Grundlagen der Programmierung

Session II - Basics of Java Programming (1/2)

Romain PELISSE - Red Hat Gmbh

Humboldt Universität, Berlin

2012

Agenda

Session outline

- how to use an IDE (Eclipse)
- control structure (if, for, while, .switch,...)
- functions
- error handling

What is an Integrated Development Environment ?

Main features

- Advanced file editing
  - highlight syntax with colors
  - syntax checker
  - auto completion
- Program building
  - compilation
  - run program inside the IDE
- **debugger**
  - execute program **interactively**
  - expose variables content, allow to modify them direclty

A short Eclipse Tutorial

Eclipse IDE

- Open Source, java based, IDE
- Most used IDE in the World
- Focus on Java programming but also used for other technologies
- Cross platform program
- Speed up development (up to 40% !)

Use Eclipse (1/2)

Instructions

- Look for Eclipse in the Application launcher menu, and start it
- Eclipse will ask for the location of the **workspace**
  - this location is where Eclipse stores the project's files
  - simply click on "OK" and accept the default location
- Go to File->New->Java Project to create a new, empty, Java project.
- Use your first and lastname (ex :romain-pelisse) as project name and click sur "Finish"
- **Right-click** on the folder 'src' and select the 'import' options :
  - a wizard will appear to specifify the import method
  - select General->File System and click on "Next"
  - use the "Browser" button to select the "Desktop" folder
  - select all the java sources files (.java) on Deskopt (all files from the previous session)
  - click on "Finish" to import all of them

Use Eclipse (2/2)

Instructions

- Compile, run and **debug** :
  - Eclipse will compile for you, automatically any Java files in the src/ folder - nothing to do !
  - To run a Java file, right-click on the file and select Run as->Java Application
  - To Debug a program, right-click on the file and select Debug as->Java Application

Use Eclipse's debugger

Instructions

- Double click on the column on the left of the source code to set a **breakpoint** - see picture below
- Run the program with the debugger (Debug as->Java Application) and ...
  - run the program step-by-step
  - examine the values of each available variables
  - set a breakpoint and execute the program up to it

The 'if' statement

The 'if' statement

- simple form of control flow statement.
- directs the program to execute a certain section of code...
- ...if and only if the test evaluates to true
- in Java, a the test must be boolean expression.

Instructions

- Download the HowToUseIfStmt.java file and save it on the Desktop
- Import the file in your project as you did before
- Implements the program as defined in the source file
- To add an 'if' statement to the code, use the autocompletion feature of Eclipse (crtl+space)

The 'for' statement

The 'for' statement

- execute a block of code continuously
- consists of tree parts :

initialization	: an expression that sets the value of the loop control variable, executed only once.
condition	: a boolean expression that tests the loop control variable against a target value and hence works as a loop terminator.
Increment/decrement	: an expression that increments or decrements the loop control variable.

Instructions

- Using Eclipse, copy the file HowToUseIfStmt.java to HowToUseForLoop.java
- This new program accepts only one character values as arguments :`java HowToUseForLoop 1 2 3 4 5 6 7` - there no more bound for the number of arguments.
- Modify the program accordingly
- To add a 'for' statement to the code, use the autocompletion feature of Eclipse (crtl+space)

The 'while' statement

The 'while' statement

- execute a block of code continuously...
- ... as long as the condition is true.
- mix between an 'if' and a 'while'
- a do...while statement also exist - to skip the first evaluation.

Instructions

- Download the file WhyWhileStmtAreAlsoCool.java
- Modify the program as requested in the code

The 'switch' statement

The 'switch' statement

- an alternative to a series of 'else if'
- allows for any number of possible execution paths
- works with the byte, short, char, and int primitive data types
- works with enumerated types (we'll see them later)
- body of a switch statement is known as a switch block.
- Any statement immediately contained by the switch block may be labeled :
  - with one label
  - several lables
  - or the default label

Remarks

- powerful but rather complex
- seldom used

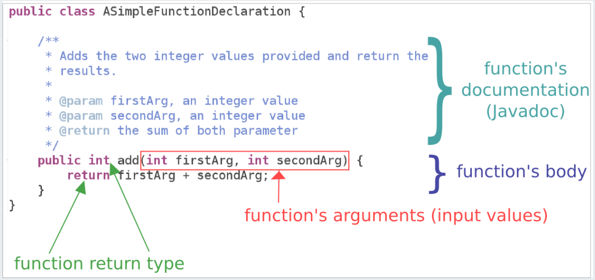
What is function ?

In computer science, a subroutine, also termed procedure, function, routine, method, or subprogram, is a part of source code within a larger computer program that performs a specific task and is relatively independent of the remaining code. - Wikipedia "Subroutine", accessed the 22.05.2012

What are their purpose ?

- reuse code easily
- make code easier to read
- breakdown code in different, meaningful, part
- hide complexity

Functions syntax



```
public class ASimpleFunctionDeclaration {

    /**
     * Adds the two integer values provided and return the
     * results.
     *
     * @param firstArg, an integer value
     * @param secondArg, an integer value
     * @return the sum of both parameter
     */
    public int add(int firstArg, int secondArg) {
        return firstArg + secondArg;
    }
}
```

function's documentation (Javadoc)
function's body
function's arguments (input values)
function return type

Instructions

- Download and import the file called LenghtyProgram
- Modify the code according to the instruction

Structure to handle error in programming language

- do nothing
  - program crashes
  - no information on the root cause
- use "status code"
  - functions can't return value
  - leads to message such as "Error 400 happened"
  - needs to have an error database to translate the status code
- return a complete structure describing in length the error
  - ideal, but...
  - ...still remove the option of having returning value