Question 1. (6 points)

a) Legal
b) Not legal
c) Not legal
d) Not legal
e) Not legal
f) Legal

```
// NegativeInteger extends Integer, so pei can reference a PriorityQueue of NegativeInteger's.
//   See slide 43 in Generics.pdf
pei = new PriorityQueue<NegativeInteger>();

// Can only add(null). See slide 43 in Generics.pdf
// pei.add(ni); // assume pei and ni are not null

// Can only add(null). See slide 43 in Generics.pdf
// pei.add(i); // assume pei and i are not null

// Can read from PriorityQueue (poll()) but the return value is an Integer (the bound of the
//   wildcard). An Integer cannot be assigned to a NegativeInteger. See slide 43 in Generics.pdf
// ni = pei.poll(); // assume pei is not null

// There is no way a PriorityQueue of Integers or subclasses can be assigned to a PriorityQueue
//   of Integers or superclasses. Otherwise, we would be able to, e.g., add an Integer
//   psi.add(new Integer ()); to a collection of, let's say PositiveInteger's (if pei was
//   previously pei = new PriorityQueue< PositiveInteger >();).
// psi = pei;

// A NegativeInteger can be added to a PriorityQueue of Integers or superclasses.
//   See slide 47 in Generics.pdf
// psi.add(ni); // assume psi and ni are not null
```

Question 2 (2 points)

The interning pattern has two drawbacks: every call to the factory for a class has to do an extra "if" comparison and hash lookup, and every instantiated object gets persisted in memory for the entire duration of the program. This would make interning prohibitively expensive in both extra computation and in memory for integers, which are often ubiquitous throughout a program, but are unique or rarely reused.

Question 3 (2 points)

False

See slide 68 from DesignPatterns.pdf. A class must implement Cloneable interface to properly support cloning.