UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Engineering, Built Environment and IT
Department of Computer Science

Software Modelling
COS 214

Examination Opportunity 2 (EO2)

28 September 2020

Examiners

Internal: Dr Linda Marshall and Mr Rethabile Mabaso

Instructions

1. Read the question paper carefully and answer all the questions.
2. The assessment opportunity comprises of **9** questions on **9** pages.
3. **2** hours have been allocated for you to complete this paper. An additional 30 min have been allocated for you to download the paper and upload your answer document.
4. Write your answers on a separate document (eg. using an editor or handwritten) and submit the document in **PDF format**.
5. Please make sure your **name and student number is clearly visible** in the document you upload. Provide an e-mail address and a phone number on your paper where you can be contacted, should there be any problems.
6. This paper is **take home** and is subject to the University of Pretoria Integrity statement provided below.
    - You are allowed to consult any literature.
    - You are not allowed to discuss the questions with anyone.
    - You may not copy from online resources. All answers must be in your own words.
7. If you have any queries when writing the paper, post them in good time on the ClickUP Discussion Board under the appropriate thread in the **Tests and EOs** Forum. Note, this forum is moderated and therefore your post will not be available for viewing to the COS214 class until is has been moderated.
8. An upload slot will be open on the module ClickUP page under the **Tests and EOs** menu option for the duration of the examination opportunity (17:30 to 19:30) and then for an additional 30 min to give enough time to download this paper and upload your PDF. **No late submissions will be accepted.**

Integrity statement:
The University of Pretoria commits itself to produce academic work of integrity. I affirm that I am aware of and have read the Rules and Policies of the University, more specifically the Disciplinary Procedure and the Tests and Examinations Rules, which prohibit any unethical, dishonest or improper conduct during tests, assignments, examinations and/or any other forms of assessment. I am aware that no student or any other person may assist or attempt to assist another student, or obtain help, or attempt to obtain help from another student or any other person during tests, assessments, assignments, examinations and/or any other forms of assessment.

Question:	1	2	3	4	5	6	7	8	9	Total
Marks:	5	5	4	4	4	10	39	9	4	84

Full marks is: 80

Short questions

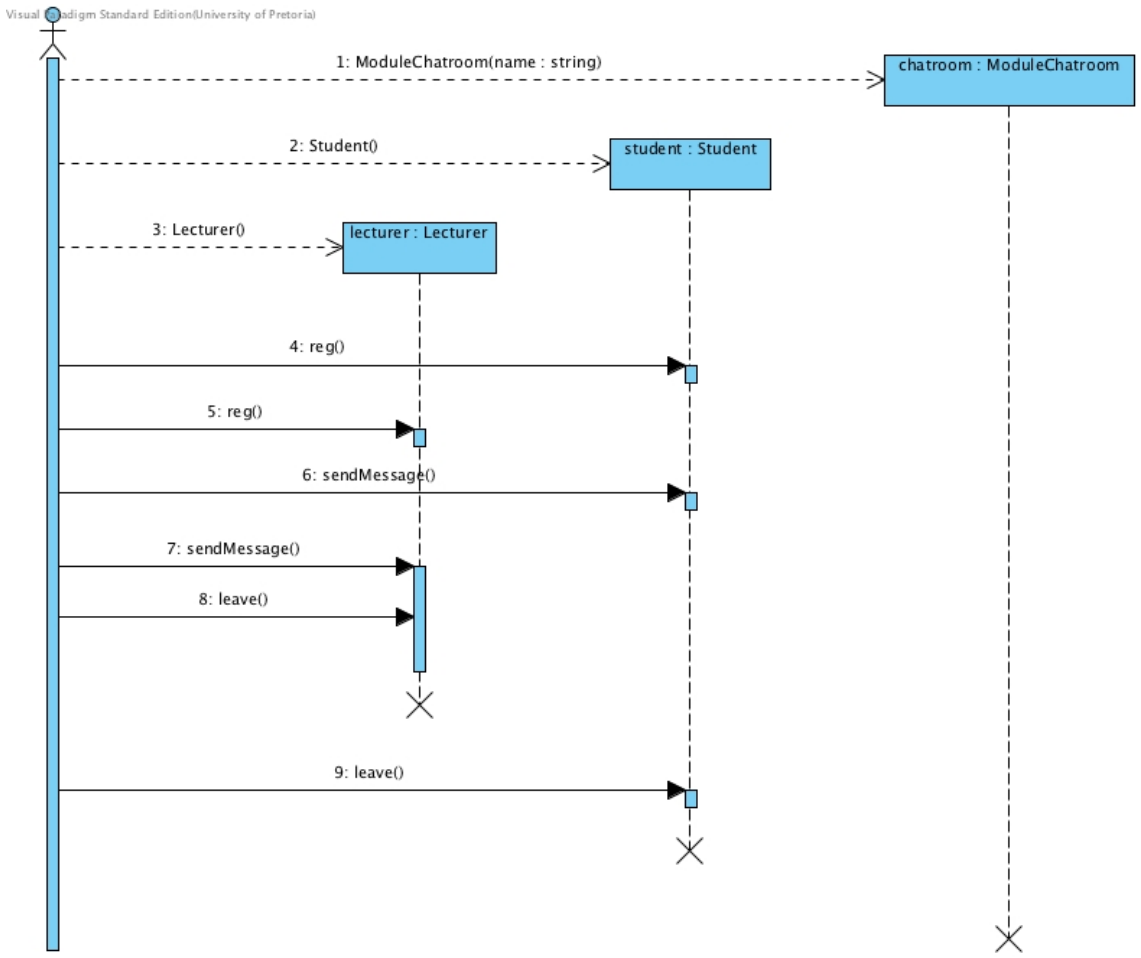
1. For each of the following, indicate if the statement is true or false.
   (a) Responsibilities can be added to an object using the Composite design pattern. (1)
   (b) The Iterator design pattern allows you to randomly walk through a C++ Container library. (1)
   (c) The Observer design pattern allows an object to be watched and an appropriate reaction to be made when the object changes state. (1)
   (d) The Decorator design pattern enables an object to look as if it is changing state. (1)
   (e) The Mediator design pattern centralises control. (1)

   Total for Question 1: 5

2. Consider the partial code given below for the Chatroom example along with a UML Sequence diagram showing the calls for a section of the code base. Answer the questions that follow.

```
class Chatter { // Colleague
  public:
    Chatter();
    virtual void receiveMessage(string) = 0;
    virtual void sendMessage() = 0;
    void reg(Chatroom*);
    void leave();
    virtual ~Chatter();
  protected:
    Chatroom* chatroom;
    int       myId;
};

class Student : public Chatter {
  public:
    Student();
    virtual void receiveMessage(string);
    virtual void sendMessage();
    virtual ~Student();
};

void Chatter::reg(Chatroom* cr){
  chatroom = cr;
  myId = chatroom->registerMe(this);
}

void Student::sendMessage(){
  string toId;
  string msg;
  cout<<"Student "<<myId<<" send message to? ";
  getline(cin,toId,'\n');
  //cin>>toIdstring;
  cout<<"Student "<<myId<<" message? ";
  getline(cin,msg,'\n');
  //cin>>msg;
  ostringstream convert;
  convert << myId;
  chatroom->talkTo(atoi(toId.c_str()),convert.str()+": "+msg);
}
```

```
bool Chatroom::talkTo(int id, string msg){
  // From which ModuleChatroom inherits
  // Must check is the participant is still registered before sending
  //              the message.
  vector<Participant*>::iterator it;
  bool found = false;

  it = participant.begin();
  while ((it != participant.end()) && (!found)) {
    if ((*it)->id == id) {
      found = true;
      (*it)->chatter->receiveMessage(msg);
    } else {
      it++;
    }
  }
  return found;
}
```



(a) Which of the following statements is an accurate depiction of Sequence 1 on the figure? (1)

A. `Chatroom chatroom("COS121");`

B. `Chatroom* chatroom = new Chatroom("COS121");`

C. `Chatroom* chatroom = new ModuleChatroom("COS121");`

D. All of the above.

E. None of the above.

(b) Before Sequence 5 is called and after Sequence 4 has been called, which object is interacted with and which feature is called? (1)

A. Object `lecturer` and feature `reg`.

B. Object `student` and feature `reg`.
C. Object `chatroom` and feature `reg`.
D. Object `lecturer` and feature `registerMe`.
E. Object `student` and feature `registerMe`.
F. Object `chatroom` and feature `registerMe`.

(c) The call to `sendMessage` is followed by a call to which of the following? (1)

A. Object `lecturer` and feature `talkTo`.
B. Object `student` and feature `talkTo`.
C. Object `chatroom` and feature `talkTo`.
D. Object `lecturer` and feature `reg`.
E. Object `student` and feature `reg`.
F. Object `chatroom` and feature `reg`.

(d) A call to `lecturer`'s `receiveMessage` takes place after Sequence 6 following a call to ... . (1)

A. `chatroom`'s `receiveMessage`
B. `chatroom`'s `sendMessage`
C. `chatroom`'s `reg`
D. `chatroom`'s `leave`
E. `chatroom`'s `talkTo`

(e) Which statements are most likely to have been called after Sequence 9? (1)

A. `student->leave();` and `lecturer->leave()`
B. `student->leave();` and `delete student;`
C. `lecturer->leave();` and `delete lecturer;`
D. `chatroom->leave();` and `delete chatroom;`
E. `delete student;` and `delete chatroom;`

Total for Question 2: 5

3. Consider the following two diagrams, the one represents a UML Activity diagram and the other a UML State diagram. Answer the questions which follow.

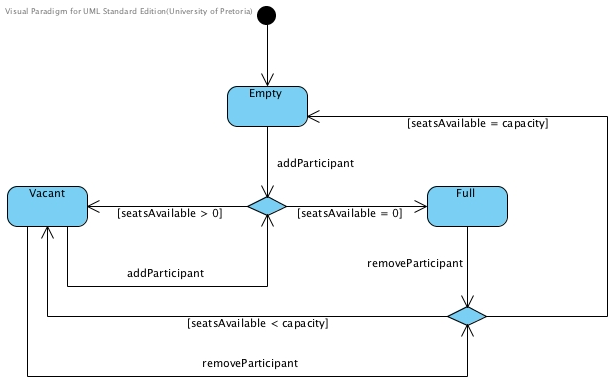


Diagram 1

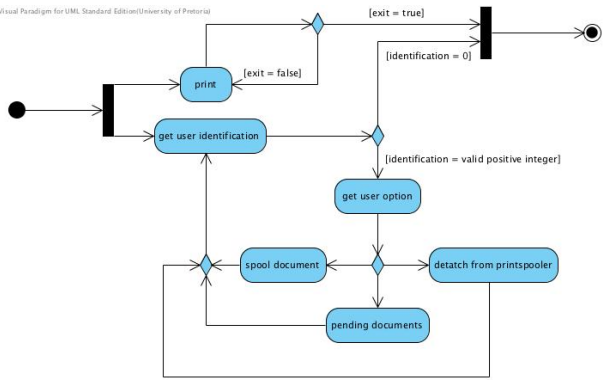


Diagram 2

(a) Which diagram represents a state diagram and which an activity diagram. (2)

(b) Structurally the two diagrams are similar, how do they differ? (2)

Total for Question 3: 4

4. (a) What is Valgrind? (1)

A. A Design Pattern
B. A programming language
C. A collection of memory debugging tools
D. A Documentation Tool
E. An IDE
F. A compiler
G. A debugger

(b) What is GDB? (1)

A. A Design Pattern
B. A programming language
C. A collection of memory debugging tools
D. A Documentation Tool
E. An IDE
F. A compiler
G. A debugger

(c) Which tool is best suited to finding the origin of a segmentation fault? (1)

A. Valgrind
B. GDB
C. Doxygen
D. IDE
E. G++
F. Symbol Table

(d) Which of the following is **not** an IDE that can be used for C++ development? (1)

A. Visual Studio
B. Clion
C. Eclipse
D. Code::Blocks
E. GDB

Total for Question 4: 4

5. Macrosoft will present their new operating system, Doors 9, to the public on the 30th of September. Since their last operating system, Doors 8 and Doors 8.1, was a huge let-down, Macrosoft wants to regain the customers' trust with Doors 9. Today, just two days before the big presentation, a segmentation fault was observed in one of the main components. Macrosoft developers tried, but were unable to find the problem. They decide to get you, a contract developer from Mybuntu, to help them out.

(a) Since you worked at Mybuntu and therefore have more programming experience than the Macrosoft developers, the first thing you try is to find the `segfault` with GDB. The Macrosoft developers indicate that they've already tried that, but were unsuccessful. When running GDB, you get the following message: (1)

```
Reading symbols from d9_menu ...(no debugging symbols found)...done.
```

What must you do to fix this error?

(b) You find the segmentation fault and fix it. Since you suspect that the Macrosoft developers didn't check for memory leaks, you run an analysis on the `d9_menu` executable. Give the command that will be used to check **all** (full) possible memory leaks. (2)

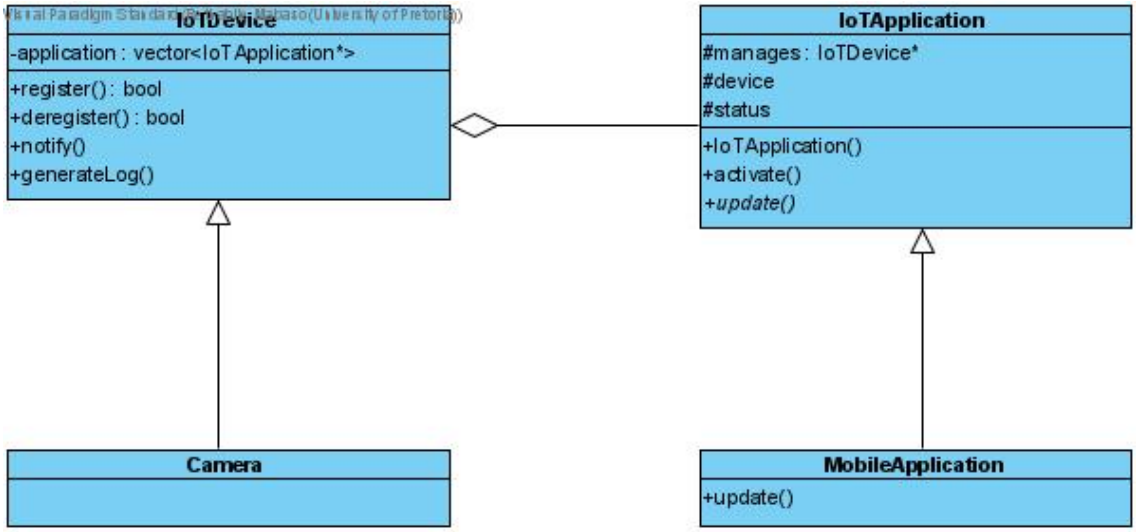
(c) You are still not able to fully resolve all the errors in Doors 9 with the above and suspect that the value of a variable in the function `inititialise` is not being returned to the calling function as required. Which tool would you use to help locate this error? (1)

Total for Question 5: 4

Long questions

6. A startup company founded by UP alumni is developing a range of home security devices that will be integrated into the company's IoT Hub. Currently, the team plans on developing surveillance cameras, motion sensors, and intrusion detectors.

The UML class diagram below shows a model of part of the system proposed by a former COS214 student.



(a) Which design pattern is shown in the model above? Identify all participants of the pattern you suggested. (4)

(b) Provide an implementation of the `notify()` function. (4)

(c) Missing from the diagram is the IoT Hub (class `IoTHub`) through which the iterations between the devices and applications are routed. Any change in the status of a device is reported to the IoT Hub, which in turn notifies active applications and alarm devices. (2)

Which pattern does the inclusion of the IoT Hub result in?

7. Consider the following class definitions and answer the questions that follow.

```
class Graphic {
        public:
                Graphic(int x = 0, int y = 0) : _x(x), _y(y) {}
                virtual void display() = 0;
                virtual void flip(bool horizontal) = 0;
                virtual void rotate(double degrees) = 0;
                virtual ~Graphic(){};
        protected:
                int getX() {return _x;};
                int getY() {return _y;};
                void setXY(int x, int y){_x = x; _y = y};
        private:
                int _x, _y;
};

class Ellipse: public Graphic {
        public:
                Ellipse(int x, int y, unsigned int r)
                : Graphic(x , y), _radius(r) {}
                virtual void display();
                void flip(bool horizontal);
                void rotate(double degrees);
        private:
                unsigned int _radius;
};

class Rectangle: public Graphic {
        public:
                Rectangle (int x, int y, unsigned w, unsigned h)
                : Graphic(x, y), _width(w), _height(h) {}
                virtual void display();
                void flip(bool horizontal);
                void rotate(double degrees);
        private:
```

```
                unsigned int _width;
                unsigned int _height;
};

class CompositeGraphic: public Graphic {
        public:
                CompositeGraphic() : Graphic(), _child() {}
                void addGraphic(Graphic* g);
                void removeGraphic(Graphic* g);
                void display();
                void flip(bool horizontal);
                void rotate(double degrees);
                unsigned int getNumOfLeaves();
        private:
                list<Graphic*> _child;
                void translate(int, int);
};
```

(a) Draw the UML class diagram showing only the classes, relationships and multiplicity. (6)

(b) Identify the design pattern and indicate the participants. (5)

(c) The instance variables `_x` and `_y` are the coordinates of the centre of a given graphical image. When primitive figures are created, it is required that their coordinates are specified. The coordinates of a `CompositeGraphic` is, however, determined by the coordinates of its children. This value is updated when a child is added or removed. (8)

To flip a composite figure horizontally, the x-coordinate of each child needs to be reversed with respect to the x-coordinate of the composite figure. Furthermore, each child also has to be flipped horizontally. Flipping vertically is the same, except that the y-coordinates of the children need to be updated and the children need to be flipped vertically. Assume the `translate`-function when called with parameters -1 and 0 will take care of the repositioning of the children when flipping a composite figure horizontally and when called with parameters 0 and -1 will take care of the repositioning of the children when flipping a composite figure vertically.

Write the code for the `flip` function of the `CompositeGraphics` class.

(d) Assume `Polygon` is to be added to the system as a new kind of primitive figure.

i. State the participant role the class `Polygon` will play in the design pattern that is used. (1)

ii. List all classes in the existing system that will have a relationship with the `Polygon` class and state the relationship. (2)

iii. List all classes in the existing system that will have to be changed to accommodate the addition of `Polygon`. (2)

(e) The following is the code of the main function of a client that was written to demonstrate the system. The `initiate()` function of the `Client` generates a random composite figure. (10)

```
int main() {
        bool HORIZONTAL = true;
        bool VERTICAL = false;
        unsigned int g1Size, g2Size;
        CompositeGraphic* graph1 = initiate();
        CompositeGraphic* graph2 = initiate();
        g1Size = graph1->getNumOfLeaves();
        g2Size = graph2->getNumOfLeaves();
        while (g1Size == g2Size) {
                delete graph1;
                graph1 = initiate();
                g1Size = graph1->getNumOfLeaves();
        }
        if (g1Size < g2Size) {
                graph1->rotate(90);
        } else {
                delete graph1;
                graph1 = initiate();
                g1Size = graph1->getNumOfLeaves();
```

```
                if (g1Size == g2Size){
                        graph2->rotate(180);
                        graph1->flip(HORIZONTAL);
                } else if (g1Size < g2Size) {
                        graph2->flip(VERTICAL);
                } else {
                        graph1->flip(VERTICAL);
                }
        }
        graph1->display();
        graph2->display();
        delete g1;
        delete g2;
        return 0;
}
```

Draw a UML activity diagram of the program flow of this code. Your diagram should have swimlanes for `Client` and `Graphic`.

(f) The `display` function for `Graphic` draws the primitive figures and composites using a 2pt black line.

i. Which design pattern would you use to provide different line thicknesses and colours when displaying the graphic? (1)

ii. Update your answer to Question 7.a, by drawing a new diagram, so that it includes the pattern you identified above. Only provide the class names, relationships and multiplicity in your answer. (4)

Total for Question 7: 39

8. The following illustrates the use of the Observer design pattern in Qt. `TetrisGame` and `ErrorDialog` are both subclasses of `QObject`.

```
TetrisGame game;
ErrorDialog dialog;

QObject::connect(&game, SIGNAL(onError()), &dialog, SLOT(showMessage()));
game.play(); // Plays a game of Tetris
QObject::disconnect(&game, SIGNAL(onError()), &dialog, SLOT(showMessage()));
```

(a) What is the intent of the Observer design pattern in the code above? (1)

(b) Identify the Observer pattern participants for the code above. (4)

(c) Which functions in the above code map onto the following function participants of the Observer design pattern? (4)

i. attach()

ii. detach()

iii. notify()

iv. update()

Total for Question 8: 9

9. The following code snippet illustrates how the Iterator for the C++ Vector container is used.

```
vector<int> values;
values.push_back(985);
values.push_back(420);
values.push_back(396);

vector<string>::iterator it;
for(vector<int>::iterator it = values.begin(); it != values.end(); ++it) {
        cout << *it << endl;
}
```

(a) What is the concrete aggregate? (1)

(b) Identify the following participant functions for the code above. (3)

- createIterator()

- next()
- currentItem()

Total for Question 9: 4

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Engineering, Built Environment and IT
Department of Computer Science

Software Modelling
COS 214

Examination Opportunity 2 (EO2)

28 September 2020

Integrity statement:
The University of Pretoria commits itself to produce academic work of integrity. I affirm that I am aware of and have read the Rules and Policies of the University, more specifically the Disciplinary Procedure and the Tests and Examinations Rules, which prohibit any unethical, dishonest or improper conduct during tests, assignments, examinations and/or any other forms of assessment. I am aware that no student or any other person may assist or attempt to assist another student, or obtain help, or attempt to obtain help from another student or any other person during tests, assessments, assignments, examinations and/or any other forms of assessment.

Personal Details:

Student Number : u18050362
Surname : Graaff
Name(s) : Werner
Email : u18050362@tuks.co.za
Cell : 0716407441

Current Address Where Exam is Taken:

Street : 11 Balboa Place
Suburb : Eldoraigne
City : Pretoria
Province : Gauteng
Country : South Africa

Please record the times below:

Time you downloaded the paper : 17:30
Time you started writing : 17:30
Time you completed the paper : 19:32
Time you will be submitting the paper : 19:40

Please highlight blocks where you had load shedding during the Exam time (if any):

17:30-18:00	18:00-18:30	18:30-19:00	19:00-19:30	19:30-20:00	20:00-20:30	20:30-21:00
21:00-21:30	21:30-22:00	22:00-22:30	22:30-23:00	23:00-23:30	23:30-00:00	

For Examiners' Use Only:

Question	Marks	Score
1	5	
2	5	
3	4	
4	4	
5	4	
6	10	
7	39	
8	9	
9	4	
Total	**80**	

Question 1

(a) False

(b) False

(c) True

(d) True

(e) True

Question 2

(a) C

(b) F

(c)C

(d) E

(e) E

Question 3

(a) Diagram 1: State diagram
Diagram 2: Activity diagram

(b) State diagrams are used to model state dependant behaviour, with the nodes containing different object states, for instance states like “full”, “empty” and “vacant”, and its edges can either have guards and/or actions. Activity diagrams models the flow of actions, with the nodes containing actions such as “print” and “spool document”.

Question 4

(a) C

(b) G

(c) A

(d) E

Question 5

(a) The -g flag must be added when compiling the program.

(b) valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes --verbose

(c) Set a breakpoint at the position where the variable is supposed to be returned and see what the content of the variable is at that stage.

Question 6

(a) Observer design pattern.
Observer: IoTApplication
Concrete observer: MobileApplication
Subject: IoTDevice
ConcreteSubject: Camera

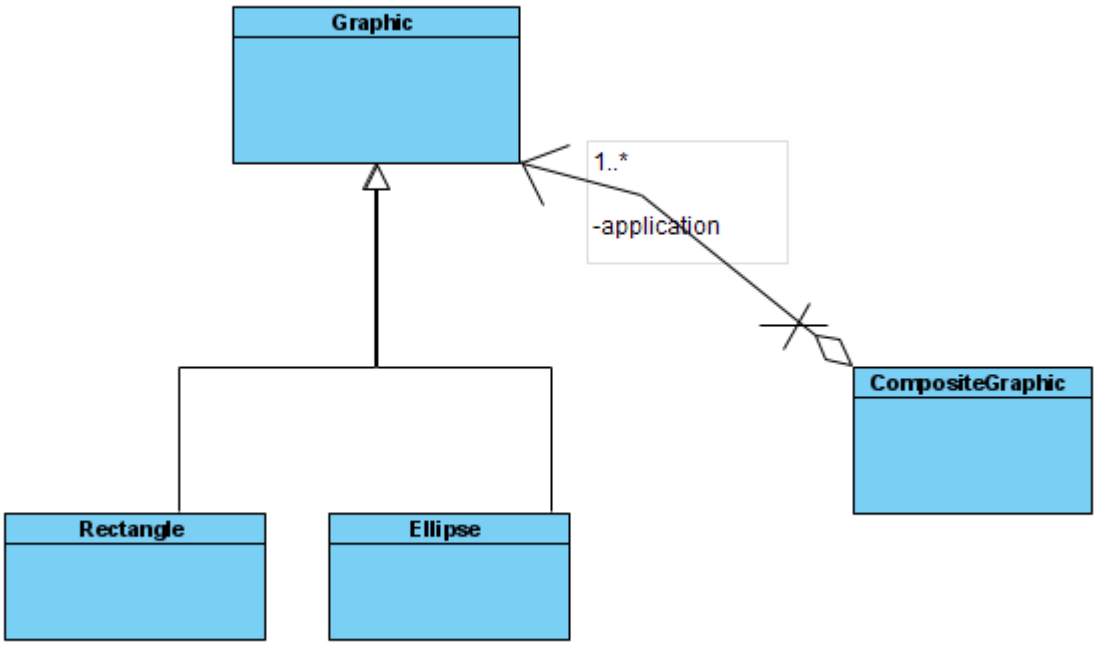
(b)

```
void IoTDevice::notify()
{
        Vector<IoTApplication*>::iterator it = application.begin();
        for(it = application.begin(); it != application.end(); ++it)
        {
                (*it)->update();
        }
}
```

(c) The mediator pattern.

Question 7

(a)



(b) Composite design pattern
Component: Graphic
Leaf: Rectangle, Ellipse
Composite: CompositeGraphic

(c)

```
    void CompositeGraphic::flip(bool horizontal)
{
    if(horizonal == true) //flip horizontal
    {
        list<Graphic*>::iterator it = _child.begin();
        for(it=_child.begin(); it!=_child.end(); ++it)
        {
            (*it)->translate(-1,0);
        }
    }
    else //flip vertical
    {
        list<Graphic*>::iterator it = _child.begin();
        for(it=_child.begin(); it!=_child.end(); ++it)
        {
            (*it)->translate(0,-1);
        }
    }

}
```

(d) (i) Polygon will be another component

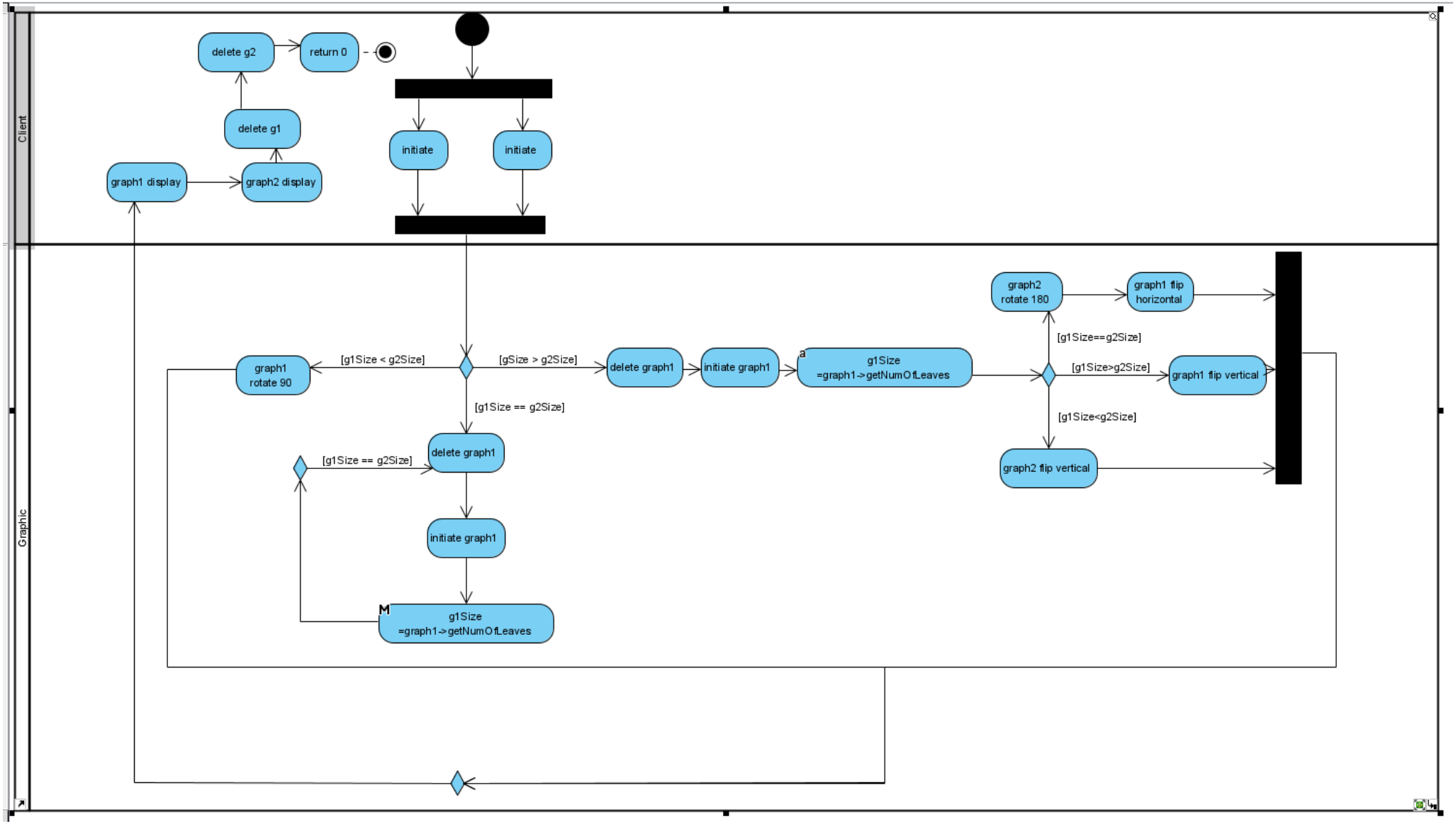
(ii) CompositeGraphic: is-a-has-a polygon

Rectangle: is-a polygon

(iii) Rectangle would now have to inherit from polygon.

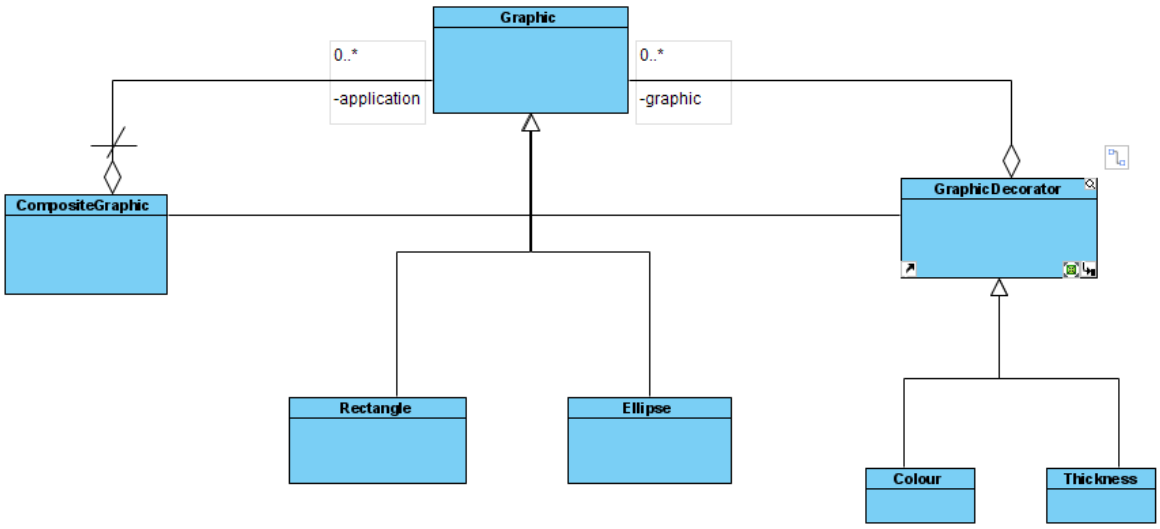
CompositeGraphic would now use a polygon.

(e)


Client
Graphic
delete g2
return 0
delete g1
graph1 display
graph2 display
initiate
initiate
graph1
rotate 90
[g1Size < g2Size]
[gSize > g2Size]
delete graph1
initiate graph1
g1Size
=graph1->getNumOfLeaves
[g1Size == g2Size]
delete graph1
[g1Size == g2Size]
initiate graph1
g1Size
=graph1->getNumOfLeaves
graph2
rotate 180
graph1 flip
horizontal
[g1Size==g2Size]
[g1Size>g2Size]
graph1 flip vertical
[g1Size<g2Size]
graph2 flip vertical

(f) (i) The decorator pattern.

(ii)



Question 8

(a) To receive error messages as the game of Tetris is played.

(b) Observer: QObject
Concrete Observer: TetrisGame, ErrorDialog
Subject: QObject
ConcreteSubject: TetrisGame, ErrorDialog

(c) (i) The connect() function

(ii) The disconnect() function

(iii) The onError() function

(iv) The showMessage() function

Question 9

(a) vector<int> values

(b) createIterator(): vector<string>::iterator it;
next(): operator++
currentItem(): *it