PV 168
Seminar 11

Agenda

- Project: Next Steps
- Project: Third Milestone
- Seminar task
- Seminar task reflection

Project: Next Steps

- Fix all issues found during the Second Milestone review
  - Problems in the GUI (usability)
  - Code quality
- Cover your code with tests (to enable further refactoring)
  - Everything but GUI
- Make sure that all DB operations are performed out of *Event Dispatcher Thread*
- Translate your application into two languages
  - English (as default)
  - Czech/Slovak/German/French/Russian/Filipino/Cebuano

Project: Third Milestone

- Implement the fully functional application
  - In the scope agreed with the customer
- Deadline: **8. 1. 2021 23:59**
  - The **master** branch of your repository at https://gitlab.fi.muni.cz contains complete application
- If you have any questions, ask assigned seminar tutor (the one who has customer role for your project)

Seminar task

- You will be now split to teams consisting of four students
- Create and push the branch (see the next slides)
- Work on the tasks in the specified order (see the next slides)
- You have **60 minutes** to solve all the tasks
- If you need any help, **Ask for Help in Zoom**

Working with branches in Git

- Clone the project https://gitlab.fi.muni.cz/pv168/employee-evidence
- Create new local branch
  - based on commit **81d72ee3** (current head of master branch)
  - with name **week11-group0X-roomY** (0X is seminar group, Y is breakout room number)
  - If you from group PV168/01 and breakout room 3, branch name is week11-group01-room3
- Pavel Hrdina or Petr Adámek will give you write access to employee-evidence repository
  - Only one person per breakout room - the one who will use the computer for coding
  - Petr or Pavel will visit your breakout room to ask who is this person
  - In the meantime work on other tasks
- Push the branch to origin repository

Seminar task

1. Tasks #1 - #4 (Thread Quiz)
   - Look at the code, discuss it with your team partners and answer these questions
     i. Is this code thread-safe?
     ii. Is the scope of the synchronization appropriate?
   - If answer to any question above is no, explain why.
   - Write down your answers to **/src/main/java/cz/muni/fi/pv168/threads/ThreadQuiz.txt**
   - Commit your answers (one commit for all 4 tasks is sufficient)
2. Tasks #5 and #6 (Multi-thread Counter)
   - Commit each task separately!
3. Task #7 (Delete Operation in Background Thread)
   - Commit your solution
   - Don’t forget to push

Task #1: Thread Quiz 1

```
final class Counter {

    private static int currentValue = 0;

    public synchronized int next() {
        return ++currentValue;
    }
}
```

Task #2: Thread Quiz 2

```
final class Counter {

    private static final Object LOCK = new Object();

    private int currentValue = 0;

    public int next() {
        synchronized(LOCK) {
            return ++currentValue;
        }
    }
}
```

Task #3: Thread Quiz 3

```
final class Counter {

    private Integer currentValue = 0;

    public int next() {
        synchronized(currentValue) {
            return ++currentValue;
        }
    }
}
```

Task #4: Thread Quiz 4

```
final class ThreadSafeContainer {

    private final List<String> rows =
        Collections.synchronizedList(new ArrayList<>());

    public void addRow(String row) {
        rows.add(row);
    }

    public synchronized void printRows() {
        for (String row : rows) {
            System.out.println(row);
        }
    }
}
```

Task #5: Multi-thread Counter

Fill in the implementation of `Counter` class to spawn 3 threads. These threads cooperate on generating **unique numbers** from **0** to **50** and printing them on standard output (not necessarily in the correct order) in the following format:

```
Thread 2: 1
Thread 3: 2
Thread 1: 0
Thread 2: 3
...
Thread 2: 47
Thread 3: 50
```

Task #6: Multi-thread Counter (in order)

- Commit the implementation of Task #5.
- Change the implementation to print all numbers in the correct order:

```
Thread 1: 0
Thread 2: 1
Thread 1: 2
Thread 3: 3
...
Thread 2: 49
Thread 3: 50
```

Task #7: Delete Operation in Background Thread

1. Modify the application not to call **`EmployeeDao.delete(...)`** in *Event Dispatcher Thread*, but in some background thread.
2. Resolve the problems caused by concurrent execution of multiple delete operations.

Link to slides

https://is.muni.cz/auth/el/fi/podzim2020/PV168/um/seminare/PV168-seminar-11.pdf

Conclusion

Any questions?