TP1

DE52

stephane.galland@utbm.fr

Introduction to Eclipse and Implementation of a Stack with Software Engineering Principles

00 | Objectives

Goals of this labwork session are to introduce Eclipse, which is the reference IDE for DE52, and to create a simple data structure, a stack, using the standard software engineering principles.

01 | Exercises

01.1 Introduction to Eclipse

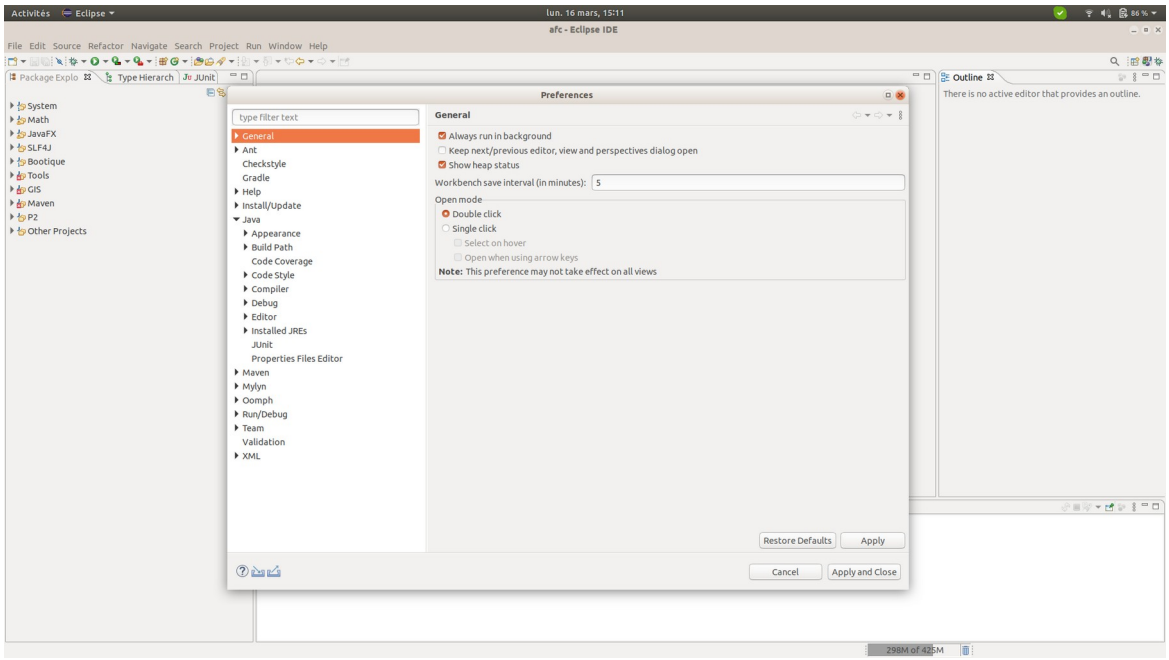
To Do:

- Install Java JDK 21 or higher
- Install Eclipse 2024-12
- Launch Eclipse
- Following the tutorial : https://perso.telecom-paristech.fr/bellot/CoursJava/tps/tp1.html

01.2 Configuration of Eclipse for DE52

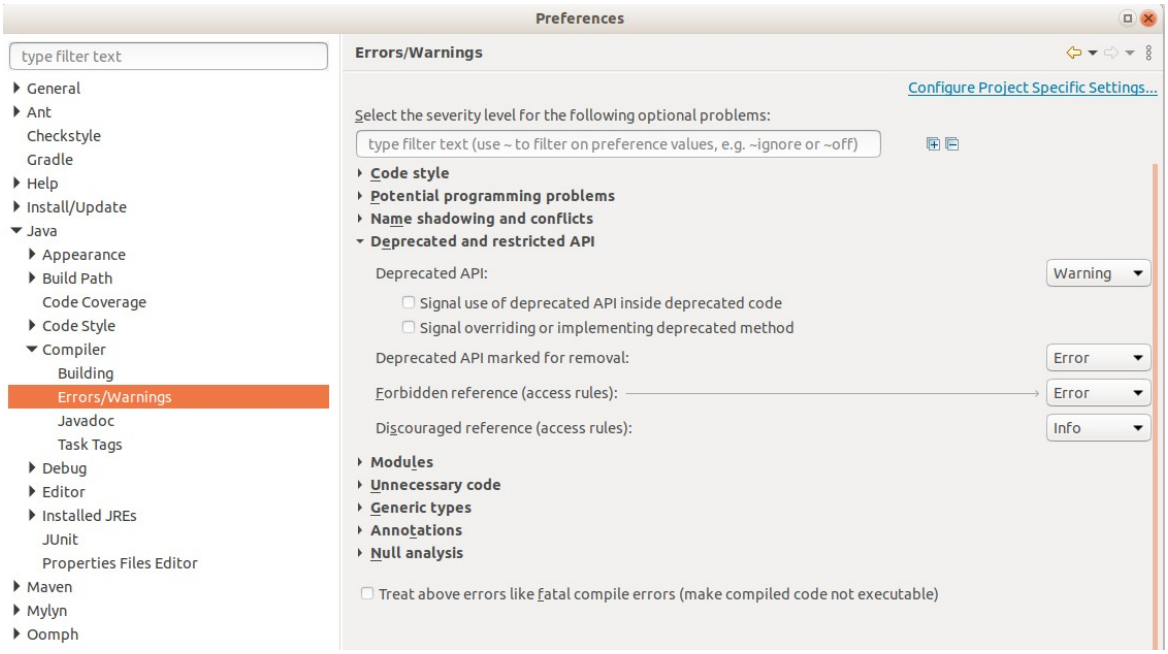
The goal of this section is to configure Eclipse in order to have a maximum of feedbacks related to the warning and error messages, that correspond to the best practices in software engineering :

- Launch Eclipse
- Open the dialog box « Preferences » : Menu « Windows » / « Préférences »



- Open the section « Java » / « Compiler » / « Errors/Warnings »

- On the right part of the window, you could see the list of warning and messages that are recognized by the Java compiler (Eclipse Java Compiler). Each error message is associated to a error level (error, warning, ignore).



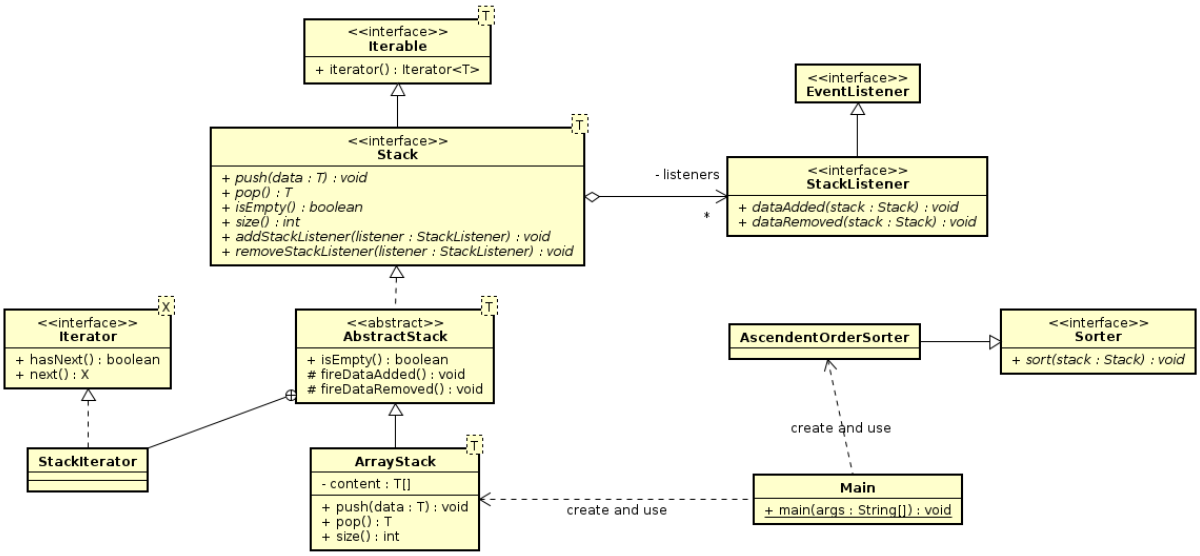
- Open each part (e.g. « Code Style ») and ensure that all the errors have the error level « Error » ou « Warnings ». You should not use any more the « Ignore » level.
- Open the section « Java » / « Compiler » / « Javadoc », et do the same configuration task.
- Save the configuration

01.2 Implementation of a stack

A stack is a data structure that allows data to be pushed and popped.

A stack can contain data of a type specified as a generic type.

The UML class diagram below shows you the structure and relationships between the types to be implemented at the end of the lab.



The 'EventListener', 'Iterable', and 'Iterator' interfaces are already provided by the Java API. The other interfaces and classes must be implemented by you.

You should adhere to the basic principles of programming inspired by software engineering best practices, namely:

1. Use clear names in English, as your code must be readable by any developer, regardless of their native language.

2. Correct the code so that there are no errors or warnings, as each error or warning indicates an anomaly related to good programming practices.

3. Write clear documentation in English (you will have warnings to help you).

4. Apply design patterns as much as possible: https://en.wikipedia.org/wiki/Design_Pattern

Work to be done:

A) Stack Structure

- Write the Stack interface representing a stack.
    - NOTE: It is forbidden to use the Stack interface already provided in the Java API.
- Write the AbstractStack class representing a common implementation for all different stack implementations. Why is it abstract?
- Write the ArrayStack class that implements the principles of a stack using an array to store the elements in the stack.

B) Design Pattern: Observer

- Write the 'StackListener' interface representing an observer on a stack. The two declared functions correspond to the two observable events in a stack.
- Modify the 'AbstractStack' and 'ArrayStack' types to store the list of observers and notify them when the stack is modified.

C) Design Pattern: Iterator

- Write the iterator for the stack, 'StackIterator'. An iterator's role is to sequentially return the list of data in the stack without modifying the stack's content (see the documentation for 'Iterator').

- Modify the 'Stack' interface and the 'AbstractStack' class to integrate the iteration function on the stack.

C) Optional Work: Stack Sorting Tool

- Write the 'Sorter' interface and the 'AscendingOrderSorter' class, representing a tool for sorting data in a stack and an implementation that sorts in ascending order, respectively.

D) Test Program

Write a 'main' function that performs the following actions:

1. Create a stack of integers.

2. Add a random number of random integers to the stack.

3. Display the contents of the stack.

4. Sort the stack.

5. Display the contents of the 'sorted' stack.

AT THE END OF THE LAB, SEND A ZIP FILE CONTAINING THE JAVA CODE YOU PRODUCED DURING THE LAB