1. (a) How does Object Oriented Programming facilitate the creation of reliable, reusable, extensible and adaptable code? [4]

(b) What is Encapsulation and what does it mean to say that C^{++} friend functions "break encapsulation"? [2]

(c) Give one advantage and one disadvantage of the Waterfall development process model? [2]

(d) Identify two techniques used for identification of candidate classes during modelling? [2]

(e) What is a Derived Association and why is it beneficial to use it when appropriate? Give an example of a Derived Association using UML. [3]

(f) The guiding principles of good Object Oriented design relate to Coupling and Cohesion. What do these two terms mean and what are the guiding principles? [4]

1. (g) The following questions are all based upon observation of the Borland C^{++} Builder (BCB) Integrated Development Environment:

(i) Identify the architectural style used in compiling a C^{++} source file under BCB? [1]

(ii) Identify two tangible aspects of BCB that provide the Framework? [1]

(iii) Identify one other architectural style found in BCB? [1]

2. (a) C++ programs make extensive use of libraries. What mechanism does C++ provide which enables a client module to respond to an error generated in a service module, when the client source code is not available? Illustrate your answer with a short extract of C++ code for handling such errors in the service and client modules. [5]

(b) Use the class diagram shown in Figure 2.1 to write C++ source code for defining the three classes as they should appear in an interface file. The UML stereotypes provide additional C++ specific implementation details [5]

(c) (i) Why might it be a good idea to create a wrapper class e.g. `TComplex` which just uses a Standard Template Library class e.g. `std::complex`, why not use the latter directly? [2]

(ii) Modify the C++ `TComplex` class definition so that the data members of the class `std::complex` use templates. You may write stubs for the body of the member functions. [3]

(iii) Write a short C++ code extract showing how `TComplex` objects can be created with different actual datatypes for the data members of the class `std::complex` e.g. `float`, `double` and `long double`. [1]

(d) What is multiple inheritance? Give an example of a C++ class definition involving multiple inheritance. Illustrate your answer with a UML diagram. [4]

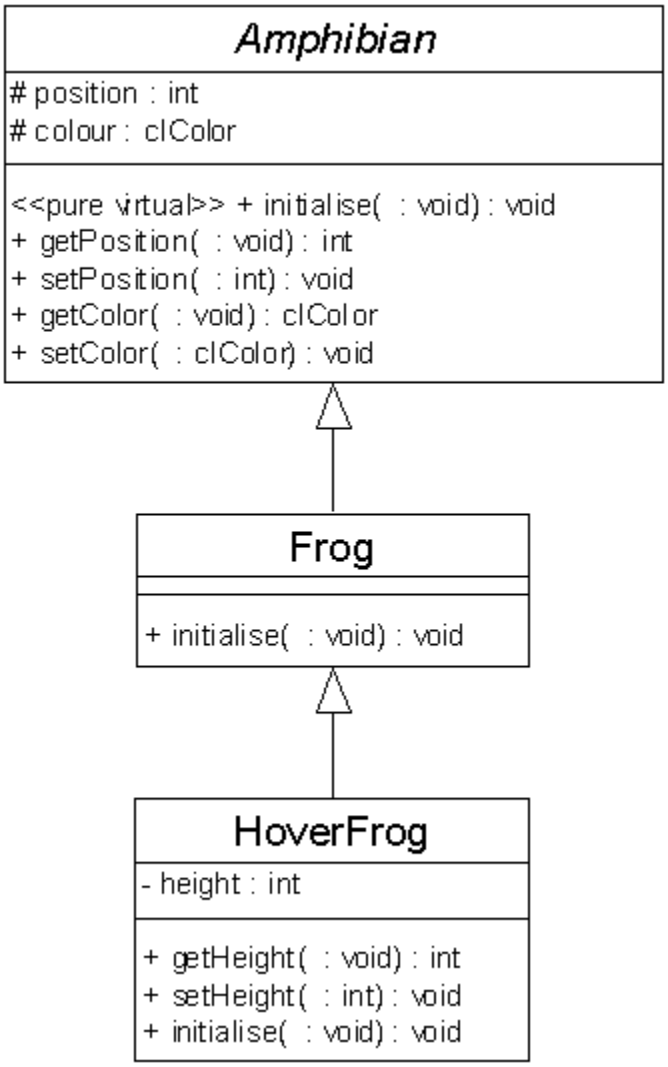

Figure 2.1

```
class TComplex{
  public:
    TComplex(double r, double i):n(r,i) {}

    complex<double> add(TComplex n1){
            return this->n + n1.n; }

    complex<double> minus(TComplex n1){
            return this->n - n1.n; }

  private:
    complex<double> n;
};
```

Figure 2.2

3. (a) Perform a textual analysis of the text below to identify the classes then draw a Class-Association diagram: [6]

> A 5.1 home cinema entertainment system consists of an amplifier which may be connected to many kinds of player devices. Player devices include Dvd and Satellite boxes. The audio output from the players connects to the amplifier which is in turn connected to six speakers. The speakers are classified as front, rear, centre and subwoofer. The video output from the players is connected to a TV.

(b) Draw a Use-Case diagram from the perspective of a user of a standard laboratory oscilloscope. Include at least three Use-Cases. You can assume that the system will not be implemented in software. [2]

(c) Identify three different omitted features of Figure 3.1 that suggest that this *is* an Initial Object Model i.e. there has not yet been any consideration of aspects of dynamic behaviour: [3]

(d) Convert the Sequence Interaction diagram shown in Figure 3.2 into a Sequence Collaboration diagram: [5]

(e) Convert the C^{++} code shown in Figure 3.3 into a class-association diagram: [4]

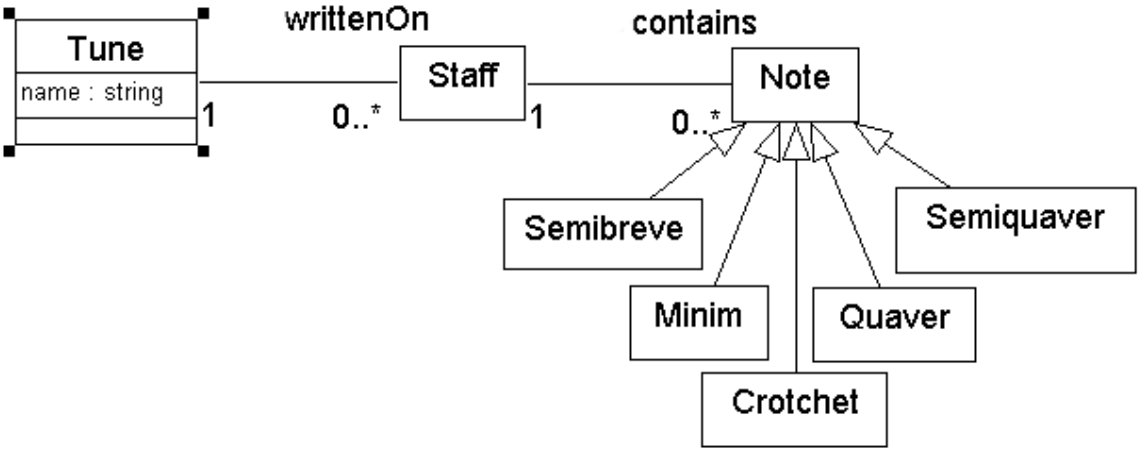

Figure 3.1

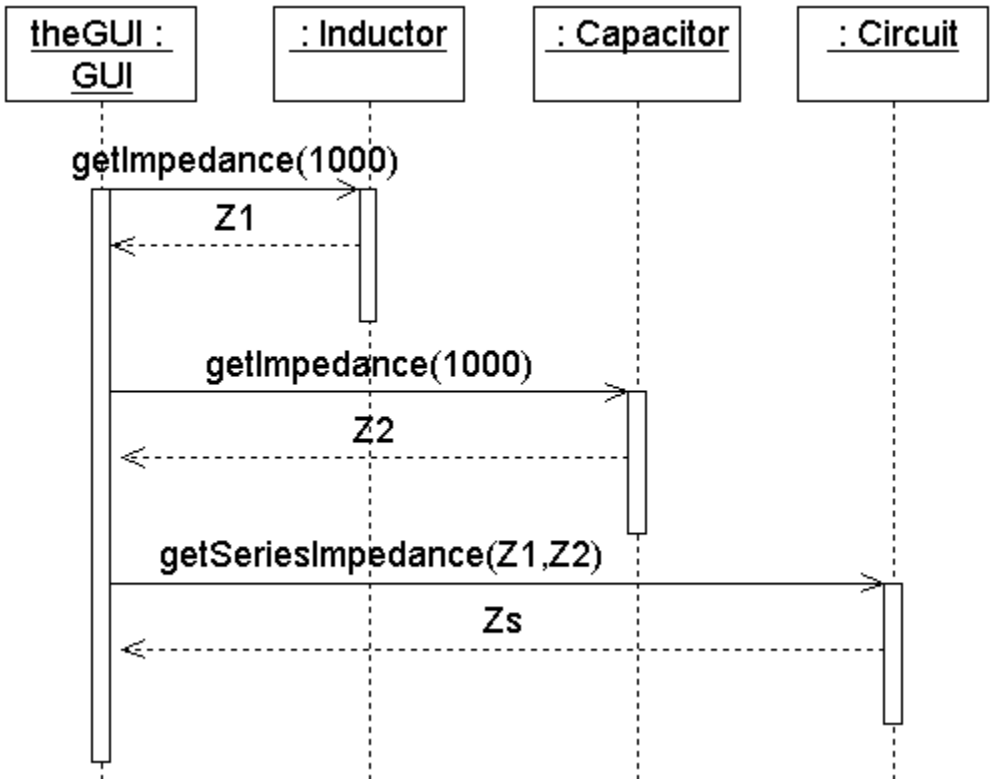

Figure 3.2

```
class Point { };
class Vector {
  public:
     Point   *includes;
};
class GeometricVector : public Vector {};
class ComponentVector : public Vector {};
class DisplacementVector: public GeometricVector{};
```

Figure 3.3

4. (a) Briefly, describe the Observer design pattern and the problem it solves. [4]

(b) Microsoft's SOAP implements the client-server model architecture with message passing in XML format. Infer what this means firstly in terms of performance and secondly in terms of ease of adapting the protocol. [4]

(c) The Java program shown in Figure 4.1 is a very basic client application. Use a UML activity diagram to highlight the key elements of this client application, which is typical of any client in a client-server architecture. [4]

(d) The XML DTD shown in Figure 4.2 provides a partial description for a for a time or frequency domain signal.

(i) Suggest three potential benefits that might be gained from describing such data in XML. [3]

(ii) Using the DTD in Figure 4.2 write some XML data for time or frequency domain data. [5]

```
int aPortNumber = 7;
String aHostName = "127.0.0.1"

s = new Socket(aHostName,aPortNumber);

br = new BufferedReader(new
                InputStreamReader(s.getInputStream()));
pw = new PrintWriter(s.getOutputStream(),true);

String message = "Simple client demonstration";
System.out.println("Message sent was: "+message);
pw.println(message);

String reply = br.readLine();

System.out.println("Message received was: " + reply);

br.close();pw.close(); s.close();
```

Figure 4.1

```
<!-- DTD for a list of signals -->
<!DOCTYPE SIGNALLIST[<!ELEMENT SIGNALLIST (SIGNAL)*>

<!ELEMENT SIGNAL
       (Description, Domain, NumberDataValues, DataValues)>

<!ELEMENT Description (#PCDATA)><!-- signal description -->
    <!ELEMENT Domain (#PCDATA)>  <!-- TIME or FREQUENCY -->
    <!ELEMENT NumberDataValues (#PCDATA)><!--an integer -->
    <!ELEMENT DataValues (#PCDATA)> <!-- set of numbers -->

    <!ATTLIST DataValues Complex CDATA #REQUIRED>
                                        <!--"YES" or "NO" -->
] >
```

Figure 4.2