

**UNIVERSITY EXAMINATIONS**



**January/February 2021**

**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 7 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). Typing and then printing is easier to read.
2. Answer all questions.
3. The mark for each question is given in brackets next to each question.
4. Please answer questions in order of appearance.
5. Note that no pre-processor directives are required unless specifically asked for.

**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.**

**[TURN OVER]**

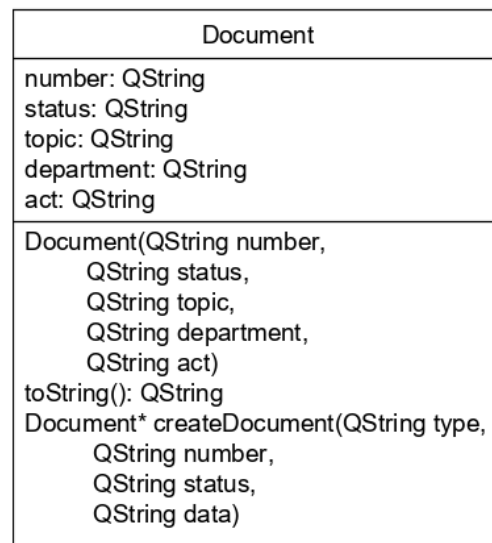
Government and state authorities produce many different kinds of documents. The questions in this paper are based on a simplified scenario for keeping a register of such documents.

Suppose that there are three types of documents that are produced: regulations (which have an Act number as they are published as acts of the relevant parliament), policies (which are issued by a particular department), and guidelines (which are formulated for a particular topic). All documents have a number and status.

### Question 1

[24 marks]

The following UML class diagram has been produced as an initial attempt at modelling this scenario.



The follow design decisions have been made:

- A classic Factory Method design pattern should be used.
- Base classes that have derived classes should be abstract.

1.1 Considering the scenario given and design decisions listed above, redesign the UML class diagram provided. You should include the necessary classes, class attributes, class constructors and operations, and class relationships to make it clear you understand how data will be managed and passed between classes. All elements in the provided UML class diagram above should be included in your design (although some of them may have to be adjusted). You do not have to include the Client/GUI class. [If you are hand drawing this answer, you can use an underline to indicate italics in the UML class diagram.] (15)

1.2 Which anti-pattern could be identified in the original design? Explain clearly why you say so. (2)

1.3 Write the implementation code (that you would expect to find in the `.cpp` file) for the function that would be used in your factory method class to create the various documents. All the data necessary to create the required objects should be passed to this function. (7)

[TURN OVER]

**Question 2****[32 marks]**

It has been decided to use a model-view approach to hold a register of all documents. The following code is offered as an initial attempt at the model part of the solution – read through the code carefully to see what approach has been proposed. Some code still needs to be provided.

This question is based on a corrected design presented in question 1.

```
// class header
class MyModel : public QStandardItemModel
{
public:
    static MyModel* MyModel();
    bool addDocument(Document* d); // to add a Document to the model
    QList<Document*> getModelData() const; // return data for writing to file
private:
    bool checkDocumentNumber(QString n); // a check function
};

// implementation detail
// constructor sets up model header row
MyModel::MyModel()
{
    QStringList headerRow;
    headerRow.append("Number");
    headerRow.append("Status");
    headerRow.append("Topic");
    headerRow.append("Department");
    headerRow.append("Act");
    setHorizontalHeaderLabels(headerRow);
}

// this function is called by addDocument() to ensure that the document
// number meets certain requirements
bool MyModel::checkDocumentNumber(QString n)
{
    // code needs to be provided here; see questions below
}

// this function is used to add a document to the model
// it uses meta-objects to find document type and properties
// after checking that the number is in the correct format
// data is added to the appropriate cells in the model
bool MyModel::addDocument(Document* d)
{
    // use the meta-object to find out what type of document d is

    QString num = // use the meta-object to find the document number

    if (!checkDocumentNumber(num)) // this code checks that num is valid
        return false;

    // add the data items to the model
    // based on what type of document it is
    // use the meta-object to access all the data in the object
```

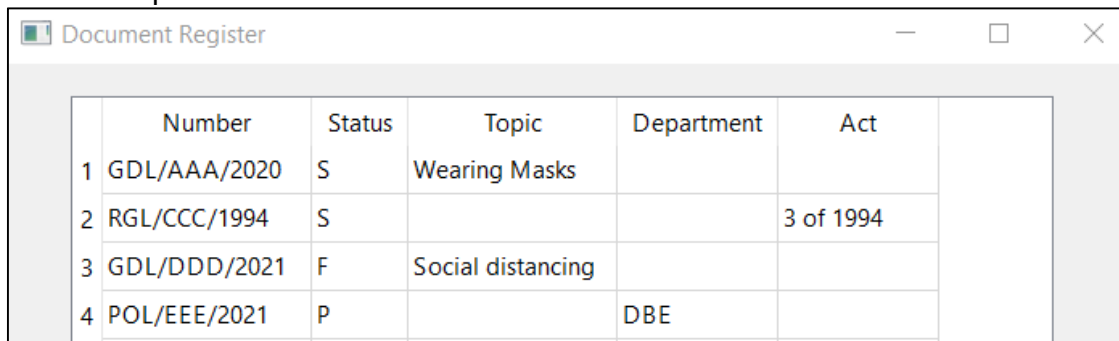
**[TURN OVER]**

```
// you need only handle the case where the document is a policy
// you may assume that all other document types are already coded

return true;
}

QList<Document*> MyModel::getModelData() const
{
    // you may assume that this has been implemented
    // it returns a QList of Document pointers from model data
}
```

Below is an example of a view of the model.



	Number	Status	Topic	Department	Act	
1	GDL/AAA/2020	S	Wearing Masks			
2	RGL/CCC/1994	S			3 of 1994	
3	GDL/DDD/2021	F	Social distancing			
4	POL/EEE/2021	P		DBE		

- 2.1 The class was supposed to be implemented following the classic Singleton design pattern. However, it is not working correctly.

Rewrite the class header as part of fixing the problems. (5)

- 2.2 You need to code the `bool MyModel::checkDocumentNumber(QString n)` function, which uses regular expressions to check that the document number meets a particular format (in the order given below).

- The first 3 characters should be GDL, POL, or RGL (for guidelines, policies, and regulations respectively).
- A slash.
- Exactly 3 arbitrary uppercase alphabetic characters.
- Another slash.
- A year value in the range from 1900 onwards.

Using a validator, check whether the `QString` passed to the function is acceptable. Return `true` if it is, and `false` if not. (14)

- 2.3 The code for `bool MyModel::addDocument(Document* d)` function is incomplete. You need to add code as indicated above so that meets the following design criteria.

- It should use meta-objects to access the data in the `Document` objects. You may assume that all objects support meta-objects.
- It should add the data from the `Document` object to the model. You need only handle the case of Policy documents.

You only need to write code for

- accessing the object data via the meta-object, and
- adding data for a policy document to the model.

You may assume that there is code for adding the other two types of documents.

[TURN OVER]

You may find it helpful to check the image of the model's view above. (12)

2.4 Which view class would you use to display the data in this model? (1)

### Question 3

[11 marks]

The model will be saved as an XML file when the application closes, and then read in again when the application opens. The following class has been designed to achieve this.

```
class ReadWriteXml
{
public:
    ReadWriteXml();
    void WriteXml(QList<Document*>);
    void ReadXml();
private:
    QDomDocument XMLdoc;
};
```

The XML file that is used is in the following format. You may assume that the tags will always be in the order in which they appear here.

```
<documents>
  <document class="Guideline" number="GDL/AAA/2020">
    <status>S</status>
    <topic>Wearing masks</topic>
  </document>
  <document class="Regulation" number="RGL/CCC/1994">
    <status>S</status>
    <act>3 of 1994</act>
  </document>
  <document class="Policy" number="POL/EEE/2021">
    <status>P</status>
    <department>DBE</department>
  </document>
</documents>
```

3.1 Which design pattern is being employed here? (1)

3.2 Has this design pattern been implemented correctly? Explain clearly why you say so. (2)

3.3 The partial code for the `ReadXml()` function that reads the XML file, recreates `Document` objects, and adds them to the model is provided below. You should use the code provided for the `MyModel` class in Question 2 as reference. Note that not all the necessary code is provided, and there are parts that you are not required to supply (it is printed in italics).

```
void ReadWriteXml::ReadXml()
{
    // open file and read all contents into a QDomDocument
    QFile file("documents.xml");
```

[TURN OVER]

```

file.open(QIODevice::ReadOnly);
XMLdoc.setContent(&file);
file.close();

// parse the QDomDocument
QDomElement root = XMLdoc.documentElement();
if(root.tagName() == "documents")
{
    QDomNode node = root.firstChild();
    while(!node.isNull())
    {
        QDomElement element = node.toElement();
        if(element.tagName() == "document")
        {
            // enter code here to access the data needed to construct
            // a Document object:
            // QString classname
            // QString number
            // QString status
            // QString otherData

            // the factory method will be used to create Document* d
            // from the data accessed above
            // you do NOT need to add this code

            // add the created Document d to the model
            MyModel::getInstance()->addDocument(d);
        }
        node = node.nextSibling();
    }
}
}

```

(8)

**Question 4****[13 marks]**

- 4.1 The document number contains three uppercase alphabetic characters, and it was decided to write a class (named `RandomLetters`) that would generate these characters randomly. Using the code below that is used to run an instance of the `RandomLetters` class in a thread, write the class header for the `RandomLetters` class.

```

QThread thread = new QThread;
RandomLetters random = new RandomLetters(3);
// passing the number of letters to generate
random->moveToThread(thread);
connect(thread, SIGNAL(started()), random, SLOT(generate()));
connect(thread, SIGNAL(finished()), this, SLOT(cleanup()));
connect(random, SIGNAL(value(QString)),
        this, SLOT(displayValue(QString)));
thread->start();

```

(8)

- 4.2 If it were decided to include memos as an additional document that should be included in the register, what changes would need to be made to the application? (3)

**[TURN OVER]**

- 4.3 A decision has been made to use cloud computing for database storage and computing power. Give two reasons why you would support such an idea in this scenario. (2)