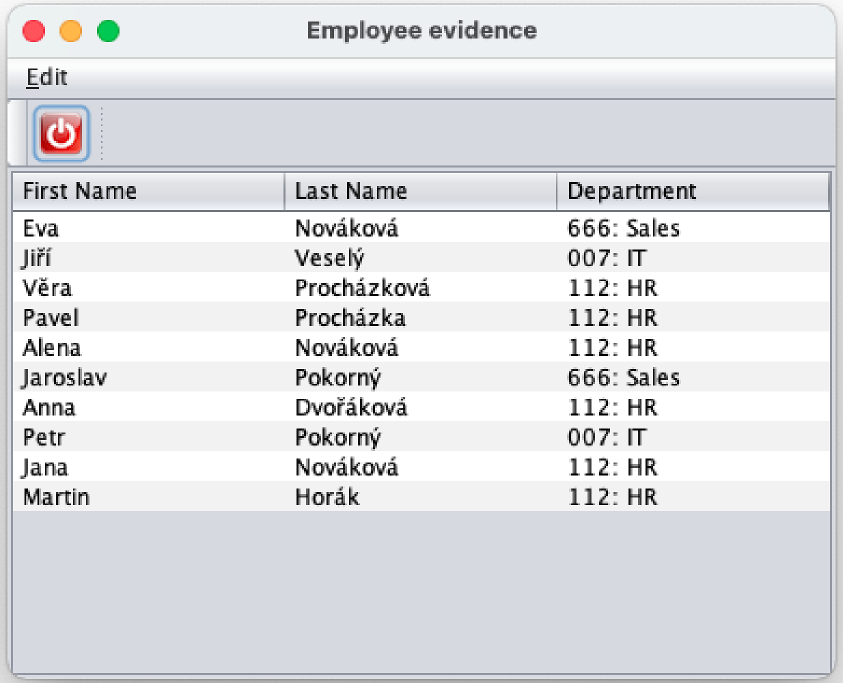


First Name	Last Name	Department
Eva	Nováková	666: Sales
Jiří	Veselý	007: IT
Věra	Procházková	112: HR
Pavel	Procházka	112: HR
Alena	Nováková	112: HR
Jaroslav	Pokorný	666: Sales
Anna	Dvořáková	112: HR
Petr	Pokorný	007: IT
Jana	Nováková	112: HR
Martin	Horák	112: HR

PV168

Employee Records

Seminar Task 1

- Place **Add**, **Edit**, and **Delete** operations to both menus and the toolbar
- Did you find it difficult?
    - There was a straightforward solution
- So what was the problem then?
    - The easy solution added enormous amount of duplicity
- What would help us to eliminate the duplicity?
    - Encapsulate the knowledge about the details and reuse it

Seminar Task 2

- Add **Gender** and **Age** to `Employee` and then to the table
- Did you find it difficult?
  - It seemed to be easy, but it was actually very difficult to make it right
- Where did you screw up the most?
  - Using `String` as the representation of gender is bad
  - Using `int` as the representation of age is bad too
- Always represent fixed set of values using `enum`
- Always prefer constant values (e.g. *birth date* instead of age)!

Seminar Task 3

- Reorder columns in the table
- Did you find it difficult?
    - It could be done easily, but perhaps you didn't like the solution
- What was the ugly part?
    - Duplicated `int` literals spread throughout 5 methods
- What would help us to eliminate the duplicity?
    - Maybe the solution is similar to that of **Task 1**

Seminar Task 4

- Enable/disable **Edit** and **Delete** operations based on selected rows
- Did you find it difficult?
  - It was very tough for majority of you due to multiple reasons
- What were the most problematic parts?
  - How to determine the number of selected rows in the table
  - Where to get all the operations to be enabled/disabled
- Many of you parsed "the DOM" of the frame to get the operations
  - **Don't do this!** You can do better as you're creating all of them

Where did the pain come from?

- You were solving high-level problems with too much low-level tools
- You were missing the appropriate *abstractions* for your tasks!

Abstraction

> “ The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context. [1] ”

[1]: Guttag, John V. (18 January 2013). Introduction to Computation and Programming Using Python (Spring 2013 ed.).

Real Life Abstractions

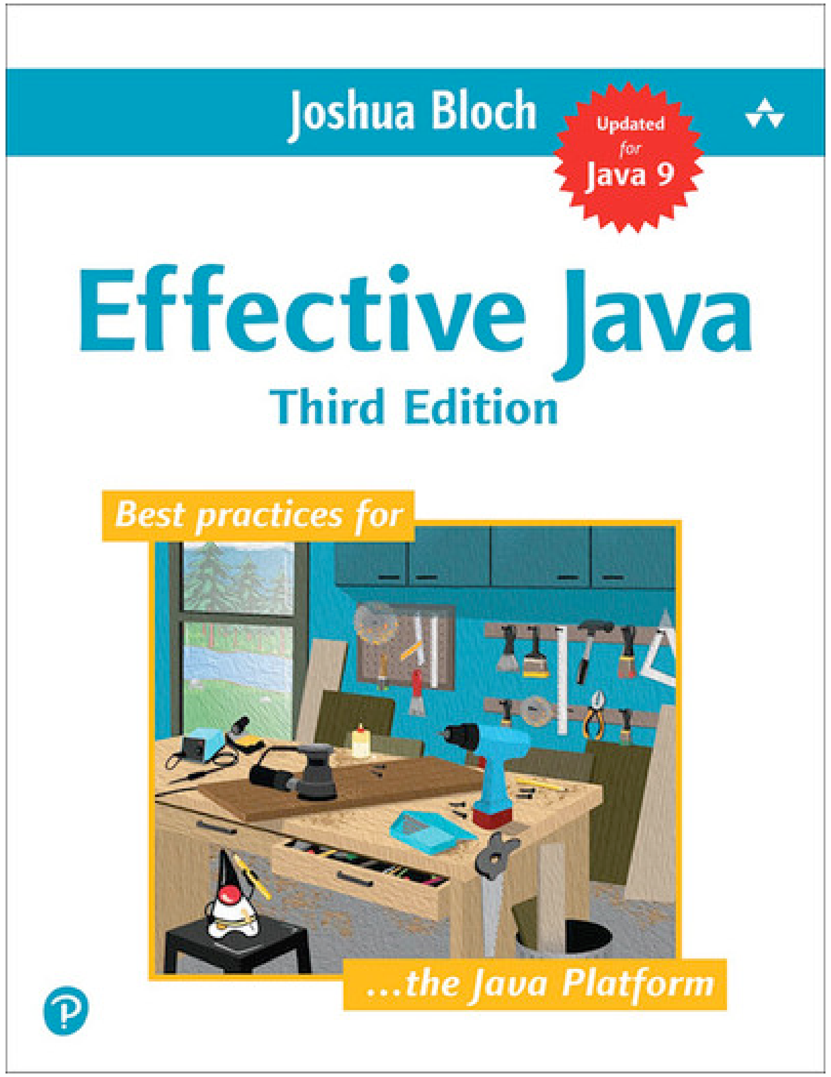
- Cars
  - Steering wheel
  - Pedals (brake and throttle)
- Electronic devices
  - Plug
  - Socket

Employee Record Abstractions

- `EntityDialog`
  - Shielding from dialog elements positioning
- `ComboBoxModelAdapter`
  - Adapting `ListModel` to `ComboBoxModel`
- `DepartmentListModel`
  - `ListModel` for departments
- `EmployeeTableModel`
  - `TableModel` for employees

Abstractions you were missing

- Java Swing `Action`
    - Substitute for menu items and toolbar buttons
- Our own `Column` (for Swing `TableModel`)
    - Associated Code Smell
        - Repeated Switches



Recommended reading

Effective Java (Third Edition)

Joshua Bloch

Addison-Wesley Professional, 2017

https://www.amazon.com/dp/0134685997

Effective Java

- Book not only about Java
  - Almost half of the book is about OOP in general
  - Present since 2001 (the first edition)
  - With code examples specifically in Java
- Items are heavily cross-referenced
  - Start reading anywhere you like

Josh Bloch (the author)

- Java architect between 1996 and 2004
- Designed a lot of cool stuff for Java
    - The Java Collections Framework (Java 1.2)
    - Assertions, amendments to `Throwable` (Java 1.4)
    - Generics, Annotations and `Enum` (Java 1.5)
    - try-with-resources and `AutoCloseable` (Java 1.7)

Item 62

Avoid strings where other types are more appropriate

- Strings are poor substitutes for
  - other value types (boolean, numbers, dates, ...)
  - enum types
  - aggregate types
- Avoid the natural tendency to represent objects as strings when better data types exist or can be written

Strings (Item 62)

- Associated Code Smell
  - Primitive Obsession
- In Employee Records
  - enum `Gender`
    - See alo **Item 34**: *Use enums instead of `int` constants*
      - when you need a set of constants known at compile time
      - `enum` is designed for evolution in time (adding constants)
      - enums in Java are **objects**, not C/C++ `int` constants!