J. Hoenicke
J. Christ

18.12.2012
Hand in solutions via email to
hoenicke@informatik.uni-freiburg.de
until 08.01.2013 (only Java sources and PDFs accepted)

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. Your tasks are indicated by ☞ inside a bigger story.
We recently received several questions regarding the lecture and, unfortunately, we did not yet have the time to answer these questions. Maybe you can help us.

Exercise 1: Operational Semantics
One day when I came to my office, an envelop was lying before my door. Wondering why it wasn't in my post office box, I grabbed it and opened it. Here is what I found.

```
Dear Lecturers,

one of my employees recently told me to "get up to date" regarding our
new production system.  This system is controlled by some Java program.
Unfortunately, I do not know anything about Java.  So I found the Java
Language Specification (JLS).  I glanced at it and quickly decided not
to read it.

Then I found the homepage of your lecture "Formal Methods for Java" and
thought I could get some insides from your slides.  Well, they were better
than the JLS.  But, looking at the formal semantics you describe in the
first few lectures, I still don't understand Java.  Why does a state
in your "transition system" consist of three pieces?  What do I need
them for?  And, even more striking, how do I use this semantics to
understand what a piece of Java code actually does?

Can you clarify these points?

Thank you,

The Bearded.

P.S.  We are very busy the whole December, so I will have someone collect
```

the answers in January. From then on, I will have some spare time to actually learn Java. Hopefully, I don't need your help next year.

☞ Find good answers to the questions asked in the letter above.

Exercise 2: Writing Specifications
We received the following email.

Dear Lecturers,

we stumbled across your lecture on "Formal Methods for Java" and need your help. We have a small program to solve a variant of a traveling salesman problem. The main variation is, that (1) we have a fully connected graph, and (2) costs also apply to the nodes.

The function signature is
Route computeRoute(double[][] costs, double tpp, int[] numPkgs);
where "costs" should be the adjacency matrix of the graph, "tpp" should be a positive number and "numPkgs" should give a non-negative number for every node. Unfortunately, the input to this function regularly gets "corrupted", i. e., is malformed in the sense that either "costs" is not an adjacency matrix, or "numPkgs" has wrong dimension, or the value constraints are violated.

We would like to specify that "costs" has to be an adjacency matrix for a fully connected graph where all entries are non-negative, "tpp" is positive, and "numPkgs" contains a non-negative number for every node in the graph. Unfortunately we failed at specifying these things. Hopefully you can help us and send us the correct specifications.

Due to intellectual property regulations, I cannot send you the source code, but hopefully you don't need it to specify this precondition.

Desperately waiting for your reply,

J.....

Our question for clarification of the shape of "costs" matrix was answered in this email.

Dear Lecturers,

> I don't understand your comments about the "costs" matrix. Shouldn't it
> be a symmetric matrix with 0s on the diagonal, e.g., have form
> 0 v1
> v1 0
> for a binary matrix.

thanks for your reply. You are right. We would like to have that specified

as well. Since we already fail on the "easier" task, I see no chance for us to specify this. Can you help there, too?

Thanks,

J.....

☞ Formulate the desired preconditions.

Exercise 3: Tracking Writes with JPF
A few days after we received the clarification for the previous exercise, we were asked to help in a related topic.

Dear Lecturers,

we investigated our code further and are pretty sure we initialize the parameters correctly. However, we call a piece of code where we do not have the sources available. As I've understood from your slides, JPF does not need source code, but can check binary code.

Is it possible to check with JPF that the array returned by an initialization function is never written anymore?

Thanks,

J.....

To clarify the task, I asked a question about the desired structure of a Java program. Here is the answer.

Dear Lecturers,

> I don't understand your setting. I assume your code looks like
> int[] arr = myInitFunc();
> fun1(arr);
> fun2(arr);
> where "myInitFunc" initializes the array, and "fun1" and "fun2" only read
> the array. The type of the array can be variable, right?

Yes, you are right. The type does not have to be "int". Furthermore, the array should not be written to after "myInitFunc" returns.

> If my assumptions are correct, we could write a listener that can be
> configured with a initialization function. The listener could flag the
> value returned by the ARETURN instruction of that function with a constant
> attribute and then check for every ArrayStoreInstruction whether the array
> we want to store anything into is flagged with this constant attribute.

```
Well, your assumptions are correct.  But how can I do that?

Thanks,

J.....
```

☞ Write a listener to check this.

Exercise 4: Different Invariant Systems
Another question recently arrived.

```
Dear Lecturers,

we have a question regarding invariants.  From the slides of your lecture we
learned that there are three different system to ensure object invariants:
1. the universes type system from JML,
2. the ownership principle with pack and unpack, and
3. the ownership principle with friendship.

We are trying to find the right system to express the object invariants for a
manufacturing and gift-wrapping system.  For every package, the system keeps
track of a state.

public class PackageState {
  private boolean manufactured;
  private boolean giftWrapped;
  private String storageLocation;
  private boolean onDelivery;
  // Getter and Setter for the members
  // Other stuff...
}

Essentially we want the invariants

1. giftWrapped ==> manufactured
2. storageLocation != null ==> giftWrapped
3. onDelivery ==> manufactured && giftWrapped && storageLocation == null

However, there are three components that individually change these statistics
objects:
1. the manufacturer sets the manufactured flag,
2. the gift-wrapper sets the giftWrapped flag, and finally
3. the storage system sets the storageLocation and the onDelivery flag.

Which system can we use to implement the desired object invariants?
```

```
Regards,

J....
```

☞ Discuss for the three system if and how the desired object invariants can be established.

Exercise 5: Sets in Jahob
A few days later, we were asked to assist in translating a piece of code from JML-annotated Java into Jahob-annotated Java.

```
Dear Lecturers,

we finally managed to formulate an algorithm using JML, but need some help to
transform this example into Jahob since we did not manage to verify the code
with JML.  We then wanted to try Jahob to verify that after inserting an
element into the tree, it is actually in it.  In JML we used JMLObjectSets to
represent sets of object.  However, Jahob cannot deal with JMLObjectSets.  The
slides of your lecture suggest that Jahob has some builtin set logic.  How can
we specify the set of nodes in a binary search tree.

The TreeNode class looks like
class TreeNode {
  TreeNode left;
  TreeNode right;
}

while the Tree class contains the lines
class Tree {
  TreeNode root;
  //: specvar nodes :: objset
  /*: vardefs nodes == ... */
  ...
}

where the remaining lines of this class are textbook standard.  Can you help?

Thanks,

J....
```

☞ Complete the definition of nodes of class Tree such that it corresponds to the set of all tree nodes in this tree. Use builtin sets, lambda expressions, and the closure predicate *rtrancl_pt*. You may assume the tree is acyclic.

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

Exercise 6: Timed Post Conditions (Bonus)
In the additional material for this exercise sheet you can find the file `PostCondition.java` that has some strange method `foo`.


Write a post condition for this method. Note that there seems to be a timing issue. So we are almost sure that the output changes over time.