IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2006

EEE/ISE PART II: MEng, BEng and ACGI

Corrected Copy

SOFTWARE ENGINEERING 2

Friday, 2 June 2:00 pm

Time allowed: 2:00 hours

There are FOUR questions on this paper.

Q1 is compulsory.
Answer Q1 and any two of questions 2-4.
Q1 carries 40% of the marks. Questions 2 to 4 carry equal marks (30% each).

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s): L.G. Madden,
Second Marker(s): J.V. Pitt,

1 (a) (i) Name the four members of the Orthodox Canonical Class Format (OCCF)? Illustrate your answer with C++ specification code for each of the four members for a class called `XYZ`.

(ii) Would you expect all the OCCF members to be implemented in an abstract class? Explain your answer. [8]

(b) Briefly, compare and contrast the "Design by Contract" and "Defensive Programming" design styles in the context of designing code that delivers a useful service. Illustrate your answer by providing a short example of C++ service code (i.e. not client) for *both* the design styles. [6]

(c) In Figure 1.1 list the class members by name for each of the four classes. [8]

(d) Briefly, compare and contrast both "deep copy" and "shallow copy" in the context of a class that has an object as a data member. Explain how an inappropriate choice of copy mechanism might compromise encapsulation [4]

(e) Consider a program that reads text from a file containing C++ source code. The purpose of the program is to remove all the text associated with C++ comments. Single-line comments follow a pair of forward-slash characters (i.e. //) and multi-line comments start with the pair of characters `/*` and end with the pair of characters `*/`

Mixed type comments and nested comments are not allowed. A file pointer references a character in the text file and is used to move through the contents of the file referencing each character sequentially.

(i) Identify three states for the file pointer described above.

(ii) Draw a UML state diagram, using the three states and annotate the transitions between states. [8]

(f) In Figure 1.2, what are the values of:

(i) the function formal arguments `x` and `y`, on entry to the function but *before* execution of the first statement in the function,

(ii) the function formal arguments `x` and `y`, on exit to the function *after* execution of the final statement in the function

(iii) the main program local variables x and y *after* execution of the function. [6]

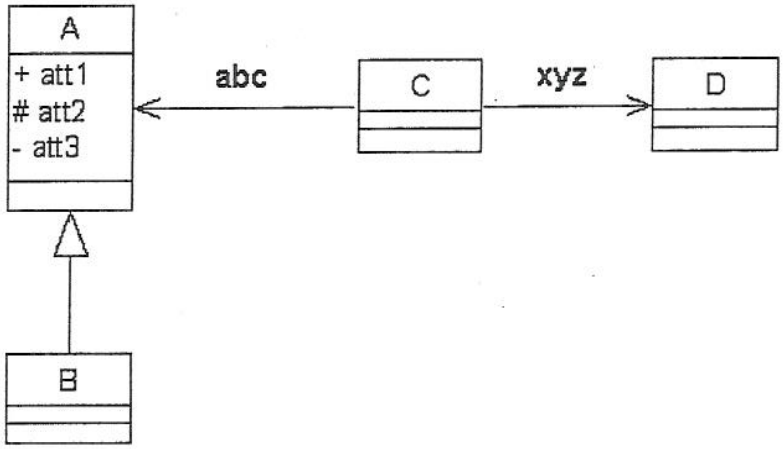


Figure 1.1

```
int obfuscate (int &, int);

int main(void){
  int x, y;
  x = 2;
  y = 2;
  x = obfuscate(y, x);
}

int obfuscate(int &x, int y){
  x = x+1;
  y = y-1;
  return x+y;
}
```

Figure 1.2

2 (a) Examine the code shown in Figure 2.1. Since the C language is a subset of the C++ language, identify the syntactic elements in this example that demonstrate that this is specifically C++ code and not shared syntax with C code. [8]

(b) Examine the C++ member function specification shown in Figure 2.2. Describe the purpose of each of the three ways that the keyword `const` has been used in the specification. Use *left*, *middle* and *right* to identify which of the 3 uses of `const` you are describing. [6]

(c) Examine the C++ ordinary function implementation shown in Figure 2.3.

(i) Briefly, describe the purpose of the function and the specific C++ feature that makes this code adaptable. Provide at least two lines of C++ code to show how the function can be used.

(ii) Explain the purpose of each line in the implementation, referring to the line numbers shown in the figure. [8]

(d) Examine the C++ code extract shown in Figure 2.4.

(i) In the context of the C++ language, which has a high dependence on the use of library based code, explain *how* the code extract demonstrates one approach for handling errors. Briefly, explain *why* this approach is important in this context.

(ii) Write implementation code for the exception class in the example called `FileNotFoundException`. The exception should record the name of the file that could not be opened. [8]

```
#include <fstream>
#include <complex>
using namespace std;
void main(void){
  ifstream ifs1;
  double re,im;
  ifs1.open("a.txt");
  if (ifs1.is_open()){
      ifs1 >> re >> im;
      if (ifs1.good()){
          complex<double> z(re,im);
          cout << z.real();
      }
      ifs1.close();
  }
  else
    cout << "File not opened";
}
```

Figure 2.1

```
const TComplex TComplex::addComplex(const TComplex &) const;
```

Figure 2.2

```
using namespace std;                    // 1
#include <string>                       // 2
#include <sstream>                      // 3
template<typename T>                    // 4
T fromStr(const string &s) {            // 5
  istringstream iss(s);                 // 6
  T x;                                  // 7
  iss >> x;                             // 8
  return x;                             // 9
}
```

Figure 2.3

```
try {
// attempt to open a file for processing here, assume
// the filename is stored in a string called aFilename
}
catch(FileNotFoundException e) {
      cout <<"FileNotFoundException:" << e.what();
}
```

Figure 2.4

3 (a) Perform a textual analysis of the text shown below. Draw a UML class diagram, based upon the information shown below, including multiplicities and names for any associations.

> Computer programs are written in a programming language. Imperative, Object Oriented, Functional and List Processing are all kinds of programming language. Each program consists of many statements. Sequence, Selection and Iteration are all kinds of statement. [8]

(b) Figure 3.1 shows six class diagrams labelled (a) - (f). For each of the following descriptions, state which is its equivalent diagram:

(i) A student takes many exams; a student also has a highest exam mark.

(ii) All students in the department study mathematics; an ISE student is a student that also happens to study Software Engineering.

(iii) The student year representative is a student

(iv) A student takes many exams; each is uniquely identified by an exam identifier number.

(v) A student may borrow a CD from the programming helpdesk many times. For each loan certain details must be recorded e.g. the CD number, the CD title, the date of the loan etc...

(vi) ISE and EEE students are a kind of student. [6]

(c) (i) The MS-Windows desktop environment typically contains a Recycle Bin icon. Any files or folders dragged onto the icon are deleted. The Macintosh desktop has a similar icon with similar functionality but is also used for ejecting floppy disks. This violates at least one design principle. Identify a violated principle and explain why it has been violated.

(ii) Identify three more design principles and briefly explain their meanings. [8]

(d) (i) Contrast and compare an association between classes and a dependency between classes.

(ii) Describe all the ways in which dependencies between classes can exist. [8]

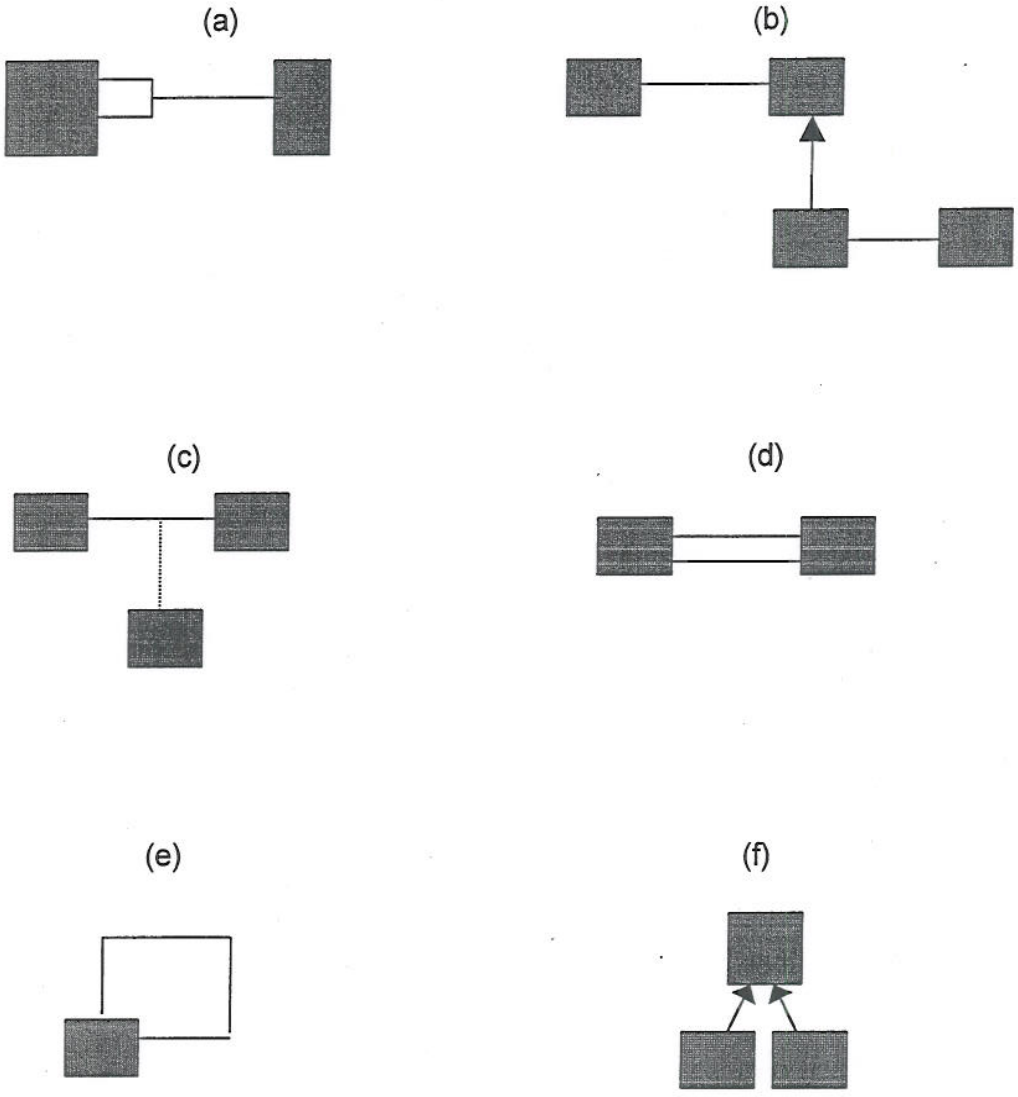


Figure 3.1

4. This question is based on the following Requirements Specification:

IP addresses are currently expressed in a dotted decimal notation, e.g. 155.198.123.115, and are divided into 5 separate classes:

Class A addresses range from 1.0.0.0 to 127.255.255.255
Class B addresses range from 128.0.0.0 to 191.255.255.255
Class C addresses range from 192.0.0.0 to 223.255.255.255
Class D addresses range from 224.0.0.0 to 239.255.255.255
Class E addresses range from 240.0.0.0 to 255.255.255.255

A program is required that will input 4 whole numbers representing a dotted decimal IP address. The purpose of the program is to output a message indicating that the address is class A or B or C or D or E. A reusable component is available to convert between strings and numbers. Additionally, the program should check for typical input errors e.g.

- any of the four numbers is not in range e.g. negative
- one or more values is a non-numeric value
- one or more values is missing at the moment of evaluation.

(a) (i) For each of the Figures 4.1, 4.2 and 4.3, describe the evidence (if any) that "Design by Contract" has been adopted (in any part of the figure).

(ii) Describe how "Correctness" and "Quality" can result from adopting "Design by Contract". [8]

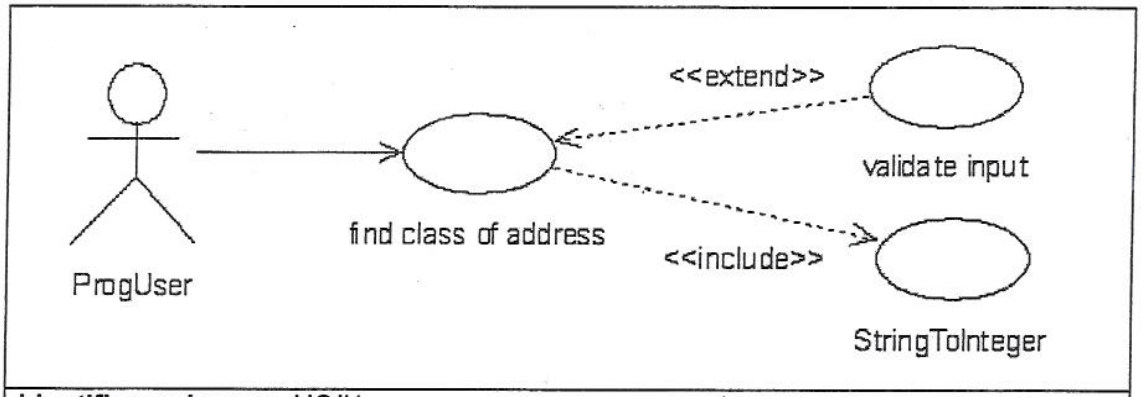
(b) The class diagram shown in Figure 4.2 shows an orchestrating instance, (i.e. `AddressSystem`) in addition to the class derived from the requirements specification above (i.e. `DottedIntegerAddress`).

(i) What is the difference between an "Initial Object Model" and an "Object Model"?

(ii) Does Figure 4.2 represent an "Initial Object Model" or an "Object Model"? Provide evidence from the diagram to justify your answer. [6]

(c) Convert the Sequence Collaboration diagram in Figure 4.2 into a Sequence Interaction diagram. [8]

(d) Using your answer to part (c), write implementation code for the member function `findAddressClass()` in the class `AddressSystem`. You may assume the function `int strToInt(string)` is available for conversion from a string containing only numeric characters into an integer type variable. [8]



Identifier and name	UC#1
Initiator	ProgUser
Goal	Determine the class of the I/P address
Pre-condition	none of the four values are not in range e.g. negative none of the four values are non-numeric e.g. a none of the four values are missing at the moment of evaluation.
Post-condition	Returns that class is A or B or C or D or E

Figure 4.1

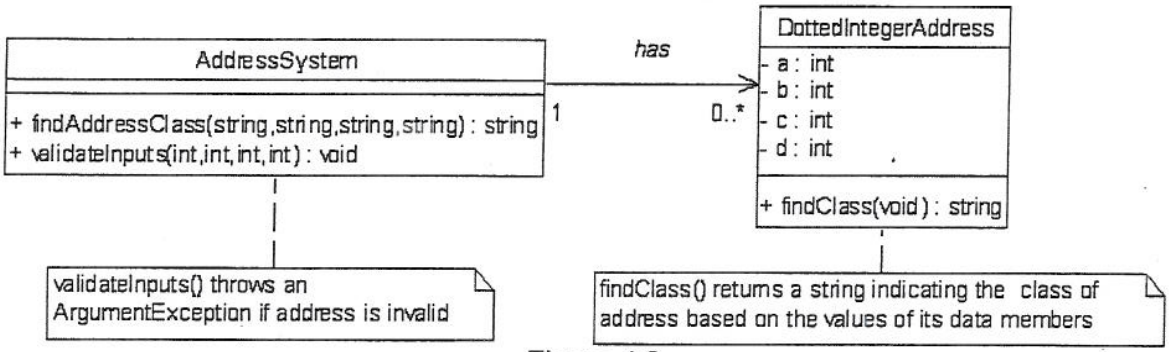


Figure 4.2

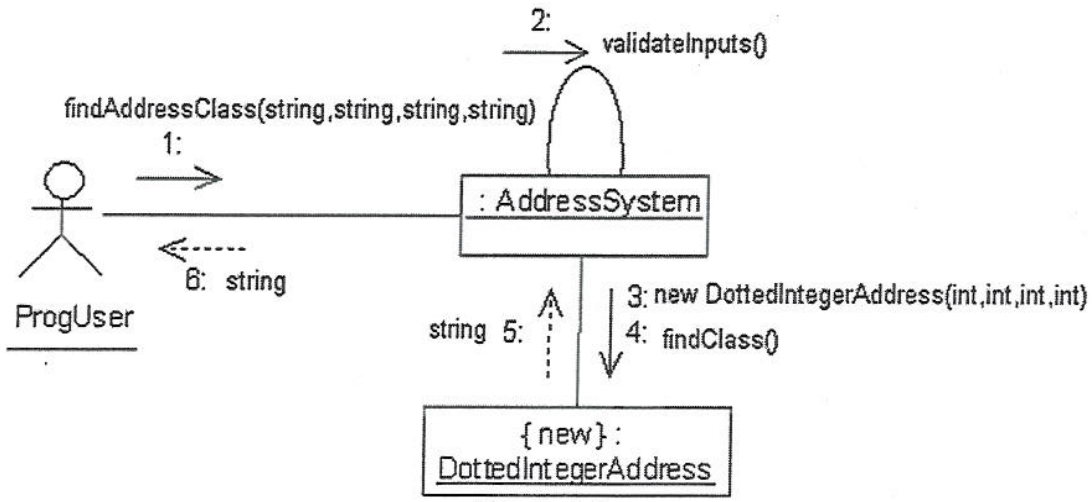


Figure 4.3

Master Solutions – June 2006

Software Engineering 2

A1(a) "bookwork" [8]

(i)

- a default (no arguments) constructor e.g. `XYZ(void)`
- a copy constructor e.g. `XYZ(const XYZ &)`
- a destructor - even if empty e.g. `~XYZ(void)`
- an assignment operator e.g. `XYZ operator=(const XYZ &)`

`names [2], code [4]`

(ii) Can not create instances of an abstract class thus can not copy the state of an object that can never be created, so in theory we do not need to create the copy constructor or the assignment operator. [1] Similarly it there is no need for code to be associated with construction and destruction. However, an abstract class does provide some implementation code for the benefit of derived (concrete) classes so it is possible that some or all of the OCCF should have implementation code. What is sought here is any reasoned argument similar to the above. [1]

A1(b) "bookwork" [6]

DbC and DP are alternative designs for handling errors. [1]

In DP it is the responsibility of the author of the service module to deal with errors [1] but in DbC it is the responsibility of the author of the client module. [1]

DP	DbC

`// service code`
`int peek(Stack s){`
`  if (!isEmpty(s)){`
`  return s[size(s)-1];`
`  }`
`  else {`
`// error handling would be here`
`  }`
`} [2]`
	`// service code`
	`int peek(Stack s){`
	`  return s[size(s)-1];`
	`}`
	
	`[1]`

A1(c) "new computed example" [8]

	Data members
A	att1, att2, att3,
B	att1, att2
C	abc, xyz
D	None

Class A has 1 private, 1 protected & 1 public i.e. **3** [3]
Class B inherits public and protected i.e. **2** [2]
Class C has 2 unidirectional associations i.e. **2** [2]
Class D has **0** DM [1]
Extra members [-1]

A1(d) "bookwork" [4]

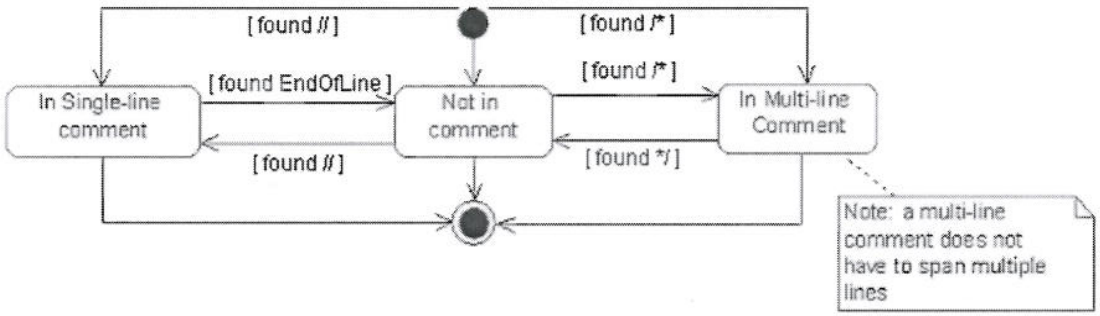
Both deep and shallow copy are concerned with copying the state of an object which has at least one data member that is an object i.e. not a primitive.[1]

A shallow copy effectively copies the reference to the original object [1] which leads to more than one route to access the state of original object, the original and copy reference the same object. This has the threat that encapsulation could be compromised by the copy having a different visibility than the original. [1]

A deep copy makes a new instance of the class and uses getter methods in order to extract the state of the object to be copied to copy to the new instance. [1] Thus the original and copy reference independent objects with the same state.

A1(e) "new computed example" [8]

States: "Not in comment", "In single-line comment", "In multi-line comment" [3]



3 States [1], Initial state and transitions [1], Final state and transitions [1]
Transistions between NIC and InSLC [1], Transisitions between NIC and InMLC [1]

A1(f) "new computed example" [6]

```
int obfuscate (int &, int);

void main(void){
  int x, y;
  x = 2;
  y = 2;
  // x = 2, y = 2
  x = obfuscate(y, x);
  // x= 4, y = 3
  // x= 4 i.e. obfuscate (), y = 3 i.e. obfuscate::x (iii)  [2]
}
int obfuscate(int &x, int y){
// x = 2, y = 2 (i)                                          [2]
  x = x+1;
// x = 3, y = 2
  y = y-1;
// SBR x = 3, SBV y = 1 (ii)                                 [2]
  return x+y;
// obfuscate = 4
}
```

A2(a) "new computed example" [8]

C++ syntax includes:
notation for header files
namespace
stream classes for I/O
cout variable
std classes e.g. complex, fstream
OO dot notation
invoking member functions on objects
constructors
overloaded stream operators
file error handling
template class [1] per observation

A2(b) "bookwork" [6]

Left: Making the function return value **const** prevents the result from becoming the target of an assignment e.g. **z1.addComplex(z2) = z3** [2]

Middle: For reasons of efficiency we use a reference variable for the formal argument **tc** which is an alias to **z2** in **z1.addComplex(z2)**. Making the reference variable **const** denies the possibility of changing the state of **z2** via **tc**. [2]

Right: Only member functions made **const** are allowed to send messages to a temporary (**const**) object. Temporary objects are often to avoid unececssary declarations, e.g. From Lab #4

TCircuit filter = TCircuit(r1,TCircuit(l1,c1).series()) [2]

A2(c) "new computed example" [8]

(i) converts a string containing *only numeric characters* into the *appropriate primitive type* initialized to the numeric value contained in the string [1]. Uses the template feature of C++ [1]

```
string s = "123.0"
double d = fromStr <double>(s);
int i = fromStr <int>(s);
```

(ii)

```
using namespace std;
# 1 needed because the standard library is wrapped in the std namespace     [1]
#include <string>       # 2 provides access to standard library string class
#include <sstream>      # 3 provides access to std. lib. string stream classes
template<typename T>    # 4 declare a generic type for use in function
T fromStr(const string &s){
# 5 input is a constant string passed by reference,output is a generic type [1]
istringstream iss(s);   # 6 invoke 1-arg. input string stream constructor   [1]
  T x;                  # 7 placeholder for result of primitive type        [1]
  iss >> x;             # 8 convert input string stream to primitive        [1]
  return x;             # 9 return primitive value
}
```

A2(d) "bookwork/new computed example" [8]

(i) Try-catch blocks are used where the source of the problem is in inaccessible in service/library code whilst the source of the solution is in the client code. The service code throws an exception and it is up to the client to decide how to deal with it.

The client module try-block contains code for normal program operation. Occasionally when calling a service routine (e.g. open file), it will fail due to some transient environmental reason (e.g. disk unavailable, filespace quota exceeded)

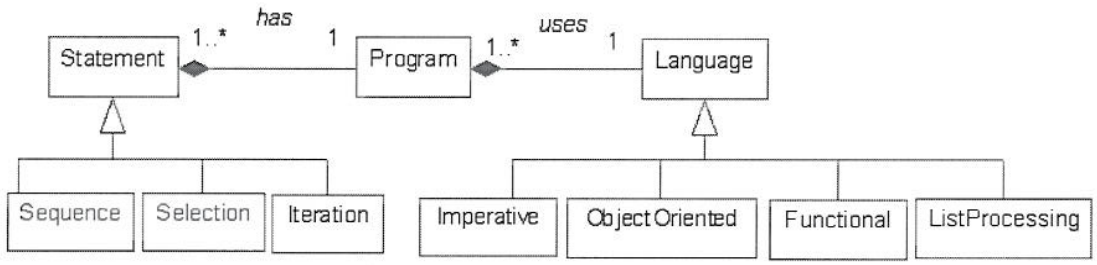
The service module detects the problem and throws an exception which retains some status information as the state of the exception object. The client module catch-block contains code for abnormal operation. The client may then inspect the state passed via the exception object and respond appropriately. [4]

(ii)

```
class FileNotFoundException : public runtime_error{
public:
  FileNotFoundException(string s):runtime_error("File "+
                                        aFilename +" not found"){};
};
```

[4]

A3(a) "new computed example" [8]



Association multiplicities: [2]
Association names: [2]
Association (Consists-of optional): [2]
Generalisation: [2]

A3(b) "new computed example" [6]

(i) A student takes many exams; a student also has a highest exam mark. (d)

(ii) All students in the department study mathematics; an ISE student is a student that also happens to study Software Engineering. (b)

(iii) The student year representative is a student (e)

(iv) A student takes many exams; each is uniquely identified by an exam identifier number. (a)

(v) A student may borrow a CD from the programming helpdesk many times. For each loan certain details must be recorded e.g. the CD number, the CD title, the date of the loan etc… (c)

(vi) ISE and EEE students are both kinds of student. (f)

A3(c) "bookwork" [8]

(i)

- Separation of Concerns [1] - each function should do one well defined and preferably small task, and no more e.g. add 2 complex numbers. [1] OR
- High Cohesion [1] - Cohesion is a measure of how closely the activities within a module are related to one another i.e. "Do one thing and do it well" e.g. complex algebra including addition, subtraction etc… [1]

(ii) 3x [1] + [1]

- Information Hiding - The combination of Encapsulation and Abstraction i.e. Abstraction/Encapsulation – client does not *need/able* to know any more than what is defined in the interface respectively.
- Least privilege - A user (programmer) should be given no more privilege than is necessary to complete the job
- Design by Contract (DbC) - The process of developing software using contracts between objects
- Low Coupling - Coupling is a measure of interdependence between modules.
- Evenly distribute responsibilities amongst classes
- Law of Demeter – used to decide whether to fork or cascade in an interaction diagram

A3(d) "bookwork" [8]

(i) An association and a dependency represent a high coupling between classes. [1]

A class that has an object as an attribute (as opposed to a primitive value) represents an association between classes e.g. a circuit contain complex components so uses complex number attributes. An association is a form of dependency, [1]

If a change to a module forces a change to another then there exists a dependency between the modules e.g. the point of intersection between 2 lines can be calculated using matrix algebra to solve the simultaneous equations. Thus there are various forms of dependency, c.f. (ii) below. [1]

(ii)

- class X inherits from class Y i.e. a generalization [1]
- class X has an attribute of class Y i.e. an association [1]
- class X uses an instance of class Y e.g.
  - as a formal argument or [1]
  - a return value for an operation [1]
- class X uses a public member of class Y [1]

A4(a) "bookwork/new computed example" [8]

(i) 4.1 text associated with Use case defines the contract in terms of a pre-condition (that which must be true before the UC) and a post-condition (that which will be true after the UC and if the pre-condition has been honoured). "Validate input" subtask in the UC diagram also alludes to the pre-condition. [2]

4.2 structurally, the existence of `AddressSystem::validateInputs()` is the pre-condition and the message answer resulting from invoking `DottedIntegerAddress::findClass`() is the post-condition. [1]

4.3 behaviourally, the invocation of `AddressSystem::validateInputs()` is the pre-condition and the message answer resulting from invoking `DottedIntegerAddress::findClass`() is the post-condition. [1]

(ii) Conditions are incorporated into all phases of development and leads to well defined interactions and mutual obligations of client and supplier. Correctness results from satisfying contracts. [2]

The contracts also provide a traceable path though the phases of program development i.e. from requirements to code. Quality results from this traceability. [2]

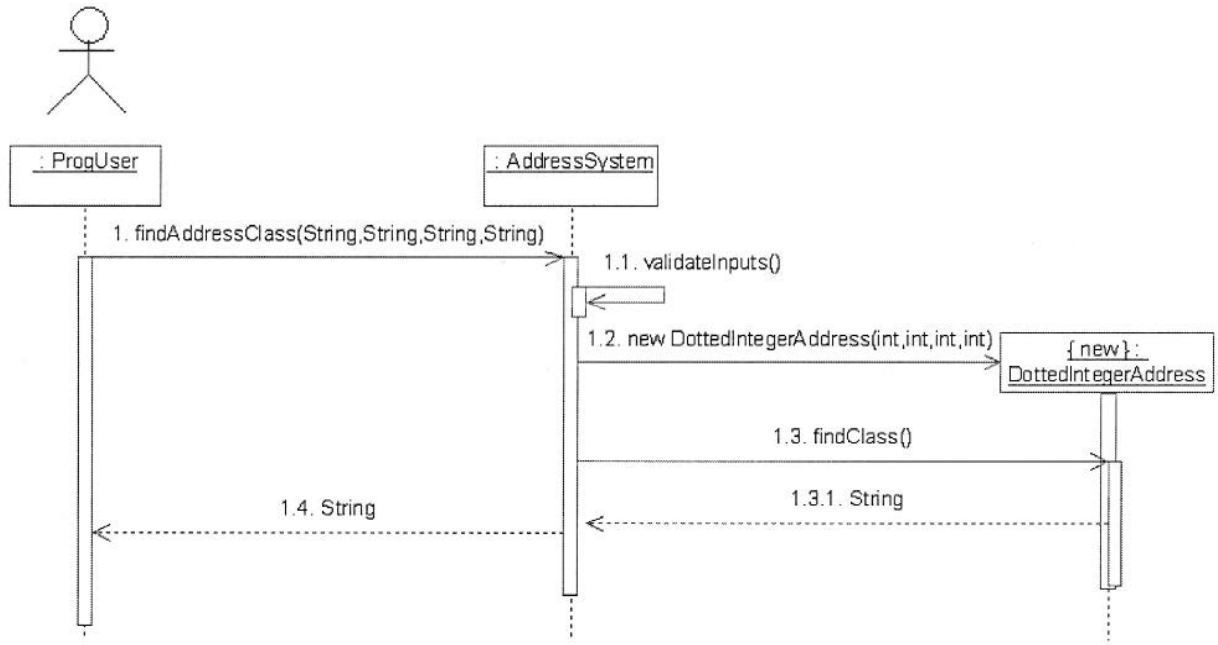
A4(b) "bookwork" [6]

(i)
The IOM is a landmark work-product, which represents enough structure to consider using the Use case scenarios to test the structure for completeness and correctness. In doing so extra detail is revealed which develops the IOM into an OM. When all the Use case scenarios have been tested then the OM should be complete and correct. Thus the IOM is starting point for performing dynamic analysis which leads to the OM. [2]

(ii)
It is an Object Model because of the existence of: [1]

- instance of the orchestrating class [1]
- association navigation arrow [1]
- operation signatures [1]

A4(c) "bookwork/new computed example " [8]



May also show `toStr()`

```
[1] ProgUser & AddressSystem
[1] message 1
[1] message 1.1
[1] message 1.2
[1] {new} DottedIntegerAddress
[1] message 1.3
[1] message answer 1.3
[1] message answer 1.4
```

A4(d) "new computed example " [8]

```
string findAddressClass(string a,string b,string c,string d){          [1]
  int w,x,y,z;                                                          [1]
   try {                                                                [1]
     w = strToInt(a); x = strToInt(b);                                  [1]
     y = strToInt(c); z = strToInt(d);
     validateInputs(w,x,y,z); // throws exception if invalid            [1]
     return DottedIntegerAddress(w,x,y,z).findClass();                  [1]
   } catch (ArgumentException  e){                                      [1]
     cout <<"ArgumentException:" << e.what() <<endl; exit(-1);          [1]
   }
}
```