UNIVERSITY EXAMINATIONS

UNISA university of south africa

June/July 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 5 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.
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.

Many different vaccines are produced globally to help provide vaccinated persons with immunity from specific infectious diseases. The following points apply to a simplified scenario to be used in this examination.

[1] All vaccines must be administered in some way (usually by mouth or by injection). Also, all vaccines should be able to return a string representation of its state.

[2] There are two types of vaccines: inactivated vaccines (where the infectious material has been destroyed using chemicals, heat, or radiation) and attenuated vaccines (where the process used to create them involves disabling the virulent properties or using a closely related but less dangerous organism).

[3] The cholera vaccine is a type of attenuated vaccine, and is normally administered in a number of doses. The rabies vaccine is a type of inactivated vaccine, and is normally based on a particular strain of rabies.

[4] Users should only be able to create instances of cholera, rabies, and similar types of vaccines.

[5] A classic factory method design pattern should be used to create instances of the types of inactivated and attenuated vaccines.

[6] There should be an object that contains a list of vaccine instances.

Using the scenario above, answer the questions below.

Question 1 **[20 marks]**

1.1 Using all the detail in the points [1] to [6] in the scenario above, draw a UML class diagram to model the scenario. There should be enough detail to show how data will be passed between objects; you do not need to include constructors or accessors/mutators, nor show a client/GUI class.
If the UML class diagram is being drawn by hand, use underline to indicate italics. (18)

1.2 Why would a factory method design pattern be a more appropriate design than an abstract factory design pattern? (2)

Question 2 **[24 marks]**

The process of adding a vaccine instance to the vaccine list is as follows.

a. An application is run by the main GUI/client as a separate process that gets vaccine details from the user. The data is returned to the main GUI/client as a string via the process's standard output.
b. Using the factory method, an appropriate vaccine instance is created in the main GUI/client code.
c. This instance is then added to the list of vaccines.
d. When a vaccine instance is added to the list, using inter-object communication, a signal is automatically sent that is made up of the string representation of the vaccine instance that was added (as was noted in point [1] in the scenario at the start of this exam paper).
e. This signal is then picked up by the main GUI/client and leads to the text being added to a `QListWidget` on the main GUI/client.

[TURN OVER]

The outline of the header of the main GUI/client code is as follows.

```
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private:
    Ui::MainWindow *ui;
private slots:
    void createVaccine();
};
```

2.1 An application named `vaccineData.exe` can be used to gather vaccine data from a user. Write the code that runs this application as a separate process, where its output is managed by the function in the main GUI/client named `create()`. (4)

2.2 Write the code for the `create()` function. It retrieves the output from the application process in 2.1 and calls the following factory method code to create an instance of a vaccine.

```
Vaccine*    VaccineFactory::createVaccine(QString    type,    QString
administrationMethod, QString str1, QString str2)
{
    if (type == "Cholera")
        return new Cholera(administrationMethod, str1, str2.toInt());
    else if (type == "Rabies")
        return new Rabies(administrationMethod, str1, str2);
    return NULL;
}
```

The output from the application read from the standard output is in the form `vaccineType*argument1*argument2*argument3`.

You do not need to include the code for adding the instance to the vaccine list. (8)

2.3 Write the code for the class declaration of the vaccine list class as well as the implementation code of the function that adds a vaccine instance to the list. Remember to take note of the process given at the start of this question (specifically point d). (8)

2.4 In point e in the process above, the text containing the string representation of a vaccine instance is added to a `QListWidget` on the GUI. The following code has been provided (where `ui->listWidget` is the `QListWidget` variable and `string` is the string representation of the vaccine).

```
QListWidget *newItem = new QListWidget;
newItem->setText(string);
ui->listWidget->setItem(1, newItem);
```

However, this code is not working correctly. Make the necessary corrections. (2)

2.5 Could a `QStringListModel` have been used instead of a `QListWidget`? If not, explain in detail why not; if it can, explain in detail how it could be achieved. Note that there are no marks for indicating yes or no. (2)

Question 3 **[21 marks]**

The vaccine list should be saved as an XML file when the application closes.

Assume a `MyVaccineList` class with the following interface.
`int size() const` // returns the number of items in the list
`Vaccine* at(int)` // returns the `Vaccine` instance at the specified position in the list

The XML file should have the following structure.

```
<VaccineList>
 <vaccine class="Cholera">
  <administered>Unknown administration method</administered>
  <process>Unknown process</process>
  <doses>0</doses>
 </vaccine>
<vaccine class="Cholera">
  <administered>injection</administered>
  <process>disabled</process>
  <doses>2</doses>
 </vaccine>
</VaccineList>
```

3.1 Reflective programming methods will be used to access data from vaccine instances in the list. Explain in detail what needs to be in place in the classes involved to allow this to be implementable. You should indicate clearly what needs to be done in which classes. (5)

3.2 Use DOM to write the data from the vaccine list to the XML structure given above. Consider the following declarations and partial code (where the vaccine list is passed as a pointer to the `write()` function). Remember to use reflective programming techniques so that the code is as generic as possible.

```
QDomDocument xmlDoc;
QDomElement root;
root = xmlDoc.createElement("VaccineList"); // set up root element
xmlDoc.appendChild(root);

void XmlWriter::write(MyVaccineList *mvl)
{
   // loop through all items in list
   for (int item=0; item<mvl->size(; item++)
   {
      const QMetaObject* meta = // get the meta-object

      // set up the vaccine tag
      // loop through all the instance's properties
         // access instance's properties
         // create the appropriate XML tags
    }
```

```
  // write data to file
  QFile xmlfile("vaccineList.xml");
  xmlfile.open(QIODevice::WriteOnly);
  QTextStream toFile(&xmlfile);
  toFile << doc.toString();
  xmlfile.close();
}
```

Supply the code where specifically required (indicated in bold italics) so that the data in the vaccine list is written to file. (15)

3.3 It has been suggested that SAX could just as easily be used to generate the XML file. Comment on this suggestion. (1)

Question 4 [15 marks]

4.1 An optional property may be added to a vaccine instance indicating the authorisation code of the user who added it. The code should be made up of three to five uppercase alphabetic characters followed by a hyphen and then 3 digits. An optional final character may be included.

Give the following:

(a) The input mask, indicating where this mask would be used. (3)
(b) The regular expression that could be used to ensure that a valid value is entered. (3)
(c) Assuming a vaccine instance named `vac`, add the user authorisation code `code` to the instance's properties. (2)
(d) Is it necessary to use both the input mask and regular expression to ensure a valid entry, explaining why you say so? Note that there are no marks for indicating only yes or no. (2)

4.2 The following `VaccineMemento` class has been used to provide run-time backup for the vaccine list. However, it is leading to errors. Rewrite the code so that the errors are fixed and the classic memento pattern is implemented.

```
class VaccineMemento
{
private:
    VaccineMemento();
    QList<QStringList> getState();
    void setState(QList<QStringList> v);
};
```

(2)

4.3 An often-given advantage of cloud computing is that there is no longer the need to guess infrastructure in-house capacity. Explain what this means in terms of the scenario used in this exam. (3)