Lutz Schröder und Till Mossakowski

Spezifikation und Verifikation von Java-Programmen Übungsblatt 4

Abgabe: zwei Werktage vor dem Fachgespräch

An integer queue is an abstract datatype Q with operations n, e, d, i, f (nil, enqueue, dequeue, isnil, first) obeying the following laws, for $q, q' \in Q, j, j' \in Z$:

- $n \neq e(j, q)$.
- $e(j, q) = e(j', q') \Rightarrow j = j' \wedge q = q'$.
- $i(n) = \top$, $i(e(j, q)) = \bot$.
- $d(e(j, n)) = n$, $d(e(j, e(j', q))) = e(j, d(e(j', q)))$.
- $f(e(j, n)) = j$, $f(e(j, e(j', q))) = f(e(j', q))$

The file `Queue.java` contains a Java class that implements an integer queue. Specify the private behavior of this class as strongly as possible; as a minimum `escjava2` shall not complain. Please note that `Queue` contains a bug that has to be fixed for this purpose. Write a program `Main` that tests the queue in a simple way. Compile the test program with the runtime assertion checking tool `jmlc` and let it run with `jmlrac`. Then also specify the public behavior of the class in a JML specification file Queue.jml using a model type QueueModel.

As a result of this exercise, deliver

a) the source of `Main.java` and of the JML annotated (buggy) `Queue.java` specifiying the private behavior;

b) the output of an execution of `Main` with the buggy class `Queue` using `jmlrac` such that an assertion exception demonstrates the bug;

c) the output of `escjava2` on `Main` and the buggy `Queue`;

d) the source of the JML annotated `Queue.java` after fixing the bug;

e) the output of a correct execution of `Main` with `jmlrac`;

f) the output of of `escjava2` on `Main` and the correct `Queue`.

g) the source of the corrected `Queue.java`, `Queue.jml` and `QueueModel.java` specifying the public behavior and the output of escjava2 on these files.

```
class Queue
{
  private int head = 0;
  private int tail = 0;
  private int count = 0;
  private int N = 3;

  public Queue()
  {
    a = new int[N];
  }

  public boolean isempty()
  {
    return count == 0;
  }

  public void enqueue(int value)
  {
   if (count == a.length) resize();
   count = count+1;
   a[tail] = value;
   tail = tail+1;
   if (tail == a.length) tail = 0;
  }

  public void dequeue()
  {
    count = count-1;
    head = head+1;
  }

  public int first()
  {
    return a[head];
  }

  private void resize()
  {
    int b[] = new int[2*a.length+1];
    for (int i=head; i<a.length; i++)
      b[i-head] = a[i];
    for (int i=0; i<head; i++)
      b[i+a.length] = a[i];
    head = 0;
    tail = count;
    a = b;
  }
}
```