Solution notes

Foundations of Programming (Java) 2005 – Paper 10 Question 2 (FHK)

THE INSTANCE METHOD AND CLASS METHOD PROBLEM

Context: Much of this question stems from MODULE3p and
MODULE3q of the Foundations of Programming Course.

Part (a). Bookwork. Almost any indication that the candidate knows that instance methods are not heralded by the static modifier but that class methods are heralded by the static modifier will be acceptable. [4 marks]

Part (b). The statement `c = p;` provokes the compiler error. The typing rules preclude assigning a variable of parent type to one of a child type. A candidate who distinguishes between static type and dynamic type will be appreciated. The simple correction is to add a cast as in `c = (Child)p;` [4 marks]

Part (c). Variable p is assigned a new object of type Parent and thereby acquires an instance method test() which, when invoked, writes out "Parent". The first `p.test()` duly does this.

Variable c is assigned a new object of type Child. Class Child extends class Parent but overrides the test() method with one that writes out "Child". The first `c.test()` duly invokes this test() method and writes out "Child".

Next, p is assigned c so both p and c refer to the same object which is of type Child. Accordingly p.test() and c.test() both now write out "Child".

Then, c is assigned (Child)p which has no effect. Both p and c continue to refer to the same object of type Child. Accordingly. the final p.test() and c.test() again both write out "Child".

In short, the output is:

```
Parent
Child
Child
Child
Child
Child
```
[5 marks]

Part (d). If both test() methods are made static there is a subtle change. The small print which says that you 'cannot override a static method but you can hide it' has important implications for the variables p and c...

Variable p is assigned a new object of type Parent but now has class method test(), NOT an instance method. The first p.test() though behaves as before and writes out "Parent".

Variable c is assigned a new object of type Child and now has the class method test() which, because of the small print, hides the test() method of Parent but does not override it. The first c.test() though again behaves as before and writes out "Child".

Next p is assigned c so both p and c refer to the same object. Notwithstanding this common reference, the class method test() to which p refers is still that in class Parent. Likewise, the class method to which c refers is still that in class Child. Accordingly p.test() again writes out "Parent" and c.test() again writes out "Child".

Next c is assigned (Child)p which has no effect. Both p and c continue to refer to the same object (thus p==c is true) but p.test() is still the class method in Parent and c.test() is still the the class method in Child so they again write out "Parent" and "Child" respectively.

In short, the output is:

```
Parent
Child
Parent
Child
Parent
Child
```

[7 marks]