UNIVERSITY EXAMINATIONS

UNISA university of south africa

January/February 2022

COS3711

Advanced Programming

80 Marks

Duration 2 Hours

EXAMINERS:
FIRST: DR CL PILKINGTON
SECOND: MR K HALLAND
EXTERNAL: DR L MARSHALL (UNIVERSITY OF PRETORIA)

This paper consists of 6 pages.

Instructions

1. You may type your answers in a word processor (and then print to PDF for submission) or handwrite your answers (and then scan to PDF).
2. Answer all questions. Please answer questions in order of appearance.
3. The mark for each question is given in brackets next to each question.
4. Note that no pre-processor directives are required unless specifically asked for.
5. Students must upload their answer scripts in a single PDF file (answer scripts must not be password protected or uploaded as “read only” files)
6. NO emailed scripts will be accepted.
7. Students are advised to preview submissions (answer scripts) to ensure legibility and that the correct answer script file has been uploaded.
8. Students are permitted to resubmit their answer scripts should their initial submission be unsatisfactory. However, only three submissions are allowed.
9. Incorrect file format and uncollated answer scripts will not be considered.
10. Incorrect answer scripts and/or submissions made on unofficial examinations platforms will not be marked and no opportunity will be granted for resubmission.
11. The mark awarded for an incomplete or illegible scanned submission will be the student’s final mark. No opportunity for resubmission will be granted.
12. Submissions will only be accepted from registered student accounts.
13. Students who do not utilise the required invigilation or proctoring tools (IRIS) will be subjected to disciplinary processes.
14. Students suspected of dishonest conduct during the examinations will be subjected to disciplinary processes. UNISA has zero tolerance for plagiarism and/or any other forms of academic dishonesty.
15. Students are provided one hour to submit their answer scripts after the official examination time. Submissions made after the official examination time will be rejected by the examination regulations and will not be marked.
16. Students experiencing network or load shedding problems are advised to apply, together with supporting evidence, for an aegrotat within 3 days of the examination session.
17. Students experiencing technical problems should contact the SCSC on 080 000 1870 or email Examenquiries@unisa.ac.za or refer to Get-Help for the list of additional contact numbers. Communication received from your *my*Life account will be considered.

Remember to complete the Honesty Declaration when submitting your answers. By submitting your answers you are confirming that your submission is your own, unaided work.

Transporting cargo around the world is essential in ensuring that customers have access to the goods they need and want.

All such items are packaged in some sort of container (which, for the purposes of this scenario, has some volume). Generally, there are two kinds of containers: (i) a box (where we want to know whether it is cube shaped or not), and (ii) a cylinder (where we want to know its diameter).

For transport, containers are packed onto pallets, and pallets are then included in a load (where each load will have a code).

Question 1 **[24 marks]**

1.1 Considering the scenario given above, draw a partial UML class diagram that captures the scenario. You should include the necessary classes, class attributes, and class relationships that are mentioned in the scenario. You do not have to include the Client/GUI class nor indicate constructors, access specifiers, or other methods in the classes you specify.
[You may use a software tool to create the UML class diagram.] (14)

1.2 Would you use aggregation or composition relationships in the design of this UML class diagram? Explain why you have or have not done so. (2)

1.3 The load code takes the following format.
- Year value between 2000 and 2099 (both included)
- Forward slash (/)
- Month value between 01 and 12 (both included)
- Forward slash (/)
- L
- A serial number starting from 1, running up to 9999

Write the regular expression (in quotes) that can be used to check that a load code meets the required criteria. An example of a valid code is `2022/01/L1`. Ensure that you use escape characters correctly. (8)

Question 2 **[40 marks]**

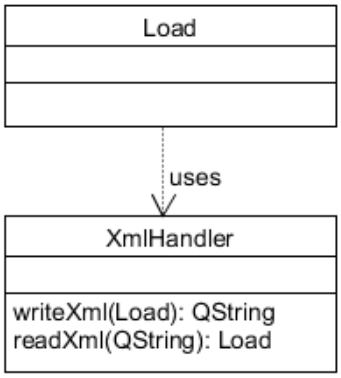
The intention is to serialise container objects using reflective programming approaches. The idea is to convert all object data in a load to XML and save this data on a network store.

2.1 What is the major benefit of using a reflective approach in this scenario? (2)

2.2 For a container object that is urgent, the users want to add a property to just this specific object. In such cases, the property name is `urgent` and its value is a message indicating its priority (such as `high priority`). Assuming that the classes are set up to allow this ability, write the code to implement this intention for an object named `obj`.
[Note that this approach can be used to add other optional properties to other objects where necessary.] (2)

2.3 Consider the requirement to serialise the load class to and from XML.

2.3.1 The following UML class diagram for the serialisation has been provided (where the `QString` is the XML text) for some `Load` class.



It has been argued that this is incorrect. Provide a better UML class diagram. (3)

2.3.2 The following partial class definition has been provided to achieve the write part of the serialisation using DOM to write to XML.

```
class XmlHandler
{
public:
    XmlHandler();
    QString writeXml(Load load);
private:
    QDomDocument doc;
};
```

The intension is to provide the following XML code. Comments on the right are for explanatory purposes.

```
<Load code="2021/01/L1">                    the load code is included
   <Pallet>                                 all containers on the pallet are added
      <Box urgent="High priority">          the type of container with message
         <volume>1</volume>                 all properties with values
         <cube>true</cube>
      </Box>
      <Cylinder>                            next container
         <volume>3</volume>
         <diameter>3</diameter>
       </Cylinder>
   </Pallet>
   <Pallet>                                 next pallet in the load
      <Box>
         <volume>2</volume>
         <cube>false</cube>
      </Box>
      <Cylinder>
         <volume>4</volume>
         <diameter>4</diameter>
      </Cylinder>
     </Pallet>
</Load>
```

Using the partial code below for the function that generates this XML text, complete the code by filling in the parts indicated. You may copy this code into your answer document and type in the necessary code.

```
QString XmlHandler::writeXml(Load load)
{
  QString loadCode = load.getCode(); // gets the load code
  QDomElement root = // set up root load tag
                     // including its load code
  doc.appendChild(root);

  foreach(Pallet* p, load) // loop for each pallet in the load
  {
    // set up pallet tags
    // as part of the root load tag

    foreach(Container* c, *p) // loop for each container
    {
      const QMetaObject *mo = c->metaObject();
      QString classname = //get the classname of the object
      QDomElement base = doc.createElement(classname);

      // handle the case where an urgent property has been added
      // to an object; not all such extra properties should be
      // added to the tag, only those named 'urgent'

      // add this base tag to its parent tag

      for (int i=1; i<mo->propertyCount(); i++)
      {
        QMetaProperty prop = mo->property(i);
        QString propertyName = prop.name();
        QString propertyValue = prop.read(c).toString();

        // create the necessary tags for the property
      }
    }
  }

  // the generated XML needs to be sent back to calling function
}
```
(18)

2.4 We come now to the part where this XML text is to be sent over a network.

2.4.1 The following partial class definition is provided for the class that will be used to send the data over the network.

```
class Serialize
{
public:
  explicit Serialize(Load l); // load object passed to constructor
  void doSerialize(); // used to transfer over the network
private:
  Load load;
};
```

Extend this class definition so that the class can be run as a thread, including all code that would be added to conform to best practice. (3)

2.4.2 Complete the following code (that you would expect to find in the client code) that will run an instance of this class as a thread. The code should start the thread and clean up afterwards.

```
Load load;
Serialize* s(new Serialize(load));
```

(6)

2.4.3 Finally, write the code for the `Serialize::doSerialize()` function that gets the XML text serialisation using the `XmlHandler` class in 2.3.2 (repeated below) and uses UDP to send it over the network using port `55555`.

```
class XmlHandler
{
public:
    XmlHandler();
    QString writeXml(Load load);
private:
    QDomDocument doc;
};
```

(6)

Question 3 **[16 marks]**

A model-view approach will be used to display a list of load codes. The following class has been proposed (which allows for the use of a memento of instances of the class).

```
class MyListWidget : public QListWidget
{
public:
    MyListWidget();
    MyListMemento* createMemento();
    void setMemento(MyListMemento* mlm);
};
```

3.1 Given the following code, write the statement that would be used to add the load instance's code to the list widget.

```
MyListWidget* mylist = new MyListWidget;
Load load;
```
(2)

3.2 Consider the code for the `createMemento()` function.

```
MyListMemento* MyListWidget::createMemento()
{
    MyListMemento *mlm(new MyListMemento);
    QStringList strlist;
    for(int i=0; i<this->count(); i++)
    {
        strlist.append(item(i)->text());
    }
    mlm->setState(strlist);
    return mlm;
}
```

Provide the class definition (that would be expected in the header file) for the `MyListMemento` class. (6)

3.3 Distinguish between the use of the serialiser and memento design patterns as they have been used in this scenario. How do they differ in terms of their ultimate purpose (apart from the fact that the design patterns are being applied to different objects)? Make sure that it is clear which pattern you are referring to in your answer. (4)

3.4 What model/view alternatives does Qt provide that could best be used in place of the `QListWidget`? (2)

3.5 Where in the scenario presented in this paper could a factory method design pattern be appropriately used? (2)