EXAM
System Validation
(192140122)
13:45 - 17:15
02-07-2012

- The exercises are worth a total of 100 points.
  The final grade for the course is $\frac{(hw_1+hw_2)/2+exam/10}{2}$, provided that you obtain at least 50 points for the exam (otherwise, the final grade for the course is a 4).
- The exam is open book: all *paper* copies of slides, papers, notes etc. are allowed.

Exercise 1: Formal Tools and Techniques (10 points)

Suppose you are asked to be a consultant for a team developing a flight control system for a space craft. The team has 1 engineer who is well-trained in the use of formal methods. This person has practical experience with many different formal techniques. The other team members are well-skilled programmers, with a good understanding of the application domain, but without any knowledge of formal methods.

A prototype for the flight control system exists, but has not been tested yet, and it is likely to contain bugs. The deadline for the completion of the flight control system is in a month from now (and no extensions are possible). The flight control system has to fulfill strong requirements. In particular, the space craft should *always* remain controllable, so that the astronauts can get back to Earth.

Give a short description (no more than 200-250 words) of your advice to this team.

Exercise 2: Specification (15 points)

Write formal specifications for the following informal requirements in an appropriate specification formalism of your choice (3 points per item). You may assume that appropriate atomic propositions, query methods and classes exist.

a You can only open a file when it is closed.

b All doors of a plane must be closed when the plane is flying.

c If you role a dice, the outcome is always between 1 and 6.

d You are only allowed to live in a country as long as your passport is valid.

e For a calendar system : the month always follow the sequence January → February → March . . . December → January → February . . ..
NB There is no need to spell out all 12 cases, as long as the general idea of your specification is clear.

Exercise 3: Abstraction (8 points)

Consider interface `GameCharacter` and two classes implementing this interface: `Giant` and `Growing-Dwarf` (on the next page).

The interface `GameCharacter` is intended to model a character that can move along a horizontal axis. Initially it starts at position 0, and then it can move left or right. (NB, it is thus okay that position becomes negative).

Different game character implementations can make steps of different sizes, i.e., a giant always makes steps of 10 units, while the step size of the growing dwarf increases with every step he makes.

Write an abstract specification for `GameCharacter`, modelling the intended behaviour of `stepLeft` and `stepRight`. Then add sufficient specifications to `Giant` and `GrowingDwarf` in order to show that these implementations respect your specification of `GameCharacter`.

```java
package abstraction;

public interface GameCharacter {

        public void stepLeft();

        public void stepRight();
}
```

```java
package abstraction;

public class Giant implements GameCharacter {

        private int pos = 0;

        public void stepLeft(){
                pos = pos - 10;
        }

        public void stepRight(){
                pos = pos + 10;
        }
}
```

```java
package abstraction;

public class GrowingDwarf implements GameCharacter {

        private int pos = 0;
        private int steps_taken = 0;

        public void stepLeft() {
                pos = pos - steps_taken;
                steps_taken++;
        }

        public void stepRight() {
                pos = pos + steps_taken;
                steps_taken++;
        }
}
```

Exercise 4: Run-time Checking (15 points)

Each subquestion is worth 3 points. Explain your answers (without explanation no points will be awarded).

a Consider class `Computation`. What will happen when the JML run-time checker is used to validate this class?

```
package rc;

public class Computation {

        /*@ spec_public */ private int val;

        /*@ ensures val == \old(val) * \old(val);
         */
        public void compute() {
                val = 2 * val;
        }

        public static void main (String [] args) {
                Computation c = new Computation();
                c.compute();
        }
}
```

b Consider class `Screen`. What will happen when the JML run-time checker is used to validate this class?

```
package rc;

public class Screen {

        /*@ spec_public */ int x_size;
        /*@ spec_public */ int y_size;

        //@ invariant (x_size % 2) == 0;
        //@ invariant (y_size % 2) == 0;

        public void updateScreen() {
        }

        public void updateHorizontal() {
                x_size++;
                updateScreen();
                x_size++;
                updateScreen();
        }

        public static void main (String [] args) {
                Screen s = new Screen();
                s.updateScreen();
                s.updateHorizontal();
        }
}
```

c Consider class `Game`. What will happen when the JML run-time checker is used to validate this class?

```java
package rc;

class Player {}

public class Game {

        /*@ spec_public */ private Player player1 = new Player();
        /*@ spec_public */ private Player player2 = new Player();

        //@ invariant player1 != player2;

        public void battle(){
                warmUp(player1);
                warmUp(player2);
                fight();
        }

        /*@ requires player1 != player2;
                ensures false;
                signals (Exception) false;
         */
        public void fight() {
        }

        public void warmUp(Player p){
        }

        public static void main (String [] args) {
                Game g = new Game();
                g.battle();
        }
}
```

d Consider class `Student`. What will happen when the JML run-time checker is used to validate this class?

```java
package rc;
import java.util.HashMap;
import java.util.Map;

public class Student {

        public static final int numCourses = 35;
        /*@ spec_public */ private int[] grades = new int[numCourses];

        //@ invariant grades != null;
        /*@ invariant (\forall int i; 0 <= i && i < numCourses;
                                  0 <= grades[i] && grades[i] <= 10);
         */

        /*@ requires 0 <= grade && grade <= 10;
                requires 0 <= course && course < numCourses;
                ensures grades[course] == grade;
        */
        public void newResult(int course, int grade) {
            grades[course] = grade;
        }

        public static void main (String [] args) {
                Student s = new Student();
                s.newResult(22, 86);
        }
}
```

e Consider class `Resources`. What will happen when the JML run-time checker is used to validate this class?

```java
package rc;

class Thing {
        //@ public static ghost int countThings= 0;

        //@ ensures countThings == \old(countThings) + 1;
        public Thing() {
                //@ set countThings = countThings + 1;
        }

}

public class Resources {

        public static final int MAX = 26;

        //@ invariant Thing.countThings <= MAX;

        /*@ requires p > 0;
               ensures Thing.countThings == \old(Thing.countThings) + p;
         */
        public void makeThings(int p) {
                Thing[] storage = new Thing[p];
                //@ loop_invariant 0 <= i && i <= storage.length;
                /*@ loop_invariant Thing.countThings ==
                                  \old(Thing.countThings) + i;
                 */
                for (int i = 0; i < storage.length; i++) {
                        new Thing();
                }
        }

        public static void main (String [] args) {
                Resources r= new Resources();
                r.makeThings(13);
        }
}
```

Exercise 5: Static Checking (15 points)

Subquestions a, b, c are worth 3 points, subquestion d is worth 6 points. Explain your answers (without explanation no points will be awarded).

a Consider again class `Resources` from exercise 4e. What will happen when ESC/Java is used to validate this class.

b Consider class `Channel`. Add a loop invariant.

```
package sc;

class InputException extends Exception {}
class NotEnoughInputException extends Exception {}

class Token{}

public class Channel {

        /*@ spec_public */ private Token [] channel = new Token[4];
        /*@ spec_public */ private int [] processed = new int[4];

        //@ invariant channel != null;
        //@ invariant processed != null;

        /*@ pure */ public boolean illegalToken(Token s) {
                return false;
        }

        /*@ pure */ public int process (Token s) {
                return 0;
        }

        /*@ requires n > 0;
               ensures (\forall int i; 0 <= i && i < n;
                               processed[i] == process(channel[i]));
               signals (NotEnoughInputException) false;
               signals (InputException) false;
         */
        public void readTokens(int n) throws NotEnoughInputException,
                                         InputException {
                if (n > channel.length) {
                        throw new NotEnoughInputException();
                        }
                processed = new int [n];
                for (int i = 0; i < processed.length; i++) {
                        if (illegalToken(channel[i])) {
                                throw new InputException();
                                }
                        else processed[i] = process(channel[i]);
                }
        }
}
```

c Assuming that your loop invariant is correct, what will happen when ESC/Java is applied on `Channel`?

d Consider class `GenderBalance`. When using ESC/Java to validate this class, two different errors will be signalled. What are these errors, and what causes them (3 points per error).

```java
package sc;

public class GenderBalance {

        /*@ spec_public */ private int nr_of_men;
    /*@ spec_public */ private int nr_of_women;

    //@ invariant nr_of_men <= nr_of_women;

    public int addMen(int m) {
        return nr_of_men + m;
    }

    public int addWomen(int w) {
        return nr_of_women + w;
    }

    /*@ requires m <= w;
          ensures nr_of_men == \old(nr_of_men) + m;
          ensures nr_of_women == \old(nr_of_women) + m;
     */
    public void recruitment(int m, int w) {
        nr_of_men = addMen(m);
        nr_of_women = addWomen(w);
    }
}
```

Exercise 6: Modeling (22 points)

Consider the Bean Can Game.

You and a friend are given a can containing N black beans and M white beans initially. The can is emptied according to the following repeated process:

- Select two beans from the can
- If the beans are:
  - the same color: put a black bean back in the can
  - different colors: put a white bean back in the can

The player who chooses the color of the remaining bean wins the game.

```
-- Bean Can Algorithm pseudocode

while (num-beans-in-can > 1) do {
   pick 2 beans randomly
   if bean1-color == bean2-color
   then put-back black bean
   else put-back white bean
}
```

a (*12 pnts.*) Write a NuSMV model for this game.

b (*3 pnts.*) Write a property to specify that the game always terminates.

c (*3 pnts.*) Write a property to specify that black always is able to win the game.

d (*4 pnts.*) Suppose you want to be sure that the model has no executions where it always picks only black beans. What change would you have to make to the model to ensure this.

Exercise 7: Traces (15 points)

Consider the following Java program:

```java
import gov.nasa.jpf.jvm.Verify;
public class Workers {
  public static Barrier barrier;
  public static Object lock1=new Object();
  public static Object lock2=new Object();

  public static void main(String args[]) throws Exception {
    barrier=new Barrier(2);
    Thread t1 = new Thread(new Runner());
    Thread t2 = new Thread(new Runner());
    Thread t3 = new Thread(new Runner());
    Verify.beginAtomic();
    t1.start();
    t2.start();
    t3.start();
    Verify.endAtomic();
    t1.join();
    t2.join();
    t3.join();
  }

  public static class Runner implements Runnable {
    public void run(){
      synchronized(lock1){
        synchronized(lock2){
          // do some work.
        }
      }
      barrier.sync();
      synchronized(lock2){
        synchronized(lock1){
          // do more work.
        }
      }
    }
  }
}
```

The code for the barrier used is:

```
/**
 A barrier is a way of synchronizing N threads.
  It makes a group of threads wait until the last one
  has reached the barrier (called sync).
 */

public class Barrier {

  private int threads;
  private int blocked;
  private boolean round;

  public Barrier(int threads){
    this.threads = threads;
    blocked = 0;
  }

  public synchronized void sync(){
    blocked++;
    if (blocked < threads){
      boolean expect = !round;
      while (round != expect) {
        try { wait(); } catch (InterruptedException e) {}
      }
    } else {
      round = !round;
      blocked = 0;
      notifyAll();
    }
  }

}
```

To analyse the presence of deadlocks for this program, we have used JPF. The input for JPF is:

```
# class to verify
target = Workers

# where to find byte code and source
classpath=.
sourcepath=.

# set heuristic
search.class = .search.heuristic.BFSHeuristic
search.multiple_errors = true

# how do we want the result?
report.console.property_violation=error,snapshot,trace
```

```

# what are we looking for?
listener=.listener.DeadlockAnalyzer

# how to display deadlocks
deadlock.format=essential
```

JPF signals that there is a problem with this code. Analyse the counter example (starting on the next page) and explain why the assertion was triggered.

When you describe the scenario, it is important to state which thread is running and precisely what the first and/or last action of the running thread during its turn is. (for example: “thread 37 starts by reading 7 from variable x and stops just before reading variable y.”) The description of the other things a thread does during its turn can and *should* be much less detailed.

```
====================================================== error #71
gov.nasa.jpf.jvm.NotDeadlockedProperty
deadlock encountered:
  thread java.lang.Thread:{id:0,name:main,status:WAITING,priority:5,lockCount:0,suspendCount:0}
  thread java.lang.Thread:{id:1,name:Thread-1,status:WAITING,priority:5,lockCount:1,suspendCount:0}
  thread java.lang.Thread:{id:2,name:Thread-2,status:TERMINATED,priority:5,lockCount:0,suspendCount:0}
  thread java.lang.Thread:{id:3,name:Thread-3,status:TERMINATED,priority:5,lockCount:0,suspendCount:0}


====================================================== snapshot #71
thread java.lang.Thread:{id:0,name:main,status:WAITING,priority:5,lockCount:0,suspendCount:0}
  waiting on: java.lang.Thread@143
  call stack:
        at java.lang.Thread.join(Thread.java:-1)
        at Workers.main(Workers.java:17)

thread java.lang.Thread:{id:1,name:Thread-1,status:WAITING,priority:5,lockCount:1,suspendCount:0}
  waiting on: Barrier@142
  call stack:
        at java.lang.Object.wait(Object.java:-1)
        at Barrier.sync(Barrier.java:23)
        at Workers$Runner.run(Workers.java:28)


====================================================== trace #71
------------------------------------------------------ transition #0 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"<root>" ,1/1,isCascaded:false}
      [2895 insn w/o sources]
  Workers.java:4                  : public static Object lock1=new Object();
      [1 insn w/o sources]
  Workers.java:4                  : public static Object lock1=new Object();
  Workers.java:5                  : public static Object lock2=new Object();
      [1 insn w/o sources]
  Workers.java:5                  : public static Object lock2=new Object();
      [2 insn w/o sources]
  Workers.java:8                  : barrier=new Barrier(2);
  Barrier.java:13                 : public Barrier(int threads){
      [1 insn w/o sources]
  Barrier.java:14                 : this.threads=threads;
  Barrier.java:15                 : blocked=0;
  Barrier.java:16                 : }
  Workers.java:8                  : barrier=new Barrier(2);
  Workers.java:9                  : Thread t1=new Thread(new Runner());
  Workers.java:21                 : public static class Runner implements Runnable {
      [1 insn w/o sources]
  Workers.java:21                 : public static class Runner implements Runnable {
  Workers.java:9                  : Thread t1=new Thread(new Runner());
      [188 insn w/o sources]
  Workers.java:9                  : Thread t1=new Thread(new Runner());
  Workers.java:10                 : Thread t2=new Thread(new Runner());
  Workers.java:21                 : public static class Runner implements Runnable {
      [1 insn w/o sources]
  Workers.java:21                 : public static class Runner implements Runnable {
  Workers.java:10                 : Thread t2=new Thread(new Runner());
      [141 insn w/o sources]
  Workers.java:10                 : Thread t2=new Thread(new Runner());
  Workers.java:11                 : Thread t3=new Thread(new Runner());
  Workers.java:21                 : public static class Runner implements Runnable {
      [1 insn w/o sources]
  Workers.java:21                 : public static class Runner implements Runnable {
  Workers.java:11                 : Thread t3=new Thread(new Runner());
      [141 insn w/o sources]
  Workers.java:11                 : Thread t3=new Thread(new Runner());
  Workers.java:12                 : Verify.beginAtomic();
```

```
      [2 insn w/o sources]
  Workers.java:13                 : t1.start();
      [2 insn w/o sources]
  Workers.java:14                 : t2.start();
      [2 insn w/o sources]
  Workers.java:15                 : t3.start();
      [2 insn w/o sources]
  Workers.java:16                 : Verify.endAtomic();
      [1 insn w/o sources]
------------------------------------------------------ transition #1 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"end atomic" ,1/4,isCascaded:false}
      [2 insn w/o sources]
  Workers.java:17                 : t1.join();
      [1 insn w/o sources]
------------------------------------------------------ transition #2 thread: 3
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"wait" ,3/3,isCascaded:false}
      [2 insn w/o sources]
  Workers.java:23                 : synchronized(lock1){
  Workers.java:24                 : synchronized(lock2){
  Workers.java:26                 : }
  Workers.java:27                 : }
------------------------------------------------------ transition #3 thread: 3
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"sharedField" ,3/3,isCascaded:false}
  Workers.java:28                 : barrier.sync();
  Barrier.java:19                 : blocked++;
  Barrier.java:20                 : if (blocked<threads){
  Barrier.java:21                 : boolean expect=!round;
  Barrier.java:22                 : while (round!=expect) {
  Barrier.java:23                 : try { wait(); } catch (InterruptedException e) {}
      [1 insn w/o sources]
------------------------------------------------------ transition #4 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"wait" ,1/2,isCascaded:false}
      [2 insn w/o sources]
  Workers.java:23                 : synchronized(lock1){
  Workers.java:24                 : synchronized(lock2){
  Workers.java:26                 : }
  Workers.java:27                 : }
------------------------------------------------------ transition #5 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"sharedField" ,2/2,isCascaded:false}
      [2 insn w/o sources]
  Workers.java:23                 : synchronized(lock1){
------------------------------------------------------ transition #6 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"monitorEnter" ,2/2,isCascaded:false}
  Workers.java:23                 : synchronized(lock1){
  Workers.java:24                 : synchronized(lock2){
------------------------------------------------------ transition #7 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"monitorEnter" ,2/2,isCascaded:false}
  Workers.java:24                 : synchronized(lock2){
  Workers.java:26                 : }
  Workers.java:27                 : }
------------------------------------------------------ transition #8 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"sharedField" ,2/2,isCascaded:false}
  Workers.java:28                 : barrier.sync();
  Barrier.java:19                 : blocked++;
  Barrier.java:20                 : if (blocked<threads){
  Barrier.java:26                 : round=!round;
  Barrier.java:27                 : blocked=0;
  Barrier.java:28                 : notifyAll();
      [2 insn w/o sources]
  Barrier.java:30                 : }
  Workers.java:29                 : synchronized(lock2){
------------------------------------------------------ transition #9 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"monitorEnter" ,1/3,isCascaded:false}
  Workers.java:28                 : barrier.sync();
------------------------------------------------------ transition #10 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"monitorEnter" ,1/3,isCascaded:false}
  Workers.java:28                 : barrier.sync();
  Barrier.java:19                 : blocked++;
  Barrier.java:20                 : if (blocked<threads){
  Barrier.java:21                 : boolean expect=!round;
  Barrier.java:22                : while (round!=expect) {
  Barrier.java:23                : try { wait(); } catch (InterruptedException e) {}
      [1 insn w/o sources]
------------------------------------------------------------ transition #11 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"wait" ,1/2,isCascaded:false}
  Workers.java:29                : synchronized(lock2){
  Workers.java:30                : synchronized(lock1){
------------------------------------------------------------ transition #12 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"monitorEnter" ,1/2,isCascaded:false}
  Workers.java:30                : synchronized(lock1){
  Workers.java:32                : }
  Workers.java:33                : }
  Workers.java:34                : }
      [1 insn w/o sources]
------------------------------------------------------------ transition #13 thread: 3
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"terminate" ,1/1,isCascaded:false}
      [2 insn w/o sources]
  Barrier.java:23                : try { wait(); } catch (InterruptedException e) {}
  Barrier.java:22                : while (round!=expect) {
  Barrier.java:25                : } else {
  Barrier.java:30                : }
  Workers.java:29                : synchronized(lock2){
  Workers.java:30                : synchronized(lock1){
  Workers.java:32                : }
  Workers.java:33                : }
  Workers.java:34                : }
      [1 insn w/o sources]

============================================================ thread ops #71
   1        3        2        0       trans      insn          loc                 : stmt
-------- -------- -------- -------- ---------------------------------------------------------
   |        |        T        |       1354   directcallreturn [synthetic] [run]
W:322       |        |        |        930      invokevirtual Barrier.java:23 : try { wait()
    ; } catch (InterruptedException e) {}
   |     B:322       |        |        930      invokevirtual Barrier.java:23 : try { wait()
        ; } catch (InterruptedException e) {}
L:322       |        |        |        930      invokevirtual Workers.java:28 : barrier.sync
    ();
   |        |      A:322      |        507      invokevirtual Barrier.java:28 : notifyAll();
   |        |        |      W:323        1      invokevirtual Workers.java:17 : t1.join();
   |        S        |        |          0
   |        |        S        |          0
   S        |        |        |          0
```