Concurrency: State Models & Design Patterns

Practical Session

Week 08

Assignment 07

<u>Discussion</u>

A07 - Exercise 1

Answer the following questions about liveness and asynchrony:

a) When should you consider using asynchronous invocations?

- When an object can distribute services amongst multiple clients.
- When an object does not immediately need the result of an invocation to continue doing useful work.

b) In what sense can a direct invocation be asynchronous?

If you invoke helpers directly, but asynchronously without any synchronization, you are able to create asynchronous invocations with direct invocations.

A07 - Exercise 1

Answer the following questions about liveness and asynchrony:

c) What is an “early reply”?

“Early reply” is a feature that allows a host to perform useful activities after returning a result to the client.

d) What are “futures”?

“Futures” is a feature that allows a client to continue in parallel with a host until the future value is needed.

A07 - Exercise 1

Answer the following questions about liveness and asynchrony:

e) When are futures better than early replies?

In general, the idea is to choose futures instead of early replies if it is required that the client has to complete some extra tasks that are not related to the communication partner, while early replies are intended for cleanup and optimization tasks on the host side.

A07 - Exercise 2

Answer the following questions about Java VM and implementation details:

a) Why does the call Thread.currentThread().join() not make much sense in a thread’s concurrent run method?

Because the tread would wait for it’s own death and that forces the thread to remain in the wait state. As these threads do not die on their own, it must be ensured manually that they will be terminated when the main application is closed.

b) Why can you encounter an IllegalMonitorStateException when calling notifyAll() in a code block that is not synchronized?

The method notifyAll() can only be called successfully when the caller has the monitor, because the caller will then give the monitor to the waiting callee that reacts to the notifyAll() statement.

A07 - Exercise 2

Answer the following questions about Java VM and implementation details:

c) What happens with the thread when the code execution gets to the end of the thread’s run method (suppose no loop is involved in the run method)?

   The thread transitions into the ”terminated” state, i.e. is not running any longer, similar to a closed application.

d) Why do some Java apps not terminate, even though the GUI has been closed and custom WindowHandlers have been set up?

   The window handlers supposedly didn’t force threads to exit and the threads were not notified about the requested termination. So the threads will still remain active, thus the application running on the system.

A07 - Exercise 2

Answer the following questions about Java VM and implementation details:

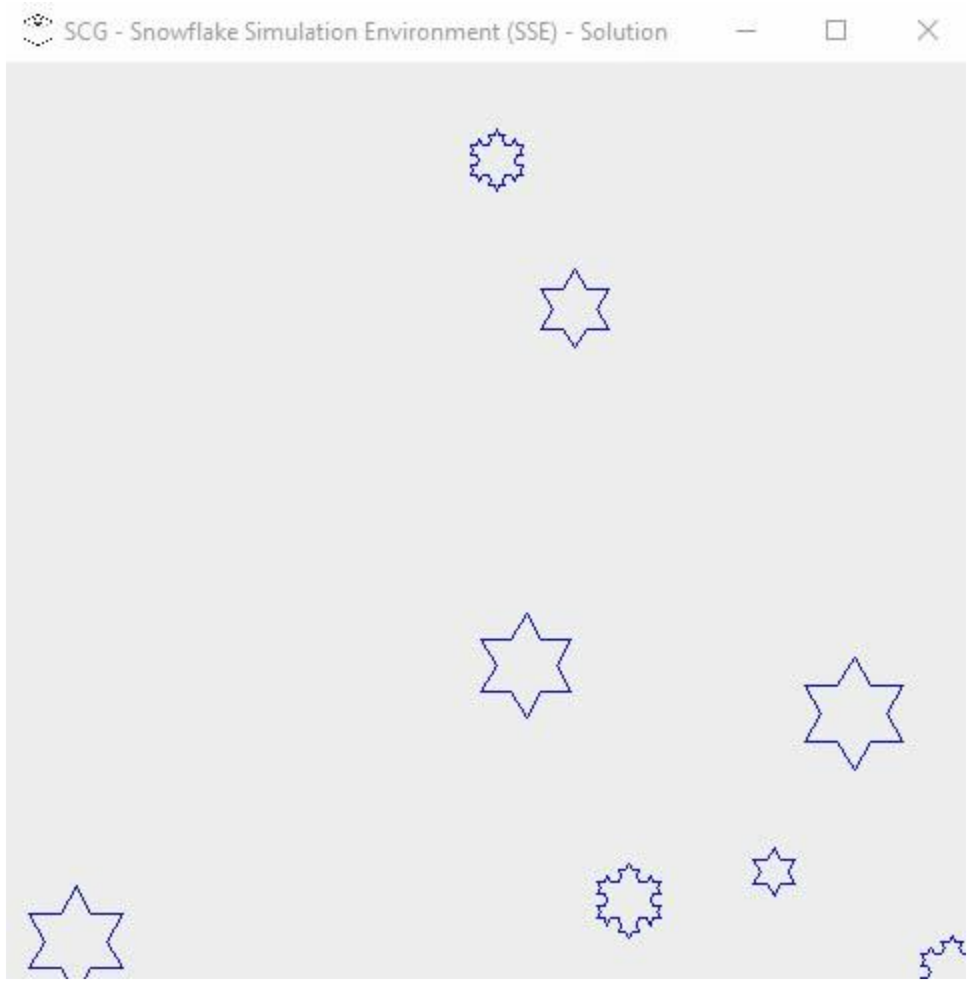
e) Name one reason for using a while loop in a thread’s run method. Justify your selected reason.

Graphical (user interface) draw calls that need to be performed repeatedly. There exist different approaches for this problem, like spawning new threads for every UI update, but they introduce an not negligible maintenance overhead.

A07 - Exercise 3

SCG Snowflake Simulation Environment

(SCG SSE)



Assignment 08

Preview

A08 - Exercise 1

Answer the following questions:

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

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

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

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

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

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.

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?

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

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

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

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

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

*You have to <u>attend the lecture</u> to reveal such slides.**

:-)

**Disclaimer:*

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