Concurrency:
State Models & Design Patterns

Practical Session

Week 09

Assignment 08

Discussion

A08 - Exercise 1

Answer the following questions:

a) Why are servers (e.g. web servers) usually structured as thread-per-message gateways?

Servers (e.g. web servers) usually must handle many requests, some of them quite resource intensive. Therefore, it makes sense to let each request (service) to be handled by a separate helper thread, as long as the overhead of thread construction does not exceed the helper costs.

b) What are condition objects? Name at least one advantage and one disadvantage of using condition objects.

Condition objects are a programming pattern used to encapsulate waiting and notification strategies in the context of guarded methods.

Advantage: By isolating conditions, one can often avoid notifying waiting threads that possibly cannot proceed given a particular state change (better efficiency)

Disadvantage: It may lead to “design overhead” in some situations

A08 - Exercise 1

Answer the following questions:

c) Why does the SimpleConditionObject from the lecture not need any instance variables?

The Condition class does not need to implement any state, rather it specifies synchronization behaviour.

d) What are “permits” and “latches”? When it is natural to use them?

Basically, permits are special condition objects that behave similarly to counting semaphores, somewhat like counting semaphores implemented as condition object.
A latch is a condition that is initially false, but once set to true, it remains true.

A08 - Exercise 2

Consider the provided sample code. In both cases, several client threads request a server to compute fibonacci numbers.

a) **Question**: Which implementation would you prefer for this kind of problem? Is there any considerable difference at all? Justify your answer!

It depends much on the order of requests which clients are served first. So the future-based solution seems more flexible in this case.

b) **Implementation**: Write a new class FutureTaskExecDemo.java that uses an executor service to compute the future task and to execute the clients, instead of creating explicit new threads. What is the benefit of using executors?

The task is usually submitted to a pool of threads of which one will execute your task.

A08 - Exercise 2

Consider the provided sample code. In both cases, several client threads request a server to compute fibonacci numbers.

c) **Implementation**: Add a time constraint such that the client thread waits for at most a given amount of time for the result.

Solution will be provided as ZIP file.

A08 - Exercise 3

Implementation:

Farmer Napoleon owns a magic chicken called Clarissa who is supposed to lay infinitely many eggs. Napoleon has hoped to dispose an endless source of eggs to build up his egg-imperium. But there is a serious deadlock problem hidden behind the story.

Your task is to resolve this problem in order to let Napoleon continously retrieve eggs from Clarissa. Be careful that your solution is data race free.

Solution will be provided as ZIP file.

A08 - Exercise 4

Thread Speed Evaluation

We provide you with an implementation of a Pi approximation tool. This tool uses the Leibniz formula for Pi with infinite series for approximation. Your task is to observe the characteristics of the threads as well as their calculation speed. **It is very important that you perform multiple runs** before answering the questions below. You can find the sample code on GitHub.

Please answer the following questions:

A08 - Exercise 4

a) What amount of processing cores does the CPU in your notebook have and what's the model / manufacturer of it?

Intel Core i7-4600 CPU, providing 2 physical and 4 logical cores

b) Does the implementation scale well? Please provide concrete runtimes you experienced!

No, it does not scale well. With just one concurrent thread we achieve runtimes for 10,000 iterations on a dual-core notebook between 950 and 3,500 ms, whereas with two concurrent threads we achieve almost identical runtimes.

c) Depending on your results, why or why not does the solution scale well?

It does not scale well, because the implementation produces way too much overhead with the thread management.

A08 - Exercise 4

d) How would you improve the runtime with respect to faster calculations (without changing the algorithm)?

We could group individual operations into much larger chunks and calculate these chunks on different threads.

e) Which algorithm would you recommend as drop-in replacement for the Leibniz formula for faster calculation?

There exist many different algorithms. One of the more promising approaches that supports quite a high degree of parallelism is called “BaileyBorweinPlouffe” formula.

f) Why do the runtimes with identical parameters vary so much?

The Java VM performs various internal optimizations (caching, command reordering, ...) and relies on external threading APIs of the underlaying operating system, that may not perform identical on each run.

Assignment 09

Preview

A09 - Exercise 1

Answer the following questions:

a) What criteria might you use to prioritize threads (list at least 5 different criteria)?

b) What are different possible definitions of fairness (list at least 3 different definitions)?

c) What are Pass-Throughs and Lock-Splitting?

d) When should you consider using optimistic methods (list at least 3 different enablers)?

A09 - Exercise 2

In this exercise you have to **implement** a class that represents graphical objects that consist of an x-coordinate, a y-coordinate, a width and a height (**= rectangle**). The class has to implement methods for:

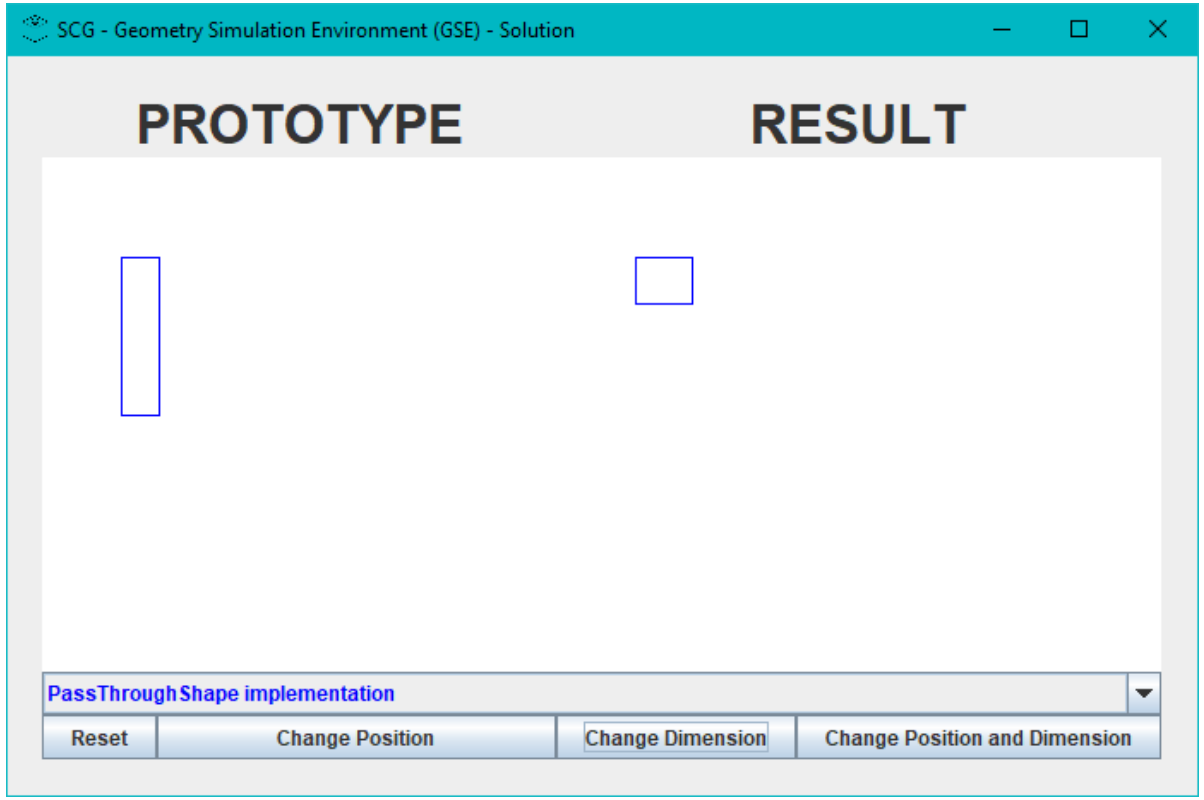
- Increase the x-coordinate by 10% and decrease the y-coordinate by 20% (change position)
- Increase the width by 50% and decrease the height by 70% (change dimension)
- Increase the y-coordinate by 40% and decrease the height by 60% (change position and dimension)

Implement it once using Lock-Splitting and once using Pass-Throughs.

A09 - Exercise 2

SCG Geometry Simulation Environment

(SCG GSE)



A09 - Exercise 3

Answer the following general questions:

a) How do threads waiting in a Thread.join() loop get aware of that thread's termination?

b) How could you optimize the code below?

```
Thread t = new Thread(<insert your runnable code here>)
t.start()
t.join()
```

c) Are String objects in Java mutable or immutable? Justify your answer!

d) Does the FSP progress property below enforce fairness? Justify your answer!

```
progress HeadsOrTales = {head, tale}
```

Lab 02

Sneak Peak

Lab 02 - Overview

Concept

- Eclipse + Java based tasks

- Concurrency code that needs work

- Work in groups of two

- The lab starts at 10:15 and ends at 12:00

- Location: usual lecture room

Grading

- 5 bonus points for 100% correctness

Requirements

- Notebook with power adapter

- Latest version of Eclipse installed

- Latest version of the Java SDK installed

- Optional: a mouse

Allowed:

- **Lecture slides and books**
- **Internet access**

Lab 02 - Workflow

1) Pull from public GitHub repository
2) Fix the four provided sample apps:
   - *Nested monitor*
   - *Concurrent read/write access*
   - *Fairness*
   - *Concurrent resource allocation*
3) Submit your zipped project solutions by mail to *pascal.gadient@inf.unibe.ch*

*You have to attend the lecture to reveal such slides.**

:-)

**Disclaimer:*

The content that has been shown on this slide is irrelevant for the exam.