

6 Class documentation and libraries

Compiling Java programs

Main concepts to be covered

Part I

- Using library classes
- Reading documentation
- Writing documentation
- Javadoc documentation tool

Part II

- Compiling and executing java programs
- Java version naming conventions

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 2

The Java class library

- Thousands of classes
- Tens of thousands of methods
- Many useful classes that make life much easier
- A competent Java programmer must be able to work with the libraries.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 3

Working with the library

You should:

- know some important classes by name;
- know how to find out about other classes.

Remember:

- We only need to know the interface, not the implementation.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 4

Reading class documentation

- Documentation of the Java libraries in HTML format;
- Readable in a web browser
- Class API: *Application Programmers' Interface*
- Interface description for all library classes

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 5

Interface vs implementation

The documentation includes

- the name of the class;
- a general description of the class;
- a list of constructors and methods
- return values and parameters for constructors and methods
- a description of the purpose of each constructor and method



the *interface* of the class

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 6

Interface vs implementation

*The documentation **does not** include*

- private fields (most fields are private)
- private methods
- the bodies (source code) for each method

➡ the *implementation* of the class

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 7

Using library classes

- Classes from the library must be imported using an *import* statement (except classes from *java.lang*).
- They can then be used like classes from the current project.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 8

Packages and import

- Classes are organised in packages.
- Single classes may be imported:

```
import java.util.ArrayList;
```
- Whole packages can be imported:

```
import java.util.*;
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 9

Writing class documentation

- Your own classes should be documented the same way library classes are.
- Other people should be able to use your class without reading the implementation.
- Make your class a 'library class'!

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 10

Elements of documentation

Documentation for a class should include:

- the class name
- a comment describing the overall purpose and characteristics of the class
- a version number
- the authors' names
- documentation for each constructor and each method

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 11

Elements of documentation

The documentation for each constructor and method should include:

- the name of the method
- the return type
- the parameter names and types
- a description of the purpose and function of the method
- a description of each parameter
- a description of the value returned

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 12

javadoc

Class comment:

```
/**
 * The Lecture class represents a lecture
 * generator object. It is used to generate an
 * automatic lecture.
 *
 * @author U. Holmer and C. Carlsson
 * @version 1.0.0.0.0.0.0.7z (9.Sep.2008)
 */
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 13

javadoc

Method comment:

```
/**
 * Read a line of text from standard input (the text
 * terminal), and return it as a set of words.
 *
 * @param prompt A prompt to print to screen.
 * @return A set of Strings, where each String is
 *         one of the words typed by the user
 */
public HashSet<String> getInput(String prompt)
{
    ...
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 14

Review

- Java has an extensive class library.
- A good programmer must be familiar with the library.
- The documentation tells us what we need to know to use a class (interface).
- The implementation is hidden (information hiding).
- We document our classes so that the interface can be read on its own (class comment, method comments).

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 15

Part II

Using Java without BlueJ

3.0

Topics

- Java file naming conventions.
- The BlueJ file structure.
- Compilation and execution of Java programs without BlueJ.
- The main method.
- Compilation of package structures.
- Java version naming conventions.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 17

Standard Java file types

- **source files: *.java**
Java source files contain the source code in readable form, as typed in by the programmer.
- **class files: *.class**
Java class files contain byte code (a machine readable version of the class). They are generated by the compiler from the source file.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 18

Java file naming

- *.java - standard Java source file (text).
 - one per class.
 - same name base and spelling as class name.
 - Ex. `class Game` is stored in `Game.java`
- *.class - standard Java code file (binary).
 - one per class.
 - same name base and spelling as class name.
 - Ex. `Game.class`

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 19

BlueJ projects

- A BlueJ project is stored in a directory on disk.
- A BlueJ package is stored in several different files.
- Some files store the source code, some store the compiled code, some store additional information.
- BlueJ uses standard Java format for some files and adds some additional files with extra information.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 20

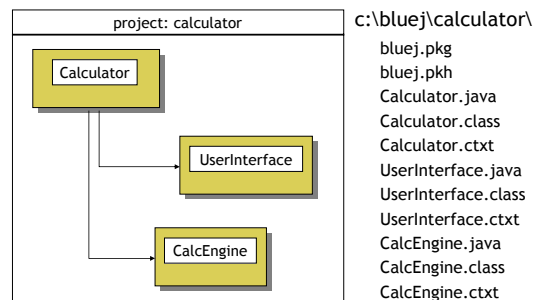
The BlueJ file structure

- *.java - standard Java source file.
- *.class - standard Java code file.
- bluej.pkg - the package file. Contains information about classes in the package. One per package.
- bluej.pkh - backup of the package file.
- *.ctxt - BlueJ context file. Contains extra information for a class. One per class.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 21

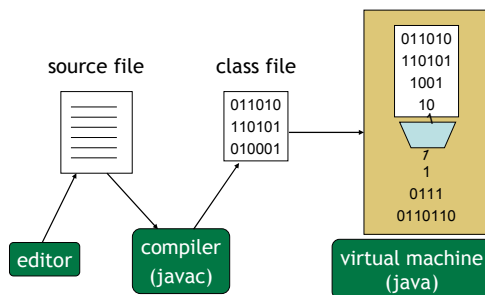
The BlueJ directory structure



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 22

The edit-compile-execute cycle



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 23

Editing

- A file can be edited in any text editor
 - Notepad, emacs, jEdit, PFE, vi, ...
- Don't use Word: by default, Word does not save in text format
- Make sure to save with a .java filename before compiling!

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 24

Command line invocation

- Compilation and execution of Java in JDK are done from a command line
- On Microsoft systems: DOS shell
- On Unix: Unix shell
- Must make sure that the commands for compiler and runtime are in the command path.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 25

Compiling

- Name of the JDK compiler: **javac**
- To invoke:
javac <source name>
- compiles <source name> and all classes it depends on
- Example:
cd C:\bluej\zuul
javac Game.java

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 26

Error messages

```
> javac Game.java
Game.java:22: ';' expected.
    private Parser parser
                   ^
1 error
>
```

The programmer has to open the file in the editor, find the line number, fix the error and recompile.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 27

Execution

- > **java Game**
- “java” starts the Java virtual machine.
- The named class is loaded and **execution** is started.
- **Other classes are loaded as needed.**
- Only possible if class has been compiled.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 28

Problem: Execute what?

- If we try:
> java Game
Exception in thread "main"
java.lang.NoSuchMethodError: main
- The problem: how does the system know which method to execute?

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 29

The main method

- The java system always executes a method called main with a certain signature:

```
public static void main(String[] args)
{ ...
}
```
- For this to work, such a method **must exist!**

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 30

The main method (2)

- “main” must exist
- “main” must be public
- “main” must be **static** (class method)
- “main” must have a String array parameter
- Only “main” can be invoked

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 31

Main method - example

```
public static void main(String[] args)
{
    Game game = new Game();
    game.play();
}
```

- The main method should
 - create an object
 - call the start method in the main object
 - be *very short*

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 32

Another example

A simple model of a mail system

- Classes
 - MailServer
 - MailClient
 - MailItem
 - Main: simple test class containing the main method
- File organization: All files are in the directory **simple_project**

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 33

Compiling the mail system (1)

Compile

```
> cd ... simple_project
> javac Main.java
```

Execute

```
> java Main
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 34

Adding package structure

- As an application grows it should be modularized using packages
 - **package mailsystem** contains the classes MailServer, MailClient and MailItem
 - **package main** contains the Main class

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 35

Adding package structure (2)

- Package declarations have to be added in the source (.java) files

```
MailServer.java:
package mailsystem;
class MailServer { ... }
MailClient.java:
package mailsystem;
class MailClient { ... }
MailItem.java:
package mailsystem;
class MailItem { ... }
Main.java:
package main;
import mailsystem.*;
class Main { public static void main ... }
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 36

Adding package structure (3)

A possible directory structure organization

```
-- package_project
-- bin
--   mailsystem
--     MailClient.class
--     MailItem.class
--     MailServer.class
--   main
--     Main.class
-- src
--   mailsystem
--     MailClient.java
--     MailItem.java
--     MailServer.java
--   main
--     Main.java
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 37

Compiling the mail system (2)

```
> cd ... package_project
```

```
> mkdir bin (if necessary)
```

Where to put binaries (.class)

Source root directory (.java)

```
> javac -d bin -sourcepath src
src/main/Main.java
```

What to compile

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 38

Executing the mail system (2)

```
> cd ... package_project
```

Root directory of binaries

```
> java -classpath bin main.Main
```

Main class (in package main)

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 39

J2SE Naming and Versioning

Platform name

2nd generation

- J2SE = Java 2 Platform Standard Edition

Products under the platform

- JDK = J2SE Development Kit
 - Tools: Compiler, Javadoc, ...
 - Java Runtime Environment (JRE)
- JRE = J2SE Runtime Environment

Abbreviation

Full name

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 40

J2SE Version strings

Shell command
> java -version

Update 09
(Bug fixes)

1.4.2_09-b05

Version 1.4.2

b05

http://java.sun.com/j2se/naming_versioning_5_0.html

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 41

Java 5

External numbering

J2SE 5.0 = Java 2 Platform
Standard Edition 5.0 ("Tiger")

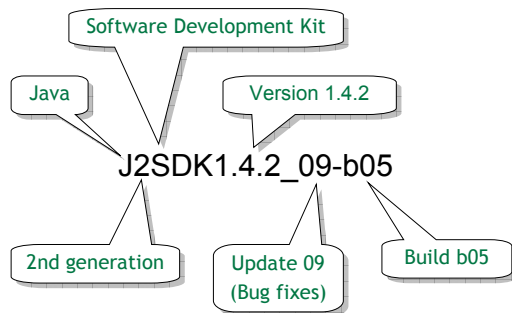
J2SDK1.5.0_xy

Internal numbering

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 42

Example: Directory naming



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 43

Some useful links

- Tools like javadoc, javac, etc.
<http://java.sun.com/javase/6/docs/technotes/tools/>
- Java version naming conventions
http://java.sun.com/j2se/naming_versioning_5_0.html

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 6 44