Chapter 1

Getting started

Lecture slides for:

*Java Actually: A First Course in Programming*
Khalid Azim Mughal, Torill Hamre, Rolf W. Rasmussen
ISBN: 978-1-844480-418-4
http://www.ii.uib.no/~khalid/jact/

Permission is hereby granted to use these lecture slides in conjunction with the book.

Modified: 2007-April-15

### Overview

- The goal of the course: problem solving on a computer
- Programming
- Editing source code
- Building programs
  - Compiling Java programs
  - Compilation errors
- Running Java programs
- Object based programming (OBP)
- The Java programming language
  - Comments and indentation
  - Program entry point
  - Statements and variables
- Byte code and the Java Virtual Machine
The goal of the course: problem solving on a computer

The goal of the course...

is not to learn a specific programming language (of which there are many: Java, C++, Python, Ada, Simula, Lisp, Pascal),

but instead...

to give an introduction to the craft of programming.

This includes:
• techniques for analysis of problems
• constructing solutions using the Java programming language

The choice of programming language is a pragmatic choice, and is not the main focus of this course.

Software development methodology is largely independent of programming language.

Programming

• How we use computers:
  • **Programmers**: write programs that accomplish a specific tasks, e.g.:
    • Word processors
    • Drawing programs
    • Spreadsheets.
  • **End users**: Use programs written by programmers.

• **Instructions**:
  • A program is a set of instructions
  • Computers execute sequences of instructions quickly, accurately and reliably.

• **Programming language**:
  • Is used to express programs
  • Is human readable descriptions of the tasks the computer should perform
  • A program in this form is called source code and is stored in text files
  • Will be translated to instructions for execution by the computer
  • Java is one of many programming languages.
Overview of programming activities

- Activities sufficient for writing small program:
  - (start)
  - Edit source code
  - Build program
  - Test program
    - [no build error]
    - [build error]
    - [deficiencies or errors found]
    - [the program is done]
  - (end)

- Many other activities are involved when writing larger programs.

Source code (example)

```java
// (1) This source code file is called SimpleProgram.java
public class SimpleProgram {
    // Print a proverb, and the number of characters in the proverb.
    public static void main(String[] args) {
        // (2)
        System.out.println("A proverb:");
        // (3)
        String proverb = "Practice makes perfect!";
        System.out.println(proverb);
        // (4)
        int characterCount = proverb.length();
        System.out.println("The proverb has " + characterCount + " characters.");
    }
}
```
**Editing source code**

- We write the source code in text files:
  - Commonly called *source code files*.
  - Describes exactly what tasks the computer should perform.
  - Contain only characters that constitute the actual text of the source code, (no formatting).
  - Choose a good editor for writing source code.

- The compiler requires the source code files to be named according to specific rules:
  - Correct: `SimpleProgram.java`
  - Incorrect:
    - `simpleprogram.java` (wrong case)
    - `SimpleProgram.java.doc` (wrong extension)
    - `Simple-1.java.doc` (Microsoft Windows short-names not allowed)

**Build program: Compiling Java programs**

> `javac SimpleProgram.java`

(a) One class in the source code file

> `javac MainClass.java`

(b) Two classes in the source code file
Build program: Compilation errors

- The compiler translates source code to byte code.
- It may detect errors in the source.
- The compiler will report any errors and terminate the compilation.
- The errors must be corrected in the source code and the compiler run again to compile the program.

```
> javac SimpleProgram.java
SimpleProgram.java:9: ')' expected
  System.out.println(proverb;                         // (5)
^  The “^” indicates where the error is located

1 error
```

(Oops, forgot the closing parenthesis.)

Running Java programs

```
> java -ea SimpleProgram
```

(Run this on the command-line)

- Specify the exact class name, without any “.class” or “.java” extensions.
- Check the use of upper and lowercase letters in the class name.
- Make sure that the source code has been compiled.
Objects and Operations

How to make an omelette:

1. Open( ) the refrigerator
2. Take out( ) an egg carton
3. Open( ) the egg carton
4. Take out( ) two eggs
5. Close( ) the egg carton
6. ...

The type of the object determines the operations that can be performed on it:

- Open( ) the frying pan  
  (a frying pan cannot be opened)

Object based programming (OBP)

- Describing tasks as operations executed on objects.
- Define objects that are useful for the problem you’re trying to solve.
- E.g. for a program to keep track of library loans, create objects representing...
  - tangible items: books, journals, audio tapes
  - non-tangible concepts: lending date, information about library users
- Programs usually have more than one object of the same type.
The Java programming language

Comments and indentation

// This is a source code comment.  (ignored by compiler)

This will technically work...

public static void main(String[] args){
  System.out.println("A proverb:");
  String proverb = "Practice makes perfect!";
  System.out.println(proverb);
  int characterCount = proverb.length();
  System.out.println("The proverb has " + characterCount + " characters.");
}

...but don’t do it.
Please.

- Use proper indentation:
  - It makes the source code easier to read and modify.
  - Java convention: use four spaces for each indentation step
Program entry point

```java
public static void main(String[] args) {
    ...
    method body containing statements that will be executed one by one...
    
}
```

- For a Java program to be executable, it must define exactly one `main()` method.

- For **very small** programs:
  - one source code file
  - primary class in the file that contains the `main()` method

- For **larger** programs:
  - split the source code into several files
  - one class in each file
  - only one file containing the `main()` method

---

Statements

```
System.out.println("A proverb:");  // method call

String proverb = "Practice makes perfect!";  // variable declaration, variable assignment

System.out.println(proverb);  // method call

int characterCount = proverb.length();  // variable declaration, variable assignment
```

- `System.out.println("A proverb:");` is a method call with a string value as parameter value.
- `String proverb = "Practice makes perfect!";` is a variable declaration and assignment.
- `System.out.println(proverb);` is a method call with a variable value as parameter value.
- `int characterCount = proverb.length();` is a variable declaration and assignment.
Variables

- named locations in the computer’s internal storage (memory)
- holds values during program execution
- often used by methods to hold intermediate results
- storing numeric values is very common
- storing other types of values is also possible

- Store a value in a variable:
  ```java
  String proverb = "Practice makes perfect!";
  ```

- Later, use value by referring to the variable:
  ```java
  System.out.println(proverb);
  ```

Sequence of method calls during program execution

```
SimpleProgram
```
```
String proverb = "Practice makes perfect!"
```
```
System.out
```
```
println("A proverb:")
```
```
println(proverb)
```
```
length()
```
```
23
```
```
println("The proverb has 23 characters.")
```
```
The main() method call returns, and program ends
```
```
Program execution starts main()
```
**Byte code and the Java Virtual Machine**

- Java programming language:
  - a high-level language
  - provides a rich set of language
  - natural for humans to read
- Java byte code:
  - a low-level language
  - provides a small set of basic instructions
  - suited for execution by machines
  - platform independent
- Java Virtual Machine (JVM):
  - a program that interprets byte code instructions
  - not a physical machine...
  ...but behaves much in the same way as a central processing unit (CPU)
  - may virtual machines interpret the byte code directly
  - or recompile it to platform specific *machine code* during execution
  - implementations exist for several platforms (Windows, Solaris, Linux)

---

**Program code at several levels**

<table>
<thead>
<tr>
<th>Source code:</th>
<th>Java byte code:</th>
<th>x86 processor instructions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.out.println(&quot;A proverb:'&quot;);</td>
<td>getstatic &lt;Field System.out java.io.PrintStream&gt;</td>
<td>8B 15 F0 04 08 mov edx, [0x80493D0]</td>
</tr>
<tr>
<td>String proverb = &quot;Practice makes perfect!&quot;;</td>
<td>ldc &lt;String &quot;A proverb:'&quot;&gt;</td>
<td>8B 0A mov ecx, [edx]</td>
</tr>
<tr>
<td>System.out.println(proverb);</td>
<td>invokevirtual &lt;Method PrintStream.println (String)void&gt;</td>
<td>FF 51 7C call near [ecx+0xe7c]</td>
</tr>
<tr>
<td>int characterCount = proverb.length();</td>
<td>astore_1</td>
<td>E8 9D FD FF FF call 0x974</td>
</tr>
<tr>
<td></td>
<td>getstatic &lt;Field System.out java.io.PrintStream&gt;</td>
<td>85 DB test ebx, ebx</td>
</tr>
<tr>
<td></td>
<td>aload_1</td>
<td>89 DB test edx, edx</td>
</tr>
<tr>
<td></td>
<td>invokevirtual &lt;Method PrintStream.println (String)void&gt;</td>
<td>83 C8 test edx, eax</td>
</tr>
<tr>
<td></td>
<td>aload_1</td>
<td>89 B5 test eax, eax</td>
</tr>
<tr>
<td></td>
<td>invokevirtual &lt;Method String.length ()int&gt;</td>
<td>83 C4 mov esp, eax</td>
</tr>
<tr>
<td></td>
<td>istore_2</td>
<td>8B 1C 24 mov esp, esp</td>
</tr>
</tbody>
</table>

*can be further translated into processor dependent code (x86-code shown here)*

*written by programmers* **interpreted by Java Virtual Machine** **executed by Central Processing Unit**

---

*Program code at several levels (x86-code shown here)*

```
System.out.println("A proverb:'");
String proverb = "Practice makes perfect!";
System.out.println(proverb);
int characterCount = proverb.length();
```
Learning to program requires time!

- Programming requires both time and effort, however, it can be fun.
- The key to success:
  - obtain practical programming experience by work through the programming exercises
  - work gradually through the book, mastering each topic before moving on to more advanced topics
- Memorization will not make you a programmer!