1. Java Generics
   Code fragments to illustrate generics from Ref: Gilad Baracha – Generics Tutorial

```java
List<Integer> lint = new ArrayList<Integer>();     //1
lint.add(new Integer(100));                        //2

Iterator<Integer> intIter = lint.iterator();       //3
Integer i1 = intIter.next();                       //4

lint.add(101);                                     //5
int i2 = intIter.next();                           //6

public class X { public X();}                      //7
public class Y extends X { public Y();}            //8

List<X> lx = new ArrayList<X>();                   //9
List<Y> ly = new ArrayList<Y>();                   //10
lx = ly;                                           //11
System.out.println(lx.getClass()==ly.getClass());  //12

void printCollection(Collection c) {              //13
    Iterator i = c.iterator();
    while(i.hasNext()) {
        System.out.println(i.next());
    }
}

void printCollection(Collection<Object> c) {       //14
    for (Object o: c) {
        System.out.println(o);
    }
}
```

Will lines 1 through 11 compile correctly?

What is the output of line 12?

Is code segment beginning at line 14 equivalent to the one beginning at line 13? What other variations are possible? Is it useful?
2. Inner Classes

Code fragments to illustrate access in inner classes. Will the following code snippets compile?

```java
public class X {                                   //1
    private int fx;

    public class I {
        int fi;
        public I() { fi = fx; }
    }
}

public class X {                                   //2
    private int fx;
    private I i = new I();
    private int fx2 = i.fi;

    public class I {
        private int fi;
        public I() { fi = fx; }
    }
}

public class X {                                   //3
    int fx;

    public static class I {
        int fi;
        public I() { fi = fx; }
    }
}
```
3. DisInherit Methods

Is it possible for a class to disinherit methods? Why or why not?

4. Lisp List Implementation

![Diagram of list structure]

We want to implement a list similar to the one defined in Lisp language using a single class Cons which represents a cons cell as shown in the figure above. The car field contains the data element and the cdr field points to the remaining list or is null. This has the nice property that if we take any sub-list, it is a list too. We want constructors that can construct an empty list or one from a given list. We want to define two operations on the list first() (or car()) that returns the cell element from the head of the list and rest() (or cdr()) that returns the remaining list. Both of these do not modify the original list. Another method we want is length() that returns the number of cells in the list.

First, write a LispList interface for this datatype and next write a skeleton for the Cons class.

5. List Of Arrays

We saw in the phone directory example the need for a data structure that supports addition and removal of elements from it. The implementation that was used there was an array but it had the disadvantage that if we ran out of space, we had to reallocate the array and copy elements over. A hybrid data structure List of Arrays is a list that contains an array as the data element. What advantages such a data structure may have? Sketch an implementation of this data structure.

---

The names are a legacy of the architecture of Lisp implementations.