Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Mohammadreza Hazhirpasand

Solution

Assignment 09 — 13.11.2019 – v1.0
Fairness and Optimism

Exercise 1 (4 Points)

Answer the following questions:

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

Priority may depend on any of:

- *Intrinsic attributes of tasks (class & instance variables).*
- *Representations of task priority, cost, price, or urgency.*
- *The number of tasks waiting for some condition.*
- *The time at which each task is added to a queue.*
- *Fairness guarantees that each waiting task will eventually run.*
- *Expected duration or time to completion of each task.*
- *The desired completion time of each task.*
- *Termination dependencies among tasks.*
- *The number of tasks that have completed.*

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

- *Weak fairness: If a process continuously makes a request, eventually it will be granted.*
- *Strong fairness: If a process makes a request infinitely often, eventually it will be granted.*
- *Linear waiting: If a process makes a request, it will be granted before any other process is granted the request more than once.*
- *FIFO (First-in First out): If a process makes a request, it will be granted before that of any process making a later request.*

c) What are Pass-Throughs and Lock-Splitting? **Answer:**

- *Pass-Throughs: The host maintains a set of immutable references to helper objects and simply relays all messages to them within unsynchronized methods.*
- *Lock-Splitting: Instead of splitting the class, split the synchronization locks associated with subsets of the state.*

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

- *Clients can tolerate either failure or retries. If not, consider using guarded methods.*
- *You can avoid or cope with livelock.*
- *You can undo actions performed before failure checks:*
  ***Rollback / Recovery**: undo effects of each performed action. If messages are sent to other objects, they must be undone with "anti-messages".*
  ***Provisional Action**: "pretend" to act, delaying commitment until interference is ruled out.*

Exercise 2 (4 points)

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 (i.e., a 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 (use the provided Shape interface of which an excerpt is listed below).

```
public interface Shape {

        public void changePosition();

        public void changeDimension();

        public void changePositionAndDimension();
}
```

You will find additional comments and hints in the code that may help and guide you. The SCG Geometry Simulation Environment is available on GitHub. You can find the project on https://github.com/pgadient/concurrency_e09t02.git. Please submit your project within a zip file. You can easily achieve that by exporting the project with the Eclipse export assistant. **Answer:**

You can find a sample implementation on GitHub at https://github.com/pgadient/concurrency_e09t02_solution.git.

Please clone immediately as the repository will be turned back into a private one in the following days.

Exercise 3 (2 Points)

Answer the following general questions:

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

All threads will call this.notifyAll() before they enter the TERMINATED state. This call is issued by the Java framework itself and therefore it is not visible in the code.

b) How could you optimize the code below?

```
Thread t = new Thread(new Runnable() {
  @Override
  public void run() {
    <insert your code here>
  }
});
t.start();
t.join();
```

Answer:

You could remove the thread instantiation and extract the plain code into an ordinary method without any asynchronity.

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

According to https://docs.oracle.com/javase/tutorial/java/data/strings.html*: The* String *class is immutable, so that once it is created a* String *object cannot be changed. Another way to argue is that* StringBuilder *classes exist in Java and they wouldn't make much sense, if* String *objects would be mutable.*

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

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

Answer:

No, it does not. When a process will choose head *in every run it doesn't violate this progress property, but it won't be fair.*