ITI 1120
Lab # 1

Contributors: G. Arbez, M. Eid, D. Inkpen, A. Williams, D. Amyot

Communicate with us …

- Who to contact
  - Not urgent and public: Use the Virtual Campus forum
  - Marking of assignments: The TA who marked it
  - Urgent and private: Professor, by e-mail

Assignment 1

- The assignment must be submitted on the due date before midnight.
- Individual work
- Read "Assignment Instructions"
  - When/how/what/where to submit assignments
  - Submission is made via Virtual Campus
  - Marks are deducted if you do not follow rules

- Recommendation: Assignments are available. Read each assignment BEFORE the lab, so that you may ask helpful questions.

3

For today's lab:

- Go the course page on virtual campus
- Follow the links to the lab notes for Lab 1.
- Save all the java programs you find there in the C:\work directory.
- We'll be using them later.

4

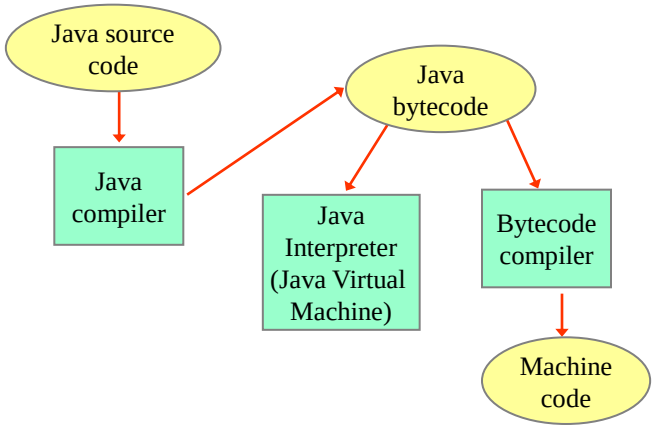
Introduction to Java

- Editing, compiling, and running Java programs
- Syntax, comments
- To get to the point that you can write Java programs:
  - Declare variables
  - Read data from the keyboard
  - Numbers, Operators, Calculations
  - Print the results.
- Translate a simple algorithm into Java

5

Translation in Java

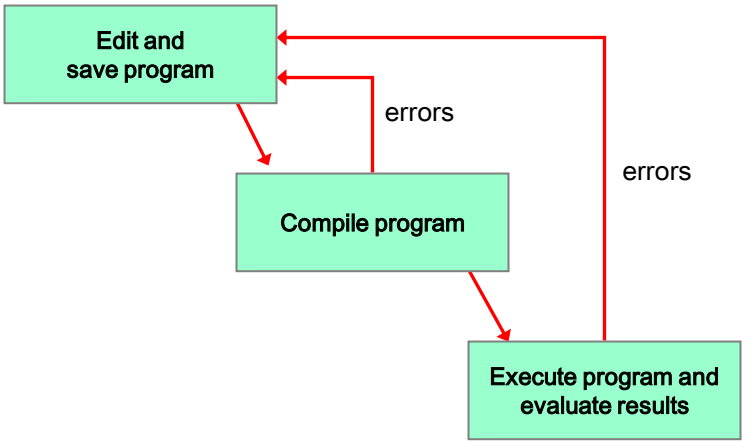
- Do you recall this image you have seen in class?



6

Basic Program Development

- These are the steps you will be following today.



7

Editing a Java program

- An editor is used to create and modify a program.
- The program is just an ordinary text file, so you can use any text editor
  - You don't need to use a special one, although editors specifically for programs can be much more convenient.
- This text file with the program in it is called the source file for the program
- Java source files have the extension `.java`

8

Compiling

- The source file - the program in textual format - cannot be directly executed.
- It needs to be translated into an executable form.
- A compiler also checks for certain types of errors, called syntax errors.
- If it detects an error you need to figure out what caused the problem and go back and edit the the source file to fix the problem.

9

Syntax Errors

- A "grammatical" error
  - Example: `whle` instead of `while`
- Found by compiler ("compiler-time error")
  - Automatically found
  - Usually the easiest to fix
- Cannot run code until all syntax errors are fixed in all open files.
- Be careful when reading messages from the compiler:
  - Problems are described from a grammatical point of view.
  - Sometimes, the message is a "best guess" of what might be wrong, and may be misleading.

10

Run

- If the compiler detects no problems, it creates a byte code file having a `.class` extension.
- It does not automatically run your program - that is done by you manually.
- Since the `.class` file stays around you can run it whenever you like, you don't have to recompile every time.
  - … But of course if you change the source code you must manually recompile or you'll be using the old executable.
- Once the program is running it is important to test it to make sure it produces correct answers.
  - You'll usually run it may times on various inputs to make sure it works fine under all circumstances.
- If you find it is producing the wrong answer, you have to debug the program
  - this will involve editing the program (the source file), recompiling, rerunning.

11

Exercise 1 - Overview of a Java Program

- Start Dr. Java
- Open ("load") the file **`Prog1.java`**
  - You should already have saved this file on your hard drive).

12

Compiling the Java Program

- To compile `Prog1.java` with Dr. Java, click on the button "Compile". This will compile all files listed in the left window.
- Compiler messages appear under the tab "Compiler Output" at the bottom of the window.
  - Shows if the compilation was successful.
  - Otherwise the compiler produces error messages.
- In the directory where `Prog1.java` is stored
  - The compiler will have produced the file `Prog1.class`

13

Running a Program

- Now that the program is compiled, you can run it
- Click on "Run" (or type F2)
- This will execute the method `main` of the program you edited.
- In the Interactions zone (see tab at the bottom), you will see the program output
  - You can also click on the tab "Console" to see only program output with any messages generated by Dr. Java

14

General Organization

- Source file contains a CLASS.
  - We will always have one class per file.
- A CLASS contains one or more METHODS.
- A METHOD contains declarations, statements, and control structures.
  - This is where you will implement your algorithms.
- A PROGRAM must include a class that has a method called **`main`**
  - We shall see in the second half of the course how many classes can make up a program.
- COMMENTS can be placed anywhere.

15

Comments

- Comments are for people reading your program.
  - In them you explain your program in English.
  - The compiler completely ignores them.
- In Java
  - Comments may be placed anywhere you like.
  - On any line, everything after `//` (to the end of the line) is a comment.
  - Anything in between `/*` and `*/` is a comment (including multiple lines)
- See **`Prog1.java`** as for examples

16

Types of comments

- Single line comment
  - Everything from `//` to the end of the line is the comment

```
some code  // This is a comment
more code
```

- General comment
  - Everything from `/*` to the next occurrence of `*/` is a comment
  - Can be part of a line `code /* comment */ more code`
  - Can be several lines

```
code /* start of comment
more comment
end of comment */ more code
```

17

Class Definition

- Has these parts:
  - Keyword `class`
    - A keyword is a word that has special meaning in the Java language. Dr. Java highlights these reserved words by colouring them blue.
    - In this case the keyword `class` tells the compiler that you are beginning the definition of a class.
  - A name for the class
    - `Prog1` is the name of a class
  - Methodes
    - An opening `{` <-- this symbol is called a brace or curly bracket
    - One or more method definitions
    - A closing `}`

- Braces are used to enclose lines of code together to form an instruction block.

18

Identifiers

- The class has the name **`Prog1`**
- In programming, the official term for the name is an "identifier".
  - Identifiers are used to name many things: classes, methods, variables, etc.
- There are rules for identifiers in Java
  - Only use the digits **`0-9`**, letters **`A-Z a-z`**, the characters **`$`** and **`_`**
  - Identifiers cannot start with **`$`** and it is not recommended to start them with **`_`** (underscore)

19

`main` method definition

- The definition of **`main`** starts with a line that we will never change for this course:

```
public static void main(String[] args)
```

- **`main`** is the name of this method; it is a special identifier, like a keyword.
  - The purpose of the **`main`** method is to tell Java, "when you run the program, start here."
- After this opening line comes:
  - An opening **`{`**
  - The "body" of the method - in the example program main's body consists of two statements
  - A closing **`}`**
- Next week in the lab session, we shall add another method that will be called by **`main`**.

20

The `println` and `print` statements

- The simplest forms:

  `System.out.println( "some string" );`
  - Go to the next line
  - `System.out.print ( "some string" );`
    - Stays on the same line, any new printed character or typed in character will follow the message
- A STRING is a collection of characters, contained in double quotes to mark the start and the end of the string.
- Whatever is between the double-quotes is written ("printed") on the console (the screen).
- After the string is printed, the cursor marking the location of where the computer will print next is moved to the start of a new line.
- Note: the quotes are not part of the string.

21

The "import" Statement

- Indicates to the compiler which libraries (or set of predefined classes/methods) the program uses (or may potentially use).
- In `Prog1.java`, we are interested in all classes (`*`) and input/output methods (io). For example, this import includes `System.out.println`
  - The current version of Java does not require this particular import; it is done automatically
- There can be many "`import`", usually placed at the start of the file (and always before any of its classes are used).

22

Syntax - General Features

- Java is "free format".
  - In general, you can have blank lines and space things the way you like.
  - However, there are some restrictions for how to space and place things. You cannot put spaces (or line breaks) in the middle of names or keywords.
  - There are conventions to make programs more readable and understandable by many people (e.g. indentation).
- Java is case-sensitive.
  - `class` and `Class` are two different words
    - keywords never use capitals
  - This is a common source of bugs
- Java is VERY PARTICULAR about punctuation.
  - If you miss a semicolon or have mismatched brackets or braces or double-quotes, or if you use a single quote (or two) instead of a double quote, you'll get a syntax error.

23

Some general rules are:

- All brackets must match with a bracket of the same type, opposite direction (open and close pairs)
  - The open-close pairs must fit ("nest") inside each other
    - You can't do this: `( [ ) ]`
- Double quotes must match up ON THE SAME LINE
- All statements end with a `;` (semicolon)
- Braces are normally NOT followed by a semicolon (there are some exceptions in special cases).
- The class name and the file name should be the same (except of course for the `.java` extension on the file name).

24

Exercise 2 - `Prog2`

- Try the same thing with `Prog2.java`
- What happened?

25

`Prog2`

- You will get error messages because there is one mistake in `Prog2.java` (the quote to end the string in the `println` statement is missing).
- This is what syntax error messages look like
  - Where does it say what line the error occurred on?
  - Why does the compiler think there are two errors?
    - Hint: Notice that Dr. Java colours strings red. Note carefully what is coloured red in this program.
- Fix the error, and re-compile
  - When you fix the error, notice the difference in what is coloured red.

26

Exercise 3 – Prog3

- This program illustrates one of the most common errors. Try it!

27

Exercise 4 – Prog4

- This program shows the difference between println and print. Try it!

28

Exercise 5 - Prog5

- Try to compile and run this program.
- What happened?

29

Exercise 6 - Prog6 - Correcting Syntax Errors

- Correct all errors in Prog6.java so that it will produce the following output:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This program used to have lots of problems,
but if it prints all the lines on the screen,
you fixed them all.
             *** Hurray! ***
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

30

Exercise 7 – Working with a Circle

- The algorithm on the following slide defines a simple program that
  - Gets a circle radius from the user,
  - Calculates the circle's diameter, circumference, and area.
  - Displays the results.
  - The constant value of 3.14 is used for π.
- Translate the algorithm to a **`main`** method in Java.
- Compile and test the program.

31

Exercise 7

GIVENS: (none)
RESULTS: (none)
INTERMEDIATES: radius (the radius of the circle)
diameter (the diameter)
circumference (the circumference)
area (the area)
HEADER: main()
BODY:

```
print("Please enter a value for radius: ")
radius ← readReal()
diameter ← 2 x radius
circumference ← 2 x 3.14 x radius
area ← 3.14 x radius x radius
printLine("The diameter of the circle is ", diameter)
printLine("The circumference of the circle is ", circumference)
printLine("The area of the circle is ", area)
```

32