Intermediate handout

The discussion in the first handout focussed on examples of class definitions where the fields were of standard simple types. However this limits their use considerably, and there is no real reason why fields of one class should not be of a type already introduced by another class definition, as some of you may have realised. For example, here are three such interdependent class definitions:

```
class TestResult {
  int mark;
  bool passed, first;
};

class StudentNumber {
  int year, number;
  char initial;
};

class Student {
  StudentNumber sn;
  TestResult   examResult;
  TestResult[] testResults;
};
```

In this case we might find code such as the following:

```
Student topStudent     = new Student();

topStudent.sn          = new StudentNumber();
topStudent.sn.year     = 63;
topStudent.sn.initial  = 'T';
topStudent.sn.number   = 0844;                    // look familiar?

topStudent.testResults = new TestResult[20];

topStudent.examResult  = new TestResult();
topStudent.examResult.mark = 98;                  // shocking!

writeLine("Results for ", topStudent.sn.year, topStudent.sn.initial,
          topStudent.sn.number, " Exam result = ", topStudent.examResult.mark);

int i = 0;
while (i < 20) {
  topStudent.testResults[i] = new TestResult();
  topStudent.testResults[i].mark = i + 81;        // gradual improvement?
  writeLine(topStudent.sn.year + 1900 + i, topStudent.testResults[i].mark);
  i++;
}
```

If you had not thought of this complication previously, ignore it, and concentrate on the cases implied by the first handout, in which the fields of classes/objects were limited to the basic types `int`, `bool` and `char` and arrays of these.

Although straightforward, there is quite a lot of extra coding needed to handle the more general case. Later in the day we shall make suggestions how this could be done, for those of you who are curious.