OOP Sample Question 1 Solution(RKH)

Note: This question was updated for clarity on 27/04/11

(*a*)

(*i*) Information hiding is the idea that data should only be accessible via a stable, well-specified set of functions. Your example should show a class with a `private` data structure and one or more `public` functions that access or mutate it in some way.

(*ii*) Inheritance is the idea that one class may automatically inherit state and behaviour from another. This is useful to represent a concept that is a specialisation of another. A simple example of two Java classes is sufficient.

(*iii*) A child class may override any behaviour it inherits from its parent, meaning the parent and child have different behaviours for the same mathod. Dynamic or ad-hoc polymorphism is when the appropriate version of the method is determined at run time, based on the most derived type of an object. A simple example of polymorphism (perhaps using two classes and a function that prints text) would be sufficient.

(*iii*) Static polymorphism applies when we have code that is written generically e.g. a template for a data structure that could be applied to a range of types. For each instance, the type is determined at compile time. A good example of Generics in Java might involve a `LinkedList` of a specific type.

(*b*)

(*i*) Because an `int` is a primitive type. We want the list to work for all objects, but this necessitates a 'wrapper' class for the primitive types (which are obviously not objects).

(*ii*) The identifier `i` is reused, and we call `intValue()` on the wrong object. The code should be:

```
public void StripNegatives(List intlist) {

   for (int i=0; i<intlist.size(); i++) {
     Object o = intlist.get(i);
     Integer x = (Integer)o;
     if (x.intValue()<0) intlist.remove(i);
   }
}
```

(*iii*) There is, however, a more insidious problem, that occurs when we have two negatives in a row. The first one will be strippoed out, and the counter incremented by one. Simultaneously the size of the list will fall by one, so the next item is not checked!

E.g. {1, -2,-3, 4}.

First loop: `intList.size()` is 4, `i` is 0. We check the number 1 and proceed to increment `i`.

Second loop: `intList.size()` is 4, `i` is 1. We check the number -1 and strip it out. We increment `i`.

Third loop: `intList.size()` is now 3, `i` is 2. So now we check the number 4—i.e. we skipped the check of the number -3.

(*iii*) The looping problem is solved by:

```
public void StripNegatives(List intlist) {
   Iterator it = l.iterator();
   while (it.hasNext()) {
     Integer x = (Integer)it.next();
     if (x.intValue()<0) it.remove();
   }
}
```

(*iv*) Something like:

```
public void StripNegatives(List<Integer> intlist) {
   Iterator<Integer> it = l.iterator();
   while (it.hasNext()) {
     Integer x = it.next();
     if (x.intValue()<0) it.remove();
   }
}
```