Concurrent Programming

Final Examination — 2001-02-19

Please answer 5 questions for a maximum of 20 points.

(Only the first 5 questions you answer will be graded.)

1. Explain when you would use the following Java statement. Be sure to state explicitly any assumptions. **(2 points)**

   ```
   new Thread(this).start();
   ```

   What is the difference between that statement and the one below?

   ```
   new Thread(this).run();
   ```

2. Is the Point class below safe in the presence of multiple threads? Why or why not? What about subclasses of Point? **(2 points)**

   ```
   class Point {        // is this class thread-safe?
     protected long x_, y_;

     public Point(long x, long y) { x_ = x; y_ = y; }

     public Point plus(Point other) {
       return new Point(x_ + other.x_, y_ + other.y_);
     }

     public Point minus(Point other) {
       return new Point(x_ - other.x_, y_ - other.y_);
     }
   }
   ```

3. One solution to the Dining Philosophers problem is to make one Philosopher always grab his right fork first, and the other four grab their left fork first. How does this avoid deadlock? **(2 points)**

4. This Lock class is implemented using an instance of Slot (from the lecture notes). Does it suffer from any liveness problems? If so, what are they, and how would you fix them? **(2 points)**

```
public class Lock {
  private Slot slot_ = new Slot();

  public synchronized void acquire() {
    slot_.put(this);
  }

  public synchronized void release() {
    slot_.get();
  }
}
```

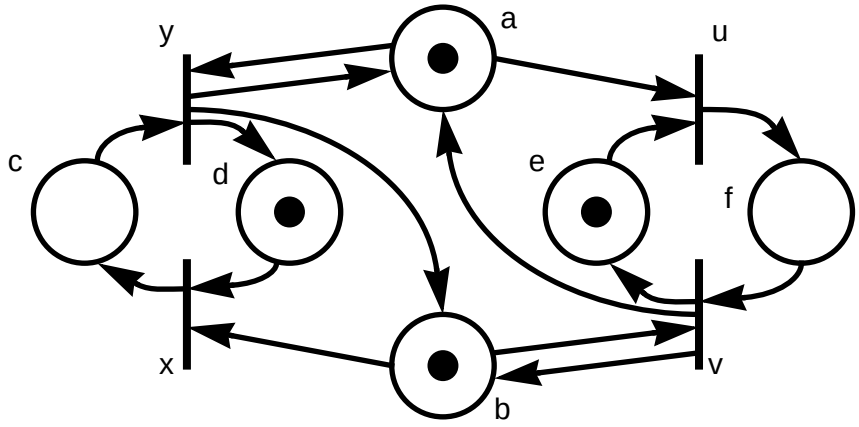
Here, once again, is the code of the Slot class:

```
class Slot {
  private Object slotVal;   // initially null

  public synchronized void put(Object val) {
    while (slotVal != null) {
      try { wait(); }
      catch (InterruptedException e) { }
    }
    slotVal = val;
    notifyAll();
    return;
  }

  public synchronized Object get() {
    Object rval;
    while (slotVal == null) {
      try { wait(); }
      catch (InterruptedException e) { }
    }
    rval = slotVal;
    slotVal = null;
    notifyAll();
    return rval;
  }
}
```

5. Consider the following implementation of a Future. Would it work the same if you replace "while" by "if" in the value() method? What if you replace "notifyAll()" by "notify()" in the set() method? Justify your answers. **(2 points)**

```
class Future {
  private Object val_ = null;
  public synchronized Object value() {
    while (val_ == null) {  // can you replace while by if?
      try { wait(); }
      catch(InterruptedException err) {}
    }
    return val_;
  }
  public synchronized void set(Object val) {
    if (val_ == null)
      val_ = val;
    notifyAll();            // can you use notify() instead?
  }
}
```

6. Is the given Petri net bounded? Safe? Conservative? Are all the transitions live? **(2 points)**



7. Draw or specify a Petri net equivalent to the following FSP. Is the resulting net bounded? Safe? Conservative? Are all the transitions live? **(2 points)**

```
P = a -> ( b -> P | c -> P )
```