Tutorial 5: Logging, XML processing, Java 2D

1. Tutorial Objectives

There are three levels of difficulty of the tutorial exercises marked with asterisks (*) as follows:

* Easy ** Medium *** Hard

It is important to do the tutorial exercises in order as they often rely on a previous question to be completed first. Hints will be marked in italics.

You are expected to be able to finish the easy and medium exercises without any help. However, never hesitate to ask the demonstrator if you have any questions.

2. Exercises

* Exercise 1 Java Logging APIs

Till now, we have been using *System.out.println()* in order to output results of the running application and information on the execution of the code to the user.

As a general rule, the use of *System.out.println()* for outputting diagnosis information is a bad practice unless it's a small java program. This is because you cannot easily turn it off/customize it (a lot of time would be wasted creating/deleting the *System.out* messages).

Today, we will learn the Logging APIs; A more powerful way to capture information about failures in the program execution, configuration errors or bugs. The *java Logging APIs* are part of the *java.util* package. They output well-formatted messages to the user of the java program.
The core package includes support for delivering plain text or XML-formatted log records to memory, output streams, consoles, and files. Furthermore, Logging Levels (SEVERE, WARNING, INFO, CONFIG, etc) can be used to indicate the importance of your information.

Use the following tutorial in order to learn how to use the Logging APIs: http://www.vogella.com/tutorials/Logging/article.html

(1) Find your solution to exercise 5 in tutorial 1 (Virtual chessboard) and try to include logging statements in the program. Explore and use some of the different log levels (severe, warning, info, etc).

** Exercise 2 XML Processing

Complete the following tutorial on Java XML processing, but please note that only the four sections (Java XML Tutorial, Java DOM Parser, java SAX Parser and java StAX parser) are mandatory to cover:
http://www.tutorialspoint.com/java_xml/

(1) Write a *graphics program* that allows a user to load an XML file from the disk using a *file chooser* GUI component. Once the file is open, its content is displayed in a java *text area* GUI component. A sample XML file can be found on moodle 'Books.xml'.

(2) Add a *verify* button in your GUI so once the xml file is loaded, it is parsed and if the file is not well-formatted XML, the errors are
   a. logged to the console using the Logging API
   b. displayed in red on your GUI. For example in an additional text area added to your graphics program.

   Note: Use only *one* of the parsing APIs learned in exercise 2.

(3) Create a *save* button in your GUI so when a user edit the opened XML file and clicks the save button, the XML file is saved with the new content.

   Note: An edited XML file is saved only if it has been verified and no error has been detected in its new content.

*** Exercise 3 Java 2D API

Complete the following tutorial on java 2D API. Please note that the section on "Advanced Topics in Java2D" is not mandatory:
https://docs.oracle.com/javase/tutorial/2d/

Use java 2D to animate three balls on a panel.

(1) The first ball moves continually on a horizontal line from right to left and from left to right.

(2) The second ball moves continually on a vertical line, up and down.

(3) The third ball moves randomly on the panel.

(4) Create a text field on your GUI that continually displays the coordinates of the centre of the three balls during movement.

(Optional) When the balls collide a message is displayed with the identifier of the balls in collision (ball 1 and 3 for example).

Note: Make sure the balls are of different colors
Hint: You can use the *javax.swing.Timer* class

~~~ ooo ~~~
~~~