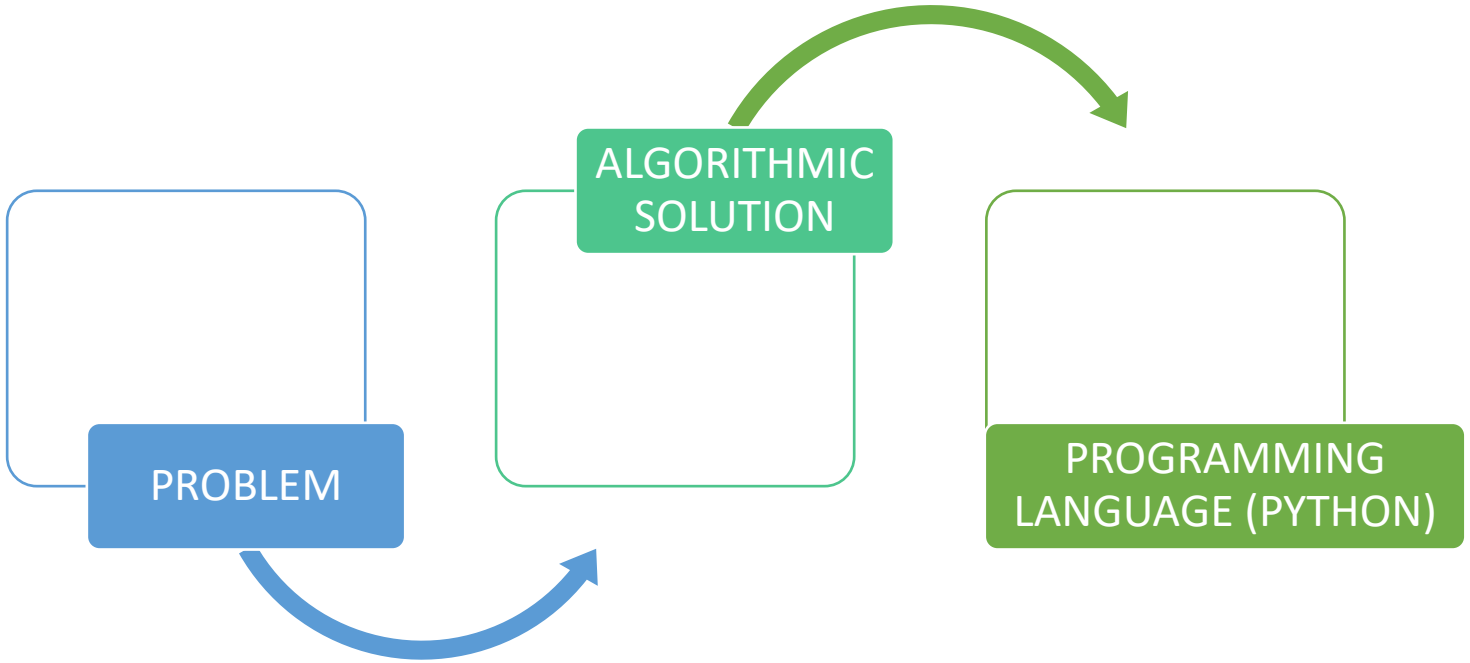
Introduction to Java

CSGE601021 Dasar-Dasar Pemrograman 2
Fakultas Ilmu Komputer Universitas Indonesia

Today's Menu

- Motivasi: Programming, Java vs Python
- Introduction to Java
- Compiling & running a Java program
- Objects, classes, methods
- Instance variables
- Local variables
- Object references
- Implicit parameters

What did you learn in DDP1?



- You organized/ wrote instructions (program) to be executed by a computer
- You're telling the computer **how** to do a certain thing!
- You're telling the computer what to do to **solve a problem.**

Untuk apa belajar DDP (lagi)?

Python Program: 1

```
‘’’ hello.py ‘’’
#!/usr/bin/env python3

print('Hello, Python!’)

$ python hello.py
```

What is the output?

Python Program: 2

```
''' greet.py '''
#!/usr/bin/env python3
import sys

def main(arg):
  print("Hello, {}!".format(arg))
```

```
if __name__ == '__main__':
        if len(sys.argv) == 2:
            main(sys.argv[1])
        else:
            print('Error: invalid number of arguments')
            print('Usage: python greet.py ASTRING')
            print('Example: python greet.py World')
```

What is the output?

```
$ python greet.py
```

```
$ python greet.py DDP2
```

```
$ python greet.py Aku Suka DDP2
```

Python Program: 3

```
''' min.py '''
#!/usr/bin/env python3
from random import randint
import sys
if __name__ == '__main__':
        if len(sys.argv) == 2:
        try:
                amount = int(sys.argv[1])
        numbers = [randint(0, 100) for x in
                                range(amount)]
        minimum = numbers[0]
for n in numbers:
        if n <= minimum:
                minimum = n
                print('Minimum: {}'.format(str(minimum)))
        except ValueError:
            print('Error: invalid type of argument')
```

```
$ python min.py 100
```

```
$ python min.py 100hehe
```

What is the output?

Computational Thinking

- Learning programming == learning to design solutions
- It should not matter what language you are using.
- The single most important skill for a computer scientist is problem solving.
- It's the ability to:
  - formulate problems,
  - think creatively about solutions, and
  - express solutions clearly and accurately.
- As it turns out, the process of learning to program is an excellent opportunity to develop problem solving skills.

Refleksi

- Bagaimana komputer membaca instruksi pada contoh program Python di awal?

Procedural Programming

- Basic programming model used from a long time ago
- Giving instructions **in order,** to execute tasks
- Data is separate from procedure

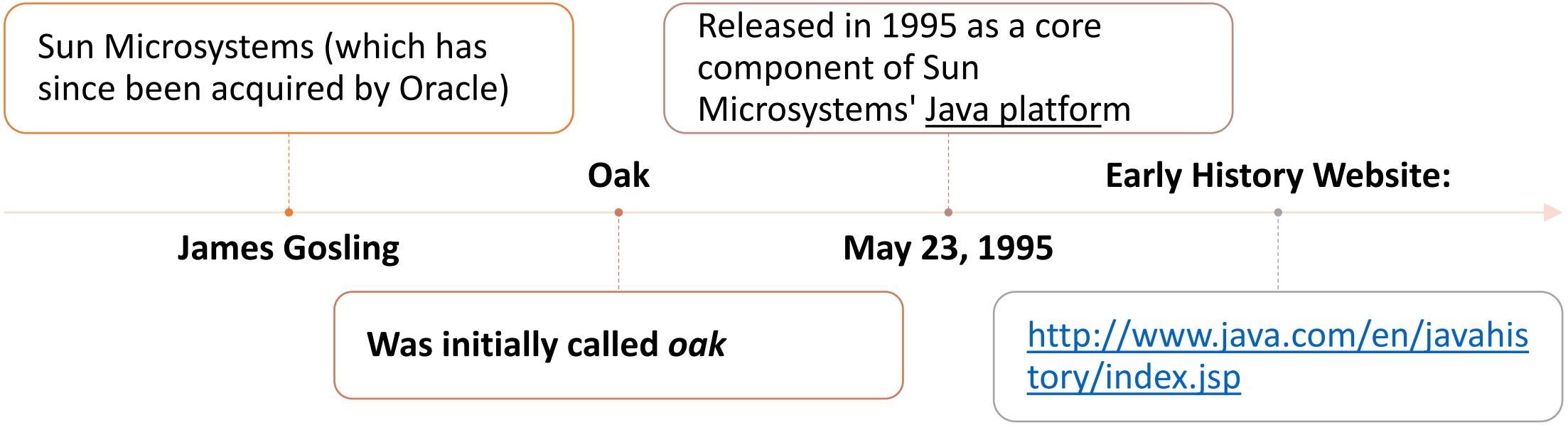
Object-Oriented Programming

- Natural mapping with concepts and their relation in context of the problem / task
- Data and procedures in 1 place (class)
- Increases code reusability

We will look into this in more detail later in the course!

We will be using Java to learn OOP

- Battle-tested, 20++ years used in industry and research
- As an “old” language, it is still continuously updated
- Java Syntax is relatively easy for understanding object-oriented concept



Characteristics of Java

- Java Is Simple
- Java Is **Object-Oriented**
- Java Is Distributed
- Java Is **Interpreted**
- Java Is Robust
- Java Is Secure
- Java Is **Architecture-Neutral**
- Java Is **Portable**
- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

- Mainly
- Object-oriented as opposed to procedural
- Reusability and modularity

- You need an interpreter to run Java programs.
- The programs are compiled into bytecode.
- The bytecode is machine-independent and can run on any machine with a Java interpreter

- WORA - Write once, run anywhere
- With a Java Virtual Machine (JVM), you can write one program that will run on any platform.

Java vs Python

	Python	Java
Language Type	Python is **dynamically** typed	Java is **statically** typed
Object-oriented Programming (OOP)	Python supports many (but not all) aspects of OOP. But it is possible to write a Python program without making any use of OO concepts.	Java supports only **object-oriented programming.**

Java vs Python (2)

	Python	Java
Variables	A variable is introduced by assigning a value to it: **someVariable = 42**	A variable must be explicitly declared of a type before assigning a value to it: **int someVariable;** **someVariable = 42;** OR **int someVariable = 42;**
	A variable that has been assigned a value of a given type may later be assigned a value of a different type: **someVariable = 42** **someVariable = 'Hello, world'**	A variable that has been declared to be of a particular type may **NOT** be assigned a value of a different type. **String someVariable = "hello";**

Java vs Python (3)

	Python	Java
Variable Types	Functions, methods, classes, code blocks, namespaces, numbers, strings, and so forth, are all treated as **Objects**. Python supports the following built-in data types: 1. **Plain integers** 2. **Long integers** 3. **Booleans** 4. **Real numbers** 5. **Complex numbers**	Java has two kinds of data types: 1. **primitive types** 2. reference types Java supports the following primitive data types: 1. **byte- 8-bit integers** 2. **short- 16-bit integers** 3. **int- 32-bit integers** 4. **long- 64-bit integers** 5. **float- 32-bit** 6. **double- 32-bit** 7. **boolean- (false and true)** 8. **char- a single character**

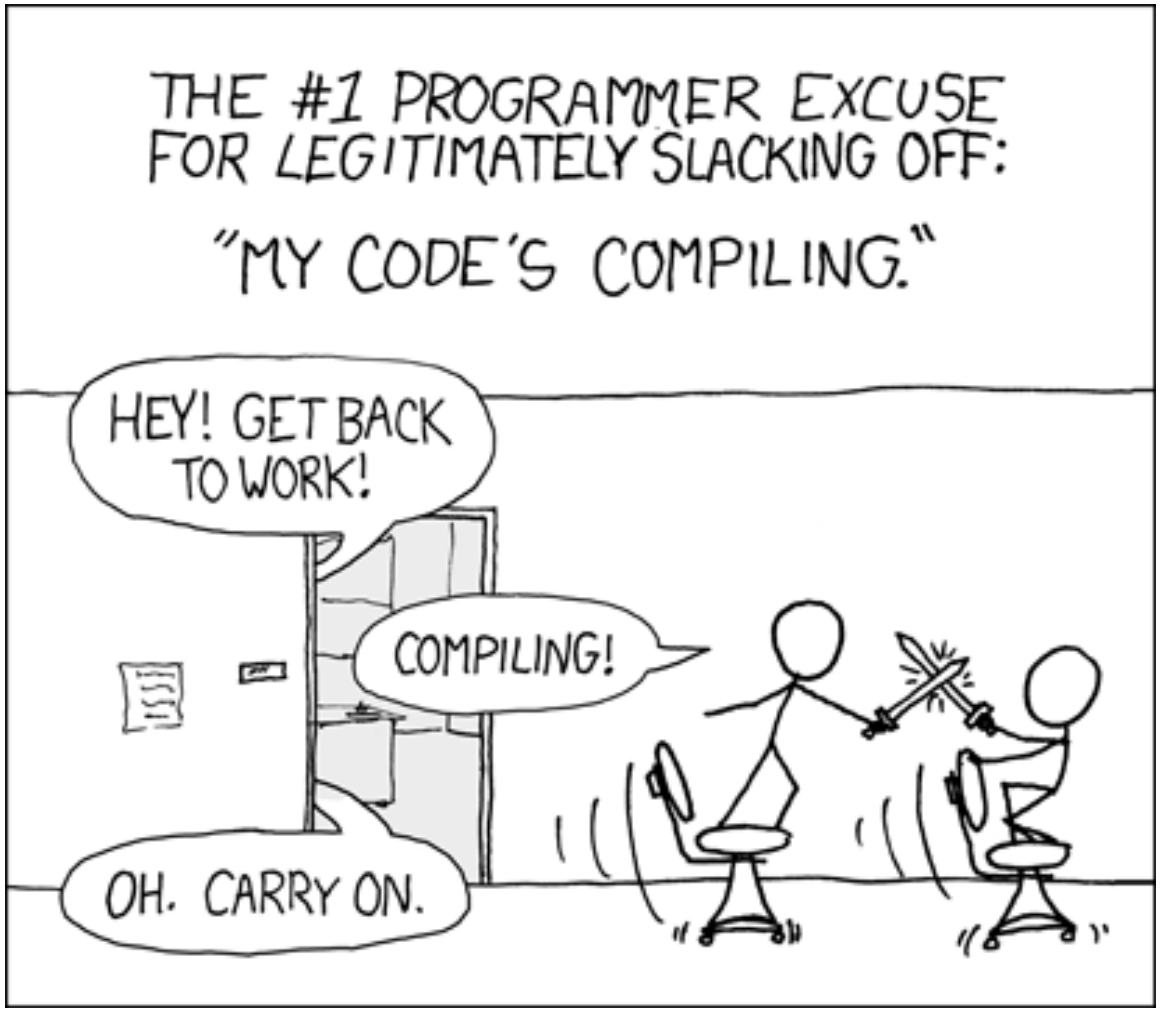
Java vs Python (4)

	Python	Java
Comparison Operator	The comparison operators **(>,<, >=, <=, ==and !=)** can be applied to numbers, strings, and other types of objects), and compare values in some appropriate way	Most of the comparison operators **(>, <, >=, and <=)** can be applied only to primitive types. Two (==and !=) can be applied to any object, but when applied to reference types they test for same (different) object rather than same (different) value.

Java vs Python (5)

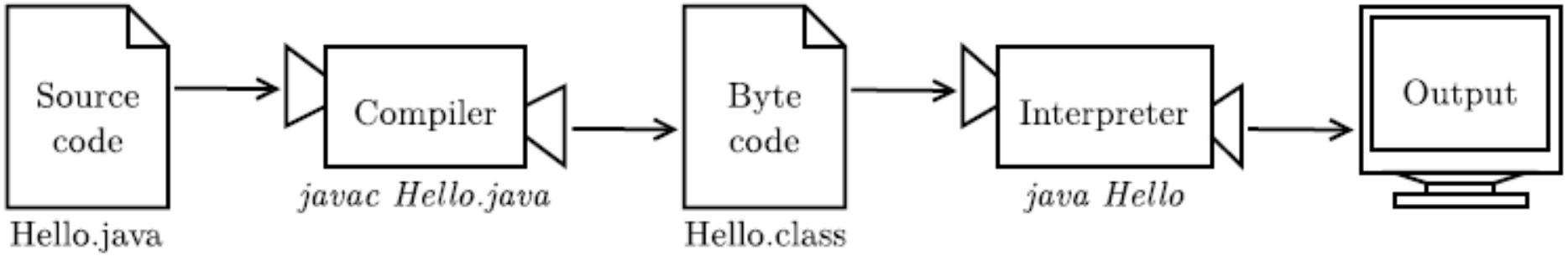
	Python	Java
Usage	Python is designed to be used **interpretively**. A Python statement may be entered at the interpreter prompt (>>>) and will be **executed immediately.**	Programs written in Java must be **explicitly compiled** into bytecodes (.classfiles)

Compiling and running a Java program



https://xkcd.com/303/

Compiling and running a Java program



- Java is both **compiled** and **interpreted**.
- Instead of translating programs directly into machine language, the Java compiler generates byte code.
- Similar to machine language, byte code is easy and fast to interpret.
- But it is also portable, so it is possible to compile a Java program on one machine, transfer the byte code to another machine and run the byte code on the other machine.
- The interpreter that runs byte code is called a "Java Virtual Machine" (JVM).

A Simple Java Program

- Suatu program terbuat dari satu atau lebih class
- Suatu class terdiri satu atau lebih method
- Suatu method terdiri dari statement program
- Suatu aplikasi Java selalu terdiri sebuah method main

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

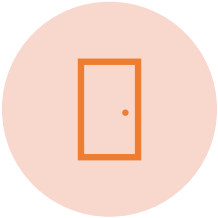
Anatomy of a Java Program

CLASS NAME

MAIN METHOD

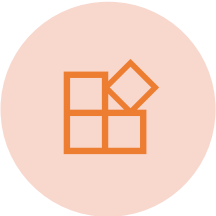
STATEMENTS

STATEMENT TERMINATOR



RESERVED WORDS

COMMENTS



BLOCKS

Class Name

- Every Java program must have at least one class.
- Each class has a name.
- By convention, class names start with an **uppercase** letter.

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

Main Method

- A Java class consists of methods
- In order to run a class, the class must contain a **method named main**.
- The program is executed from the main method.

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

Statement

- A statement represents an action or a sequence of actions.
- For example: `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!");
  }
}
```

Statement Terminator

- Every statement in Java ends with a semicolon (;).

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

Reserved words

- Reserved words or keywords are words that have a specific meaning to the compiler
- Cannot be used for other purposes in the program.

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

Find the Special Symbols

- { … }, (…), […], //, “ … ”, ;

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

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.

Ensuring Readability

- We could write something like this

```
public class Goodbye { public static void main(String[] args)
{ System.out.print("Goodbye, "); System.out.println
("cruel world");}}
```

- This one reads better :)

```
public class Goodbye {

    public static void main(String[] args) {
        System.out.print("Goodbye, ");
        System.out.println("cruel world");
    }
}
```

Blocks

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

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

Class block

Method block

Block Styles

- Use end-of-line style for braces.

Next-line style

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

End-of-line style

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

Programming Style and Documentation

Appropriate Comments	Summary at the beginning of the program /class to explain the program /class Include details at the beginning of the program.
Naming Conventions	Choose meaningful and descriptive names. Class names - capitalize the first letter of each word in the name.
Indentation and Spacing Lines	Indentation - Indent two spaces. Spacing - Use blank line to separate segments of the code.
Block Styles	Use end-of-line style for braces.

Old School Java Program

- Compiling
  - Write your program in a text editor (e.g. edit, notepad, notepad++)
  - Save file .java, for example: `ProgramKu.java`
    - Careful: case sensitive!
  - Open command prompt
  - Compile Java byte code: `javac ProgramKu.java`
- Running
  - Program will run "on" Java Virtual Machine (JVM)
  - Execute byte codes
    - `java ProgramKu`
  - Done!

Common Java IDEs

- Eclipse
- Intellij
- NetBeans
- VSCode
- ..
- And so many more….

Error types

Syntax error

Runtime error

Logic error

Syntax Errors

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

- Also known as compile-time errors

```
C:\Users\Laksmita Rahadianti\Desktop>javac ShowSyntaxErrors.java
ShowSyntaxErrors.java:2: error: invalid method declaration; return type required
  public static main(String[] args) {
                ^
ShowSyntaxErrors.java:3: error: unclosed string literal
    System.out.println("Welcome to Java);
                       ^
ShowSyntaxErrors.java:3: error: ';' expected
    System.out.println("Welcome to Java);
                                         ^
ShowSyntaxErrors.java:5: error: reached end of file while parsing
}
 ^
4 errors
```

Runtime Errors

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

- It will compile
- Errors occur during running

```
C:\Users\Laksmita Rahadianti\Desktop>javac ShowRuntimeErrors.java

C:\Users\Laksmita Rahadianti\Desktop>java ShowRuntimeErrors
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ShowRuntimeErrors.main(ShowRuntimeErrors.java:3)
```

Logic Errors

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

- It will compile and run
- But will not complete the task correctly

```
C:\Users\Laksmita Rahadianti\Desktop>javac ShowLogicErrors.java

C:\Users\Laksmita Rahadianti\Desktop>java ShowLogicErrors
Celsius 35 is Fahrenheit degree
67
```