Lab 07 Review, Inheritance and More

Q1-6 MC Questions - finish on **Canvas**

Q7 [Homework] Programming - submit to PASS

Q1-Q6 Read the following notes. Then answer the questions on **Canvas** (MC question).

Review

[redacted]

Visibility (public / protected / private):

- When we implement or redefine a method in the subclass, it must be at least as "visible" as one in the superclass. static ✓ non-static ✓
- A subclass cannot access the private members in the superclass.

```
class X {public void f1() { }}
class Y extends X {
                    void f1() {}
}
```

Must be public

x protected

x private

x [empty]

"package private"

The static keyword

Used to denote fields and methods that belong to a class (but not to any particular object).

The abstract keyword

The **abstract** keyword is applied for **classes** and **nonstatic methods**:

- When applied for a nonstatic **method**: means that we intend to provide **no implementation**; and the implementation will be provided in concrete subclasses.
- When applied for a **class**: means that the class may or may not include abstract methods. Abstract classes **cannot be instantiated** (ie. cannot be used to instantiate any object), but they **can be subclassed**.

- **abstract** is NOT for fields (no matter static or nonstatic)
- **abstract** is NOT for constructors or static methods

Polymorphism – An object variable can refer to different actual types.

Superclass

Superclass and subclass which are concrete

Dynamic Binding – Automatically select the appropriate **method** at runtime (Note: not fields!!)

Q1 Use of `abstract`

Read the code below. Which lines contain invalid code? Explain.

```
abstract class A
{
    public int p1;              //line 1
    public abstract int p2;     //line 2 × abstract is not for fields

    public void x1() {}         //line 3
    public void x2();           //line 4 × not an abstract method
                                           must provide method code

    public abstract void y1() {}   //line 5 × is an abstract method
    public abstract void y2();     //line 6     should not give code

    public abstract static void z1();   //line 7 × abstract method must NOT be static
}
```

reason: abstract method aims at giving the magic: 'dynamic binding' such magic occurs on objects (select the appropriate method by checking the actual type of the object, not the type of object variable).

static means 'not to any particular object'

Hint: There are choices in the MC questions on blackboard!

Q2 static and non-static fields and methods

Read the code below. Which lines contain invalid code? Explain.

```
class A {

    private int i;
    private static int j;

    public static void f1() {
        i++;   //line 1     x   static method does not run on
    }                           any particular object
                                (i.e. no 'this', the implicit parameter)
    public void f2() {          therefore, it doesn't know what i is.
        i++;   //line 2
    }

    public static void f3() {
        j++;   //line 3
    }

    public void f4() {
        j++;   //line 4
    }

}

public class Main_Lab07Q2
{
    public static void main(String[] args)
    {
        A obj = new A();

        obj.f1(); //line 5
        obj.f2(); //line 6
        obj.f3(); //line 7
        obj.f4(); //line 8

        A.f1();    //line 9
        A.f2();    //line 10  x Non-static method must be invoked
        A.f3();    //line 11          for a particular object.
        A.f4();    //line 12  x ..
    }
}
```

Q3 Visibility in inheritance

Read the code below. Which lines contain invalid code? Explain.

```
abstract class A {
   public int i;
   protected int j;
   private int k;

   public static void print1() {}
   public void print2() {}
}

class B extends A
{
   protected static void print1() //line 1  × Visibility cannot be reduced
   {
      System.out.println(i); //line 2  × see Q2 line 1
      System.out.println(j); //line 3  ×
      System.out.println(k); //line 4  ×
   }

   protected void print2() //line 5   ×
   {
      System.out.println(i); //line 6
      System.out.println(j); //line 7
      System.out.println(k); //line 8  × Cannot access private field
                                          defined in superclass.
   }
}
```

Q4 Polymorphism and Dynamic binding

Read the code below. Which lines contain invalid code? Explain.

```
abstract class A {
   public abstract void fi();
}

class B extends A {
   public void fi() {}
   public void fj() {}
}

public class Main_Lab07Q4
{
   public static void main(String[] args)
   {
      A a;           //line 1
      a = new A();   //line 2  × abstract class cannot be instantiated
      a.fi();        //line 3  (Assume any error(s) in line1-2 are removed)

      B b;           //line 4
      b = new B();   //line 5
      b.fj();        //line 6

      B b1 = a;      //line 7  × the type of a is A, which is not a kind of B
                                                                 subclass
      A a1 = b;      //line 8    if apply casting: B b1 = (B)a;
      b.fj();        //line 9       compile OK, but runtime error can occur.
      a1.fj();       //line 10 × the type of a1 is A
   }                              A does not have method fj
}                                 but we can write ((B)ai).fj();
```