




## Tutorials for “Formal methods for Java” Christmas Exercises

This exercise sheet is a revision of most of the topics covered in the lecture so far and does not need to be submitted. Your tasks are indicated by  inside a bigger story targeting to verify a (slightly fictional) software system that is contained in the additional material download.

We recently received some emails concerning the topics of the lecture. Hopefully you can help us in answering these questions.

### Exercise 1: JLS and Operational Semantics

We received this email:

Dear Lecturers,

we came across your lecture and had some questions regarding your operational rules. My lead programmer likes to use as few variables as possible and tends to program like this:

```
[snip]
int g;
{
  int i = 0;
  // do something with i
  g = i;
}
{
  int i = 0;
  // do something with i
  g = g + i;
}
[snap]
```

I never understood that code and wanted to use your rules for the operational semantics of Java. Unfortunately, I do not know how to evaluate these curly brackets. Can you help me?

Thank you,

The Bearded.


My reply was only part of the solution:

Dear Bearded,

I do not wonder about this. Neither do we have a rule for blocks of expressions (that is what the curly brackets are), nor do we have a rule for variable declarations. I am a little bit short on time and have to think about the rules for some time. So please be patient.

Regards,

J

 Think about a rule that handles variable declarations in the operational semantics, and a rule that deals with blocks of expressions. I think, at least one component of the state might change when we leave a block.

## **Exercise 2: Writing Specifications**

We received some requests to write specifications.

Dear Lecturers,

at the north pole we have a system to produce nice presents. Since too many exploits and security problems are inherited from languages like C we decided to use Java. Furthermore, Java has a clean semantics and some automatic tools to ensure that a system satisfies its specification. Unfortunately, we did not manage to write the specification for our system. So we kindly ask you to help us out with this.

The system is a resource bounded production system. It consists of several components that I will shortly describe:

1. Every unit of work that needs to be done is represented by an object of class "Task". A task is a prioritized piece of work and, hence, needs some time to build and some amount of "elfin dust" (I don't think you know about that but this is not important). If we do not have enough time or elfin dust left, the task is skipped and cannot be performed. In this case, the "exec" method of class "Task" should throw an "InsufficientResourcesException".
2. The available amount of time and elfin dust is kept in an object of class Statistics. There should only be one element of this class available in the system.
3. An "Executor" executes all tasks according to their priority. Note that we

defined the priority not intuitively, i.e., the lower priority task is executed first. Since some tasks might actually reschedule, the executor keeps a flag that tells tasks whether the executor is currently running, or not.

4. A "Wish" adds multiple wishes for multiple people to the task list. After adding all tasks, it starts the executor if it is not already running.


Can you please write the specification for us without changing the source code. My boss does not allow me to recompile the code.

I attached the source files of the relevant classes. Please note that these classes are intellectual property of the Santa Inc., North Pole.

Thanks,

Jordi

Technical Lead Elf  
North Pole

 Specify the desired properties in JML. There seem to be some hidden assumptions that might be common sense, but are actually needed.

### Exercise 3: Invariants

We were also ask to formulate some invariants.


Hi,

we have heard about different systems to describe invariants (the universe type system, ownership, friendship) but do not know any details about them. We want to formulate the invariant, that if the executor is running, the task queue is not empty. Furthermore, only the executor should be able to remove tasks from the queue and wishes should be the only possibility to add new tasks to the queue.

Additionally, we would like to specify that the available time and elfin dust is always non-negative. This should be obvious, but we do not know how to formalize this invariant. Can you help use?

Thanks,

Jordi

 Explain for both invariants and every system (Universe Type System, Ownership, or Friendship) whether the invariant can be expressed using that system.

## Exercise 4: Sequent Calculus

"The Bearded <santa@clause.np>" wrote:

> One of my colleagues had a question regarding your presentation of sequent  
> calculus. He said that your axiom "close" is actually too liberal.

That is not true.

> He said that the close axiom should only apply to atoms, but not to whole  
> formulas.

You can use structural induction to derive the axiom presented in the lecture from the version of the close axiom you mentioned. Structural induction, in this case, works as follows:

- 1) Base Case: Prove that the close axiom is applicable to atoms.
- 2) Induction Case: Assume the close axiom can be derived for  $F$  and  $G$ . Show that the close axiom can be derived for  $F \text{ op } G$  where  $\text{op}$  is a Boolean operator, for  $\text{not } F$ , and  $Q x. F$  where  $Q$  is a quantifier.

Regards,

J



Assume the close axiom  $\Delta, P \implies \Gamma, P$  is only applicable if  $P$  is an atom. Use structural induction and this assumption to prove that the close axiom presented in the lecture can be derived from the close axiom mentioned above.

## Exercise 5: KeY

We received a request to specify and verify a small function using the KeY prover.

Dear Lecturers,

Could you help us with the attached class file? It extracts the minimum value and the first position it encountered this value from a non-empty array. We tried to specify and verify this class using the KeY prover but failed. Since we are not sure what was wrong we removed the specification. Can you help us in formalizing the desired behavior of the function "min" and come up with an invariant/variant proof that shows total correctness of this function?

Thanks from all computer scientists at the north pole.



Specify the desired behavior of the min function of class MinValue that can be downloaded from the webpage. Give an invariant/variant proof in the KeY system.

**We wish you a merry Christmas and a happy new year.**