Eclipse IDE

www.eclipse.org

Jörg Steffen,DFKI

steffen@dfki.de

15.11.2022

What is the Eclipse IDE?

- IDE = integrated development environment
- Built atop Eclipse Platform
- Free and open-source
- Most popular free Java development environment
- Helps programmers write and maintain Java code
- Takes care of translating Java sources to binaries
- Tool support for Git, Maven, …
- And much much more...

Current Version

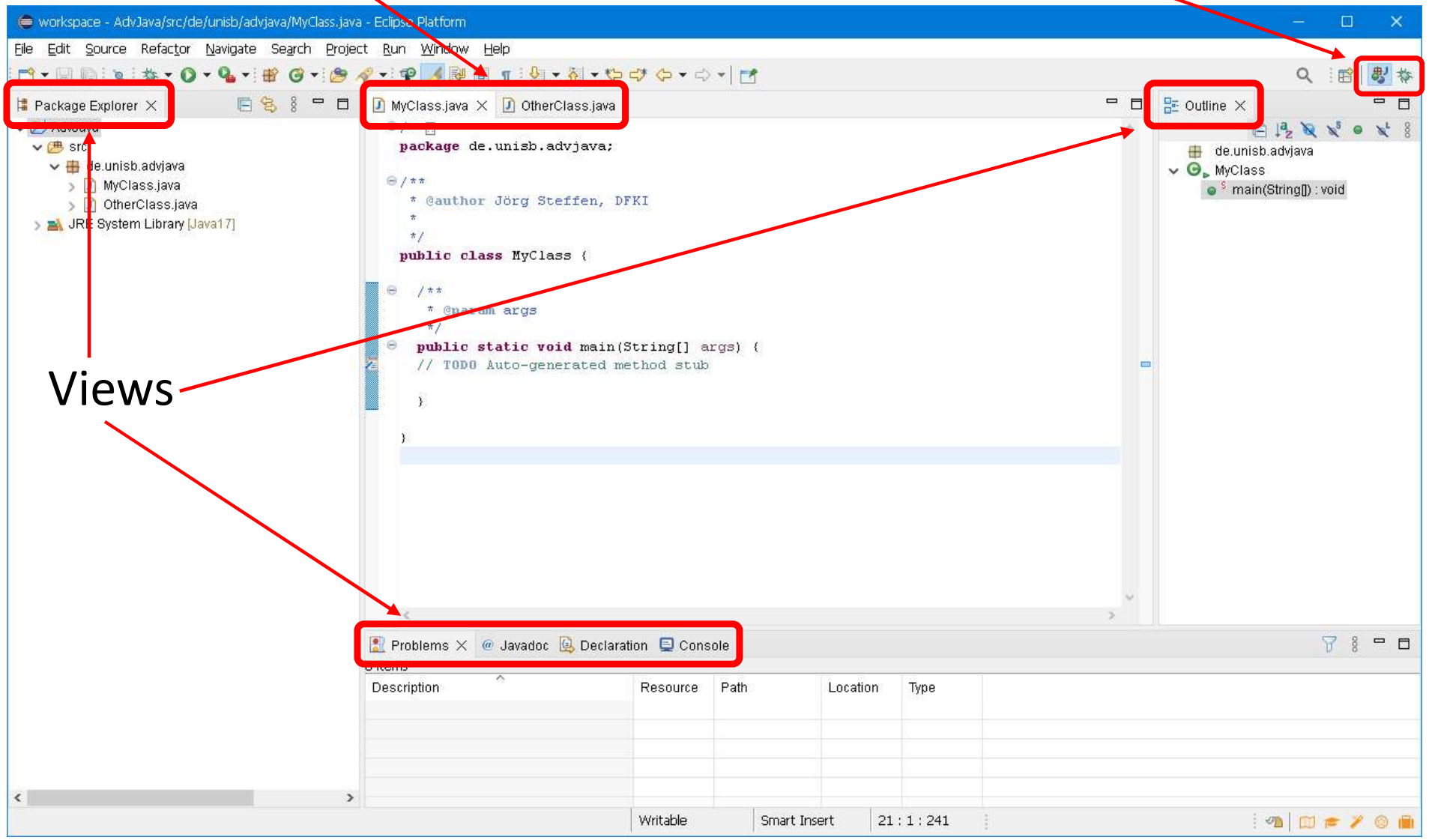
- Current version is the 2022-09 (4.25)

- Recommended version:
  - Eclipse IDE for Java Developers

The Eclipse Workbench

- Editors
  - Depending on the type of file that is being editing, the appropriate editor is displayed in the editor area
- Views
  - Support editors and provide alternative presentations or navigations of the information in the Workbench
- Perspectives
  - Defines the initial set and layout of views and editors in the Workbench window

The Eclipse Workbench



Creating a Java Project

- File -> New -> Java Project
- Set name and location
- Disable module
- Click ‘Next’
- Define the build settings, e.g. make external jars available for the project (not required now)
- Click ‘Finish’

Creating a Package with a Class

- In the package explorer, right-click on ‘src’ and select ‘New Package’
- Name the package, e.g. de.unisaar.jfap.ex1
- In the package explorer, right-click on the new package and select ‘New Class’
- Name the class and optionally change the modifiers, interfaces, etc.
- Check ‘public static main(String[] args)’ to create the stubs for the main method

Running a Java Program

- Write some code in the main method that prints the first given argument:

```
if (args.length > 0) {
  System.out.format("The first arg is %s%n", args[0]);
}
```

- Select Run -> Run Configurations...
- Chose ‘Java Application’, right click it and select ‘New Configuration’
- Click the ‘Arguments’ tab and enter some program arguments: **`arg1 arg2 arg3`**
- Click ‘Apply’ and ‘Run’ to run the program
- Notice that a new view ‘Console’ is opened in the workbench that shows the output of the program
- To run the program again, select the class in the package explorer and press Ctrl-F11

Configuring Eclipse

Coding Standards Tools

- Java compiler warnings
  - Show warnings for certain compiler problems in the Java source editor
  - Default warnings are ok
- Javadoc warnings
  - Show warnings for documentation problems
- Java code formatter
  - Automatically formats your code according to a predefined profile
  - Use Google code formatter
- Checkstyle plugin
  - Allows fine grained definition of a coding standard
  - Shows warning in the Java source editor if coding standard is not met
  - Use Google checkstyle profile
- Goal: write code without warnings

Google Code Formatter

- https://github.com/google/google-java-format
  - Go to release page and download google-java-format-eclipse-plugin-1.13.0.jar
  - Copy to Eclipse plugin folder and restart Eclipse

- **`Java/Code Style/Formatter`**
  - Change Formatter implementation to ‘google-java-format’
  - Create dummy profile for proper indentation when editing
    - Tab policy: spaces only
    - Indentation size: 2

Google Code Formatter

- Fix for Eclipse running with Java >= 16
  - Add these lines at the end of eclipse.ini

```
--add-exports=jdk.compiler/com.sun.tools.javac.api=ALL-UNNAMED
--add-exports=jdk.compiler/com.sun.tools.javac.file=ALL-UNNAMED
--add-exports=jdk.compiler/com.sun.tools.javac.parser=ALL-UNNAMED
--add-exports=jdk.compiler/com.sun.tools.javac.tree=ALL-UNNAMED
--add-exports=jdk.compiler/com.sun.tools.javac.util=ALL-UNNAMED
```

Checkstyle

- Help → Eclipse Marketplace...
- Search for *checkstyle*
- Select *CheckStyle Plug-in 10.2.2*
- Press *Install* and follow the instructions
- Restart Eclipse
- Open settings: Windows → Preferences
- Select **`Checkstyle`**
- Make sure Google Checks is the default configuration
- In the project properties, you can now activate Checkstyle for this project

General Settings

- **`Java/Installed JREs`**
  - Check if Java 17 is available
  - If not, download from OpenJDK releases and add it
- **`General/Editors/Text Editors`**
  - Set print margin to 100 and make it visible
  - Set tab with to 2 and use spaces
- **`Java/Code Style/Organize imports`**
  - Sort imports alphabetically
  - Remove all, add new *

General Settings

- **`Java/Editor/Save Actions`**
  - Format source code and organize imports
  - Additional action: remove trailing whitespaces
- **`General/Appearance`**
  - Activate ‘Dark’ theme if it suits you ;-)

Tips & Tricks

- Content Assistant
  - Suggests completions for partially entered strings
  - Lists all methods for a class
  - Expands templates → **`Java/Editor/Templates`**
  - Ctrl-Space
- Add Import
  - Creates an import statement for the class under the cursor
  - Ctrl-Shift-M
- Organize Imports
  - Adds all necessary import statements
  - Removes all unneeded import statements
  - Ctrl-Shift-O

Tips & Tricks

- Quick Fix
  - Suggests a solution to a problem or other helpful operations
  - Indicated with an electric bulb on the left side
  - Ctrl-1
- Show Javadoc for class/method as tooltip
  - Click on class/method name in editor and press F2
  - Alternative: Shift-F2 opens doc in browser tab
- Jump to source code of class/method
  - Ctrl-click a class or method name in the editor

Tips & Tricks

- Highlight occurrences
  - Click element to mark its occurrences in the current file
  - Distinguishes read and write occurrence
- Correct Indentation
  - Mark code and press Ctrl-I
- Format Code
  - Mark code and press Ctrl-Shift-F
- Add Javadoc comment to class or method
  - Alt-Shift-J
- Generate getters and setters
  - Context menu: Source → Generate Getters and Setters...

Tips & Tricks

- Rename classes/methods/variables
  - Also updates references
  - Alt-Shift-R
- Find/Replace strings in code
  - Ctrl-F
- Incremental find
  - Ctrl-J
- Toggle Comment
  - Mark code and press Ctrl-/ (Windows), Esc Ctrl-C (Linux)

Tips & Tricks

- Revert back to or compare with a previous file version
  - Right-click method/class and chose
    ‘Compare/Replace With Local History...’
- Show class hierarchy
  - Hierarchy containing the super and sub classes of a selected class
  - F4
- Show call hierarchy of method
  - List all call occurrences of a selected method/field
  - Ctrl-Alt-H

Eclipse Debugger

- Double-click on marker bar to set breakpoints
- Run code in debug mode
  - Run → Debug As → Java Application
  - Use F11 to debug with last run configuration
- When code stops at break point
  - Step Into (F5): step into call
  - Step Over (F6): step over call
  - Step Return (F7): return to caller
  - Resume (F8): continue running (until next breakpoint is hit)
- Check variable values in the ‘Variables’ view