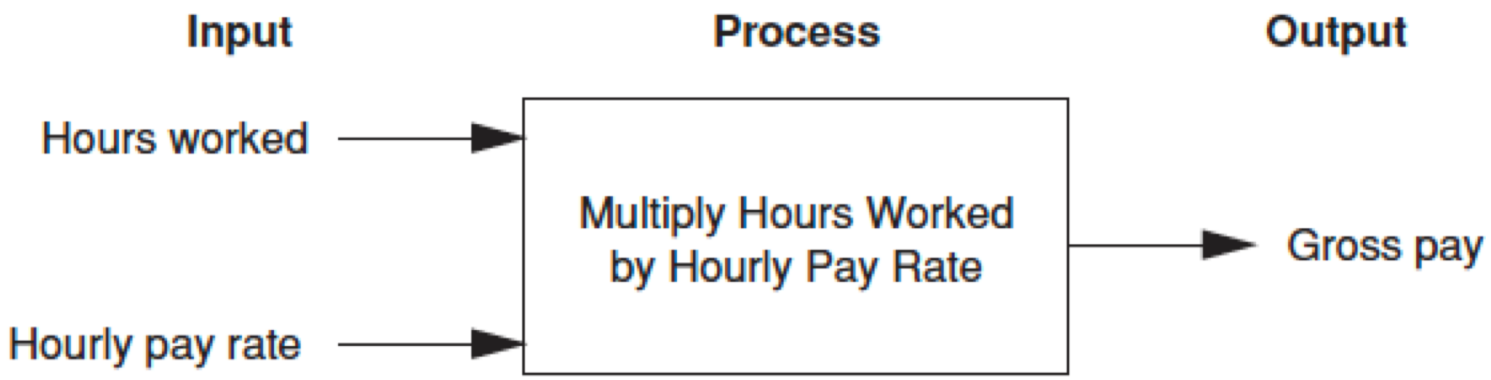
1. The input, processing and output of a program
2. Introduction to the Scanner class
3. Working with variables
4. Tracing a program
5. Introduction to programming errors

Main references

- *Textbook: Chapter 1 and Chapter 2*
- *Reference book: Starting Out with Programming Logic and Design (4th Edition) - Tony Gaddis*
- *Other online references posted on Ed*

The input, processing and output of a program

- A program typically perform 3 steps:
  - **Input** is received
  - Input is **processed**
  - **Output** is produced
- Going back to the example last week:



The input, processing and output of a program

- Using IPO chart to describe the input, processing and output of a program

IPO Chart for the Pay Calculating Program		
Input	Processing	Output
Number of hours worked Hourly pay rate	Multiply the number of hours worked by the hourly pay rate. The result is the gross pay.	Gross pay

The input, processing and output of a program

- Last week we have seen a simple program that displays a message

```
public class Welcome {

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

}
```

The input, processing and output of a program

To change the *output* of the program, we can change the *input*, in this case the text written inside the quotation marks

** In programming, a sequence of characters enclosed in quotation marks is called a *String* (we will discuss more later)

- Now consider writing a program that does some ***Processing*** to the input

The input, processing and output of a program

Write a program that computes the area of a circle

- First, break down the program with pseudocode or flowchart

```
1.  Input circle radius
2.  Compute the area using formula: area =
    radius x radius x pi
3.  Display the area
```

INFS1609/2609 Fundamentals of Business Programming

Write a program that computes the area of a circle

- You have learned that all program must have a `main` method

```
public class ComputeArea{
   public static void main (String[] args){



   }
}
```

Write a program that computes the area of a circle

- You now need to translate this pseudocode to Java code

```
public class ComputeArea{
    public static void main (String[] args){
        // Input circle radius

        // Compute the area using formula:
        area = radius x radius x pi

        // Display the area
    }
}
```

Write a program that computes the area of a circle

- The program needs to first **read an input** (i.e. read the `radius` entered by the user from the keyboard)
- This raises two issues:
  - How to read user input?
  - Where to store the input?

Storing user input – working with variables

- Quite often a program needs to store data in the computer's memory so it can perform operations on that data
- Programs use **variables** to store data in memory
- Think of **variable as a value stored in a box in the computer's memory**
- Variable is **represented by a name**

Storing user input – working with variables

```
public static void main (String[] args){
        // Input circle radius

        // Compute the area using formula:
        area = radius x radius x pi

        // Display the area
    }
```

How many variables we need here and what are they?

Storing user input – working with variables

- We need to specify the **data type** of a variable to let the compiler know what kind of data is stored in a variable
- For example, we may have two variables, `radius` and `area`, but compiler doesn't know if they are numbers, characters, strings etc. until we **declare** them
- Java provides different data types which we can use to declare variables

Storing user input – working with variables

- We will learn more about data types later
- For this example, both `radius` and `area` are real numbers (i.e. numbers with decimal point) – these are represented as "floating point" in computer
  - In Java, you can use the keyword `double` to declare a floating point variable

```
double radius;
double area;
```

Storing user input – working with variables

- For now, we assign a fixed value to `radius`:

```
radius = 20;
```

- Next we compute `area`:

```
area = radius * radius * 3.14159;
```

- Final step is to display the value of area

INFS1609/2609 Fundamentals of Business Programming

Listing 2.1 (textbook page 57)

https://liveexample.pearsoncmg.com/html/ComputeArea.html

This method of reviewing the program is called *Tracing a program*

INFS1609/2609 Fundamentals of Business Programming

Checkpoint

Variables

- Variables can represent data of different **types**
- When you declare (i.e., create) a variable, you need to tell the compiler two things:
  - The **name** of the variable
  - The **type of data** it can store

Declaring Variables

```
int x;              // Declare x to be an
                    // integer variable;
double radius;      // Declare radius to
                    // be a double variable;
char a;             // Declare a to be a
                    // character variable;
```

INFS1609/2609 Fundamentals of Business Programming

Different **Data Types** in Java:

Name	Range	Storage Size
`byte`	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed
`short`	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
`int`	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
`long`	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
`float`	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
`double`	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

You can also **assign/change the values** of the variables:

- Assignment statements:

```
x = 1;              // Assign 1 to x;

radius = 1.0;       // Assign 1.0 to radius;

a = 'A';            // Assign 'A' to a;
```

You can do both in one step:

- `int x = 1;`
- `double d = 1.4;`

Named Constants

- While the values of a variable can change, a named **constant** represents data that never changes
- Syntax:

```
final datatype CONSTANTNAME = VALUE;
```

- Example:

```
final double PI = 3.14159;
final int SIZE = 3;
```

** A constant must be declared and initialized in the same statement

Reading User Input from Console

We have learned to compute the `area` of a circle with a fixed `radius`. Now let's learn how to create a program that can accept input (e.g. different radius) from the user

The Scanner Class

- **The Oracle Java .doc**
  https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

1. Create a Scanner **object**

```
Scanner input = new Scanner(System.in);
```

2. Use a **method** to obtain a value. e.g.,

```
int value = input.nextInt();

double d = input.nextDouble();
```

Methods that are available for the Scanner object

Method	Description
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.

Java `import` statement

```
java.util.* ; // Implicit import
```

(asterisk (*) wildcard character – import entire package)

```
java.util.Scanner; // Explicit Import
```

(No performance difference)
https://docs.oracle.com/javase/tutorial/java/package/usepkgs.html

Lets see it in action!

ComputeAreaWithConsoleInput

ComputeAverage

Quickly re-examine the code from the input-processing-output perspective

Now let's discuss some good programming practices

which are important to prevent errors

INFS1609/2609 Fundamentals of Business Programming

Your turn to write some code!

Ed Workspace

Identifiers

- Names given to identify an element (e.g. class, method, variable)
- Descriptive identifiers make programs easy to read (avoid abbreviations)
- Some rules for Java:
  - An identifier must start with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit
  - An identifier cannot be a reserved word (what are some reserved words you have learned today?)
  - Java is **case sensitive**: `area`, `Area` and `AREA` are all different identifiers

Naming Conventions

Variables and method names:

- Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name
- For example, the variables `radius` and `area`, and the method `computeArea`

Class names:

- Capitalize the first letter of each word in the name
- For example, the class name `ComputeExpression`

Naming Conventions

Constants:

- Capitalize all letters in constants, and use underscores to connect words
- For example, the constant `PI` and `MAX_VALUE`

About Programming Errors

What happened if:

- You try to assign a value to a variable before declaration?
- You try to use the Scanner without import?
- You try to assign a value of different type to a variable?

About Programming Errors

Syntax Errors

- Detected by the compiler

Runtime Errors

- Causes the program to abort

Logic Errors

- Produces incorrect result

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

ShowSyntaxErrors

```
public class ShowRuntimeErrors {
  public static void main(String[] args) {
    System.out.println(1 / 0);
  }
}
```

ShowRuntimeErrors

```
public class ShowLogicErrors {
  public static void main(String[] args) {
    System.out.println("Celsius 35 is Fahrenheit degree ");
    System.out.println((9 / 5) * 35 + 32);
  }
}
```

ShowLogicErrors

Reflection

What have you learned today?

INFS1609/2609 Fundamentals of Business Programming

Up next!

- Data types
- Operators, increments, decrements
- Casting conversion
- Comparing primitive type variables with reference type variables

Thank you!