Question 1

The students mostly have no previous programming experience and find nuts-and-bolts Java programming very difficult. Partially correct solutions will therefore be marked generously.

There was only one, partial, attempt at this question.

This question is about "nuts and bolts" Java programming, involving arrays and loops.

1.1 Write a class called Calculator which has an int instance variable to represent the current stored value. The constructor should set that value to zero. It should have a method void add(int other) which adds another value to the one stored, and a method void display() which prints the current stored value. (5 marks)

```java
public class Calculator {
    private int _value;

    public Calculator() {
        _value = 0;
    }

    public void add(int other) {
        _value = _value + other;
    }

    public void display() {
        System.out.println(_value);
    }
}
```

1.2 Now we will change this into a slightly unusual calculator in that it will store several values at once. Replace the int instance variable with an array of ints. Have the constructor take the required size of the array as its parameter, and create the array. Hint: when you create an array of ints, the values in it are all zero by default, so you don't need to set them. Change the add() and display() methods so that each takes another int parameter which says which value in the array to add to or display. You don't need to check that the parameter is valid. (5 marks)

```java
public class Calculator {
    private int[] _values;

    public Calculator(int size) {
        _values = new int[size];
    }

    public void add(int other, int index) {
        _values[index] = _values[index] + other;
    }

    public void display(int index) {
        System.out.println(_values[index]);
    }
}
```
private int[] _values;

public Calculator(int size) {
    _values = new int[size];
}

public void add(int other, int index) {
    _values[index] = _values[index] + other;
}

public void display(int index) {
    System.out.println(_values[index]);
}

1.3 Write a method int maximum() which will return the maximum value stored in the array. You can assume that this value will not be negative. (5 marks)

public int maximum() {
    int max = 0;
    int i = 0;
    while (i < _values.length) {
        if (_values[i] > max) {
            max = _values[i];
        }
        else {
            // Do nothing
        }
        i++;
    }
    return max;
}

or, with a for-each loop:

public int maximum() {
    int max = 0;
    for (int value : values) {
        if (value > max) {
            max = value;
        }
        else {
            // Do nothing
        }
    }
    return max;
}

Question 2

This question is about inheritance. It involves writing code, but you only need to write the code which exactly corresponds to the items mentioned in the question.
Everybody did this question.

2.1 Explain how inheritance in programming languages is related to the way we manage complexity in the real world, and what its main advantages are. (4 marks)

Bookwork (see week 7 lecture notes and slides)

This was generally answered well, the main problem was stating four different points (I accepted anything relevant to the question)

2.2 Draw a UML class diagram which shows an abstract class TimeDisplayDevice, with subclasses Clock and Watch. (3 marks)

![UML Diagram](image)

Correct use of UML notation - the triangle to represent inheritance - and no extra clutter required for full marks.

Most people got this correct – better than my second year CS students in fact – well done!

2.3 Show how the three classes would be declared in Java. (3 marks)

```java
public abstract class TimeDisplayDevice { ... }

public class Clock extends TimeDisplayDevice { ... }
```
public class Watch extends TimeDisplayDevice {

1 mark for the abstract class roughly right, one for the subclasses likewise, full marks for exactly correct syntax.

*Most people also got this right, or very nearly.*

2.4 Write the TimeDisplayDevice class, assuming that:
- It represents the time with two ints, for the hours and minutes. Initial values for these are provided as parameters to the constructor
- There are public methods to get the hours and minutes
- There is a method to display the time as a String. This method will be implemented differently in each subclass. (5 marks)

```java
public abstract class TimeDisplayDevice {

    private int _hours, _minutes;

    public TimeDisplayDevice(int hours, int minutes) {
        _hours = hours;
        _minutes = minutes;
    }

    public String getHours() {
        return _hours;
    }

    public int getMinutes() {
        return _minutes;
    }

    public abstract String displayTime();

}
```

Again, partially correct solutions to be marked generously.
Here is course is where it all fell down. Most solutions had enough along the right lines to get a few marks. Congratulations to the one person who got it spot on,
Question 3

This question is about **collections**.

*Nobody attempted this question!*

3.1 State TWO ways in which an ArrayList is similar to an array (2 marks)

**Both are ordered sequences of values**
**Both allow the value at an index to be retrieved directly and efficiently**
Marks also for other correctly stated similarities

3.2 State FOUR ways in which an ArrayList is different from an array (4 marks)

**ArrayLists can change their size after they're created**
**Arrays can contain primitive types as well as objects**
**Arraylist is a library class, arrays are built in**
**Arrays have special syntax, Arraylists just use message sends**

3.3 Assume you have a `Train` class, and each `Train` has a unique ID and a current position, accessed by the methods `String getID()` and `Position getCurrentPosition()`. Briefly explain what the following method does. (4 marks)

```java
public Position find(Train[] trains, String trainID) {
    Position result = null;
    int i = 0;
    while(i < trains.length) {
        if (trains[i].getID().equals(trainID)) {
            result = trains[i].getCurrentPosition();
        } else {
            // Not found yet, do nothing
        }
        i++;
    }
    return result;
}
```

It finds the train with the given ID, if any, by running through an array of trains, and comparing the ID of each with the required ID. If a match is found the variable result is the position of the train in question, which is returned at the end.

3.4 If instead, the condition was written

```java
if (trains[i].getID() == trainID)
```

it would not work correctly. Explain why not. (1 mark)

This compares for identity, - whether the two ID strings are the same object, not equality - whether they contain the same characters, and so won't work unless the happen by coincidence to be identical
Actually it’s worse than that - because of the way the Java compiler stores strings, it probably will work - most of the time - depending on how the strings are created. However, this is not something these students should need to know about.

3.5 Explain how, by using a suitable collection, we could avoid using a loop altogether. (It’s not necessary to write the actual code so long as the explanation is clear). Hint: an ArrayList is not a suitable collection for this job. (4 marks)

By using a Map<String, Train>

where the String is the trainID. Then we could use containsKey() to check that the ID matches and actual train, and get() to get the corresponding Train object so we can ask it its position. This would be far more efficient if we had lots of trains.

Generous marks for anybody who gets the right general idea.

Question 4

This question is about storing external data.

Note: most students will do this question as it doesn’t require writing code and it should be much easier for them than the other questions. Hence less generous marking.

Yep, almost everybody did this!

4.1 Give a brief outline of how, in a Java application, you would store data about trains, such as owner, capacity, position etc. using each of the following formats.

a). Java serialised objects (2 marks)

We simply declare an ObjectOutputStream and write Train objects to it and the objects are with their positions etc. are converted to bit streams automatically.

b) Comma-separated value (CSV) files.

These are simple text files, where typically we would have represent one train object on a line, with attributes such as owner etc. separated by commas.

c) XML files

We would define, using a DTD or schema, a format with tags corresponding to the attributes of trains, and store them in text files in this format. This would be better than b if the data is more structured.
d) A relational database

We would define tables to represent trains etc. and use JDBC to manipulate the database from Java.

For full marks, answers should be specifically related to the example.

I actually accepted quite a lot of different answers, provided they related to the example and clearly showed that they understood what the formats were.

4.2 Suppose the reason you are storing this information is that you are building an information system for Network Rail. Which TWO formats would you seriously consider using, and why? (4 marks)

Since this is a large and complex system, only a relational database or XML are worth considering.
A database would enable us to store the data securely, and query it in lots of different ways, as well as supporting transactions etc. XML would make it easy to present the information in different ways, and may help with dealing with data in different formats (held by different train companies for instance). If I was implementing this for real I’d probably use a database as the main means of storage, but XML might well also have a role.

Most people got this right.

4.3 Suppose the reason you are storing this information is that the trains are not real ones (phew!) but models owned by enthusiasts, each of whom will have at most a few tens of them, collected over a period of years. Which TWO formats would you seriously consider using, and why? (3 marks)

In this case a database would be overkill, requiring the enthusiasts to have access to a database server. Serialisation isn't really appropriate for long-term data storage. XML is a contender, as it is supported directly by Java, and is very flexible. CSV could also be considered, as even XML might well be overkill The choice between XML and CSV would be made on the basis of how complicated the information to be stored was (e.g. if it included a detailed breakdown of the components of each train, XML would cope better than CSV). For this simple application, it's unlikely that more than one form of data storage would be necessary.

Again, a fair number of people got this right, although several suggested a database (not very practical for the reason stated above) and others suggested serialization – never a long term data storage option.