BASE JAVA COURSE

Lectures: 8 hours with 10 exercises

Requirements: basic HTLM concepts.

If the student does not have this requirement, one extra lecture is necessary (from page 2 to page 8).

Lecture	THEORY	PRACTISE
1	The JRE concept, OOP concept, applications and applets.	`Sample01`: example of Java source
2	The JDK concept, the API documentation, installing the JDK. Compiling and running.	`Sample02`: extending `Sample01`
3	Primitive types: numerics, characters, booleans, Strings. Numeric and logical operators.	`Sample03`: practising primitives `Sample04`: boolean & Strings
4	Basic AWT concepts. The Basic Applet API specifications.	`Sample05`: actions & listeners `Sample06`: Buttons & Textfields
5	Implementation of methods. The `static` concept: the `main()` method and applications. The basic iteration and selection structures: the `for` loop, `if` and `else` statements.	`Sample07`: Hello World! `Sample08`: printing ASCII codes
6	The `switch`, `while` and `do` statements. Ending loops: `break` and `continue`.	Arrays and Vectors

7	Java keywords. Practising with loops and basic programming.	`Sample09`: loops & `drawString()`
		`Sample10`: `Vector` & `Calendar`

8	Handling exceptions: the `try()` … `catch()` syntax. Managing applets: `init()`, `start()` and `stop()` and `destroy()`.	`Sample11`: the applet life cycle `Sample12`: a simple game (catch the oval)

Examinations

1. `Moving`: moving objects on the applet surface: moving with `setLocation()`.
2. `Moving02`: inherithing from Moving and ovverriding: moving with `drawString()`.
3. `Handling`, `Foo1`, `Foo2`: handling exceptions: reading the `stackTrace()` information.
4. `MyPrinter`: printing using the `Painter` and `Printable` objects.

1. *Introduction (1h)*

- Java Run Time (page 11); Structured Programming and OOP (page 1); Objects (page 10)
- Applications and Applets (page 25); Applets (page 9)
- The Source File (page 12) ; Convention (page 16)
- *First example* : testing the **`Sample01`** applet in the browser.

2. *Coding (1h)*

- Analysing the code of the **`Sample01`** applet.
- Browsing the Sun DC-ROM's: JDK, Forte® for Java, API Documentation.
- *Exercise 1* : install the JDK on your PC, compile the **`Sample01`** applet and run it.
- *Exercise 2* : write the **`Sample02`** applet. This applet must extend the **`Sample01`** applet with some new features:
  1. The text showed in the label which follows the mouse is different.
  2. The **`Sample02`** applet adds two more labels on the applet, named `one` and `two`. These labels do not move from their initial location.
  3. These two labels must have different font sizes.

  Note: when compiling this applet, take care to specify the necessary classpath !

3. *Primitives (1h)*

From page 17 to page 23, including page 26.

Exercise 3: write the **`Sample03`** applet, which must contain following code :

- Declaration of integer types, floating types and char type
- Definition of values for Float and Double (testing the "F" and "D" literals)
- Definition of a char type using Escape code
- Casting from char to byte and from long to int
- Experiment the operators "+ +" and "- -"

Exercise 4: write the **`Sample04`** applet, which must contain following code :

- Declaration of Boolean types and String types
- Experiment the logical operators "= =" and "! ="
- Experiment following String's methods: **`toUpperCase()`**, **`length()`**, **`charAt()`**.

4. *Basic AWT (1h)*

The package concept (page 24). Browsing the Basic Applet API Specifications document and the **`Sample05`** exercises.

Exercise 6: write the **`Sample06`** applet. This applet must contain 2 input **`TextField`**'s (*input1* and *input2*), 1 **`Button`** and 2 output TextField's (*next* and *half*):

- 1st time button is pressed the applet gets: `next = input1 + 1`; `half = input2/2`;
- 2nd time button is pressed the applet gets: `next = next + 1`; `half = half/2`;

5. Applications and conditional statements (1h)

Review of page 25. Then page 39-40 (emphasise "Static" concept). Short overview of the "main" method (page 41). First example of Java Application: the `Sample07` class ("Hello World!").
The "for" loop and the "if … then … else" structure (1st part of page 27).
Exercise 7: write the `Sample08` application, which must print of screen the list of the ASCII characters from 1 to 128 associated to even numbers (i.e.: 2, 4, 6, 8 etc.).

6. Loops, Arrays and Vectors (1h)

The "switch" statement, the "while" loop and the "do" loop (2nd part of page 27).
Ending loops with the "continue" statement or with the "break" statement (page 28).
Arrays and Vector (page 29-30).

7. Java Keywords (1h)

From page 13 to page 15. Introduction to the following exercises:

Exercise 8: write the `Sample09` applet, which must:

- It offers three buttons having labels "WHILE", "DO-WHILE" and "SWITCH".
- When pressing one of these 3 buttons a loop of that type is executed.
- Any loop will be "4 steps long" (i.e. the "switch loop" must consider 4 different cases).
- The *n*-step of each loop must draw a log-string on the applet's surface, this string must contain the message "Step N. <*n*>: PERFORMED". Each loop must use a different colour for this string, this can be achieved using the **`drawstring()`** method of the **`Graphics`** class.
- The 2° step of the "while" and "do" loops must activate a **`break`** statement or a **`continue`** statement.
- The "switch" codeblock must select the "case" option by a random choise (from 1 to 4).

Exercise 9: write the `Sample10` application, which allows the user to choose between four options:

- option 1: returning on the screen the current date (using **`GregorianCalendar.get()`** method).
- option 2: inserting the current time into a **`Vector`** named "history".
- option 3: print all the data contained in the vector "history" (and the vector's **`capacity`**).
- option 4: quit the application.

8. Conclusions (1h)

Handling exceptions, from page 43 to page 44.
Analysing the code of the **`Sample11`** applet, which counts and shows how many times the following methods are called by the browser: **`init()`**, **`start()`**, **`stop()`**, **`paint()`** and **`destroy()`**.
Moreover, the applet must contain a Button which simply adds a Label when pressed. Discussing and testing in which way following methods are useful to show the new Label: **`invalidate()`**, **`validate()`** and **`repaint()`**.

Exercise 10: write the **`Sample12`** applet. This applet implements a very simple game. A black oval (use **`graphics. FillOval()`**) moves randomly on the applet, the player must catch it by clicking over it with the mouse. Some hints:

- the oval must move only when the player moves the mouse !!!
- the game ends after the first try (one click by the user) and declares if the player won