UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
University of Pretoria

Software Modelling
COS121

Examination Opportunity 3 (EO3)

29 October 2015

Name (Initials and Surname): _____

Student number: _____

Examiners

Internal: Dr Linda Marshall, Mr Marius Riekert, Mr Christoph Stallmann and Mr Brian Nyatsine

Instructions

1. Read the question paper carefully and answer all the questions below. / *Lees die vraestel aandagtig deur en beantwoord al die vrae wat volg.*
2. The examination opportunity comprises of **9** questions on **11** pages. / *Die eksamineringsgeleentheid bestaan uit **9** vrae op **11** bladsye.*
3. You have **2** hours to complete the paper. / *Jy het **2** ure om die vraestel te voltooi.*
4. This is a closed book paper. You are therefore not allowed to have any study material with you. / *Hierdie is 'n toeboek vraestel. Jy mag dus geen studiemateriaal by jou hê nie.*
5. Please switch off your cell phone, and keep it off for the duration of the paper. / *Skakel asseblief u selfoon af en hou dit af vir die volle duur van die vraestel.*

Question:	1	2	3	4	5	6	7	8	9	Total
Marks:	7	12	6	8	9	7	5	5	26	85
Score:	7	7	2	5	8	3	3	4	0	

Basics of Design Patterns

1. Consider the following respective lists of patterns, intents and classifications. Match the design pattern with its intent and classification by completing the table below. The intent and classification for the Bridge pattern has been completed in the table to provide a hint as to how to answer the question. (7)

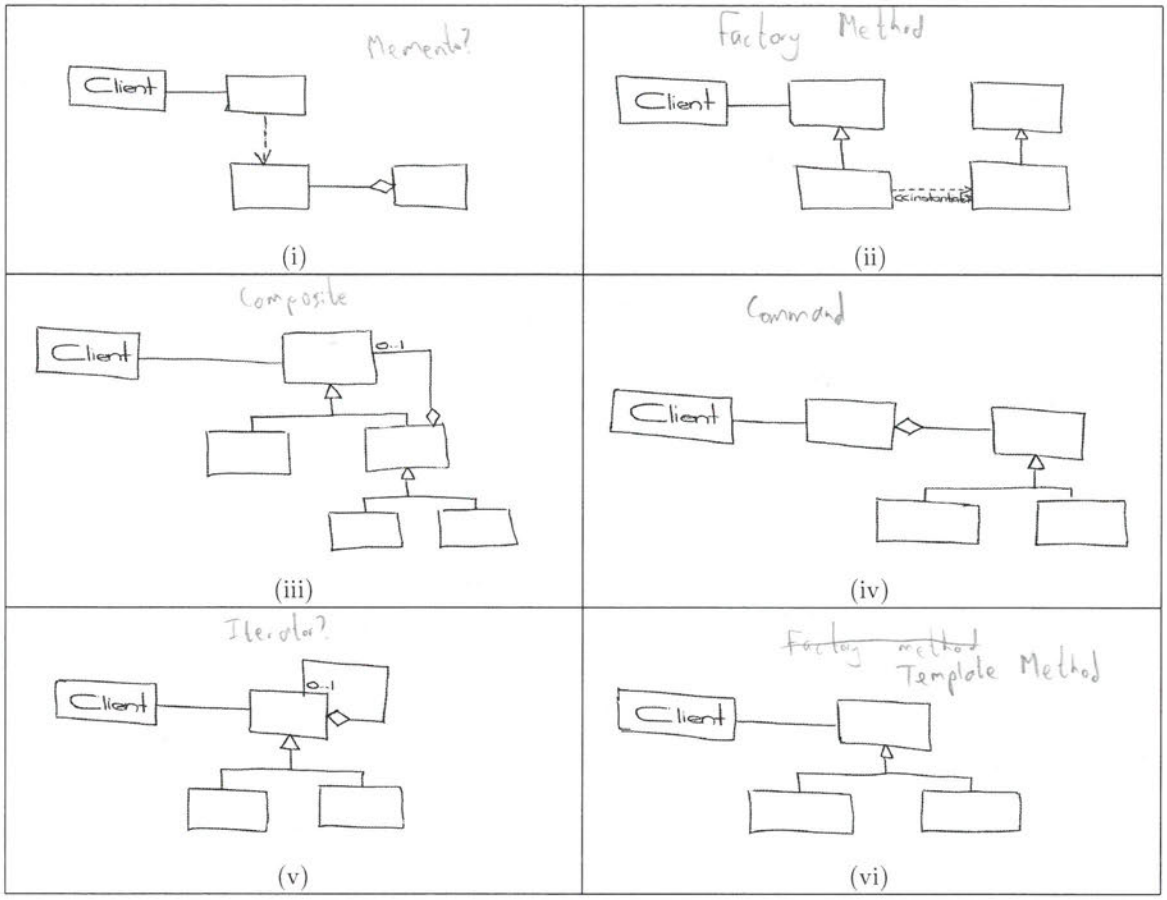
List of Design Patterns	List of Intents	List of Classifications
1. Factory Method	A. Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.	a. Behavioural Patterns
2. Iterator	B. Decouple an abstraction from its implementation so that the two can vary independently.	b. Structural Patterns
3. Composite	C. Define an interface for creating an object, but let subclasses decide which class to instantiate. This pattern lets a class defer instantiation to subclasses.	c. Creational Patterns
4. Bridge	D. Provide a unified interface to a set of interfaces in a subsystem. The pattern defines a higher-level interface that makes the subsystem easier to use.	
5. Façade	E. Compose objects into tree structures to represent whole-part hierarchies. It lets clients treat individual objects and compositions of objects uniformly.	

Design Pattern	Intent	Classification
4.	B.	b.
3	E.	b
5.	D	b.
1	C	c.
2	A	a.

7

2. Identify the design pattern from the skeletal UML class diagrams. Also state whether the pattern is classified as creational, behavioural or structural. (12)

7



(i) Pattern: Memento ✓ Classification: Behavioural ✓	(ii) Pattern: Factory Method ✓ Classification: Creational ✓
(iii) Pattern: Composite ✗ Classification: Structural ✓	(iv) Pattern: Command ✗ Classification: Behavioural ✓
(v) Pattern: Iterator ✗ Classification: Behavioural ✓	(vi) Pattern: Template Method ✗ Classification: Structural ✗

3. The following questions apply to creational design patterns.

(a) What is the intent of creational patterns, that is, why would a programmer use a creational pattern? (1)

To create instances of classes, dependant on what is needed. 0

(b) With regards to the relationship between classes, what is the main difference between the Abstract Factory and the Facory method? (2)

There is inheritance from more than one class (Products from Abstract Products) 0

(c) Is the factory method a specialisation of the template method (Yes/No)? (1)

Yes.

(d) If you want to create one of two concrete products, both inheriting from the same parent class, would it be more appropriate to use the Abstract Factory or the Factory Method to produce the concrete product? (1)

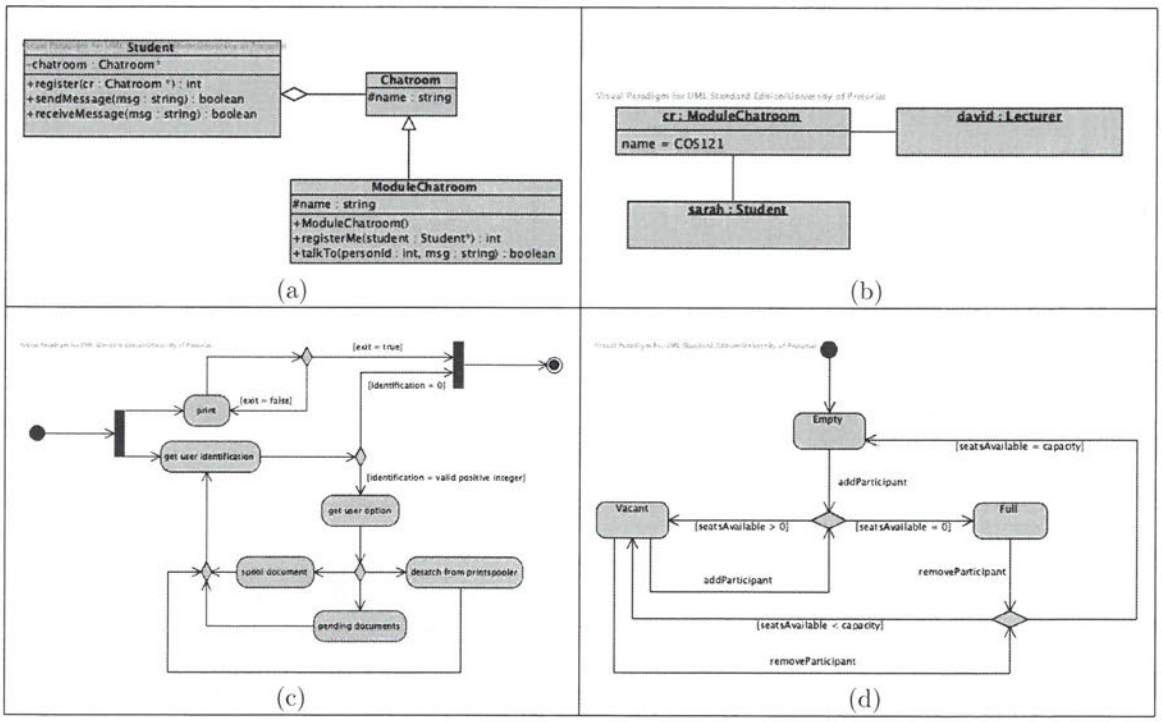
Factory Method

(e) In the Prototype design pattern, you want to maintain a list of prototypes and let a separate class handle the creation, deletion, and maintenance of these prototypes. Name the participant that would be responsible for this task. (1)

Prototype Manager

UML Diagrams

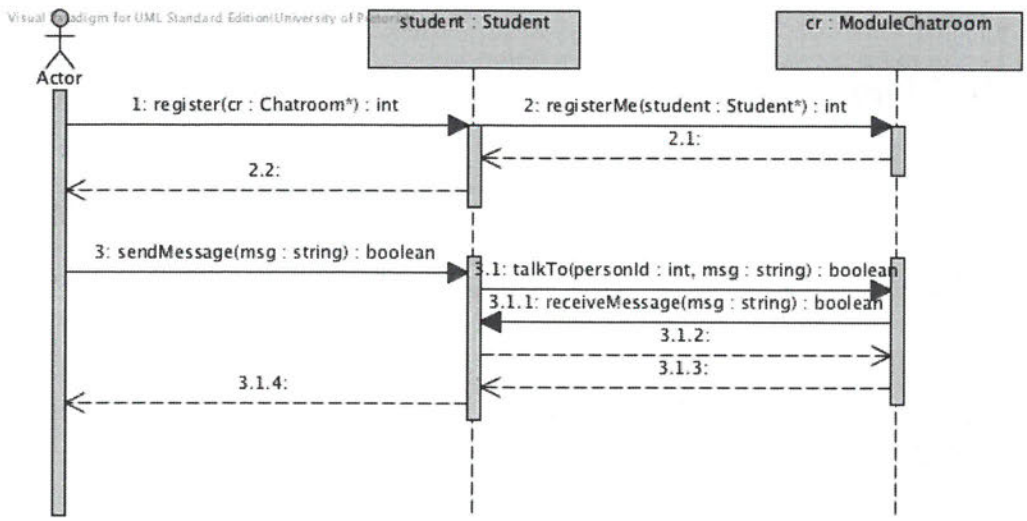
4. Identify each of the following UML diagrams and state whether they are classified as *Structural* or *Behavioural*. You must write your answers in the space provided below. (8)



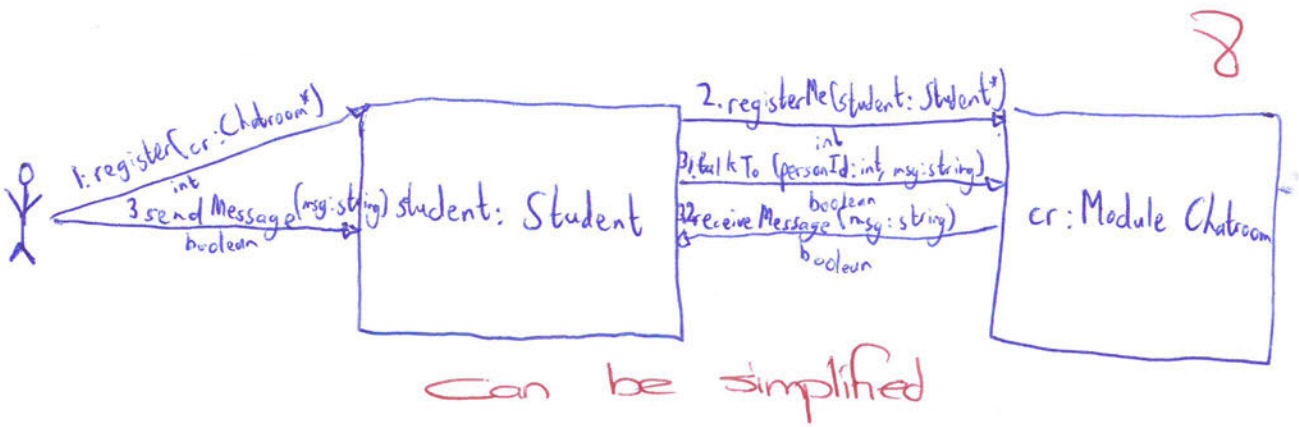
(a) UML diagram: Class diagram ✓ Classification: Structural ✓	(b) UML diagram: State Diagram ✗ Classification: Behavioural ✗
(c) UML diagram: Activity diagram ✓ Classification: Behavioural ✓	(d) UML diagram: ✗ Classification: Behavioural ✗

5

5. Consider the following sequence diagram and answer the questions which follow.



(a) Convert the given sequence diagram into its corresponding communication diagram representation. (8)



8

(b) What is the main difference between a sequence and a communication diagram? (1)

Communication diagram shows how close the communication between classes is.

Combining Design Patterns

6. The following questions apply to the Command design pattern.

(a) What are macro commands? (1)

When the command design patern is implemented with the prototype design pattern, you get Macro Commands.

0

(b) Why is the Decorator not always appropriate to create macro commands? (2)

State or behavioural additions may not be necessary.

6

(c) Which other design pattern can be used to create macro commands? (1)

Prototype. 0

(d) Assume you have a collection of commands and during compile time you are unaware which command will execute which function. Which other design pattern can be used to group all the commands together and pass the request from one command to the other until one of the commands can execute the specified functionality? (1)

Chain of Responsibility 1

(e) Which other design pattern can be used to efficiently traverse through the grouped commands and move from one command to the next? (1)

Iterator.

(f) Which other design pattern can be used to undo the operation of a command that was mistakenly executed? (1)

Memento.

7. A local area network (LAN) broadcasts packets to the entire network, irrespective of who the receiving node is. Hence, if a packet has to be transmitted from `192.168.1.11` to `192.168.1.22`, the packet is sent to the router at `192.168.1.1` and then the router broadcasts the packet onto the network. All nodes on the network may receive the packet, but only node `192.168.1.22` will open and inspect the contents of the packet.

(a) Why is the Observer design pattern not appropriate to broadcast packets on the network? (1)

It is only supposed to observe changes in state of the subject. 0

(b) Which design pattern should rather be used on the router to receive packets from any node on the network and then forward it by broadcasting it to all other nodes on the network? (1)

Mediator. 1

(c) You, the network administrator, suspect that certain nodes on the network are intruders controlled by hackers. Which design pattern should be used to observe certain nodes on the network and notify the router of any suspicious activity, so that the router can automatically disconnect these malicious nodes? (1)

Observer. 1

(d) You want to connect two LANs running on different technologies, one is a Windows network, the other one an Apple network. Which design pattern is most appropriate to be implemented on the router between the two LANs, in order to convert the calls from the Windows LAN into calls that are understood by the Apple LAN? (1)

Bridge. 0

(e) Routers typically have very slow processors and code has to be written efficiently to avoid overloading. Each time a new node is connected, the router creates an object in memory containing a lot of information that is exactly the same for all newly connected nodes. Only after initialisation, the nodes change certain information in this object, such as the IP address. Which design pattern can be used to reduce computational time by avoiding the creation of the large object from scratch each time a new node is connected? (1)

~~Factory Method~~ Prototype. 1

8. You are writing a C++ library to do some linear algebra, such as matrix-vector multiplication and linear regression. The underlying mathematics are very complex and you want third-party users with very little mathematical background to easily understand and use your library.

(a) Which design pattern provides a single point of access to a complex system, allowing users to easily use your library without the need to understand the inner working and mathematics behind the higher-level operations? (1)

Facade. /

(b) Various algorithms exist to multiply matrices, such as the Strassen algorithm and the iterative cache multiplication. Which design pattern is most appropriate for implementing these various multiplication algorithms and allow the addition of other algorithms at a later stage? (1)

Strategy. /

(c) A lot of regression algorithms are very similar in nature where only certain tasks, such as matrix decomposition, is done slightly different among the various algorithms. Which design pattern should be used to retain the generic parts of the algorithms and only delegate the steps that differ to the subclasses? (1)

Template method. /

(d) During nonlinear regression, certain states of a matrix are used during different steps of the regression. Hence, some calculations are done on matrix **A** to temporarily produce a matrix **B**, which is then used in various places of the algorithm. Which design pattern can be used to temporarily save matrix **A** as matrix **B** and simply recall its state when needed, instead of recalculating the entire matrix again? (1)

Memento. 0

(e) You want your library to be able to load matrices from files produced by `Matlab` (another programming language and environment). Which design pattern is most appropriate to read the `Matlab` syntax from file and translate it into a C++ notation understandable by your library? (1)

Interpreter. /

9. Each of the following questions describes a scenario which can be implemented using design patterns. For each question you have to identify the most suited design pattern and participants and answer the questions pertaining to each scenario.

   (a) Assume that a program which will support Boolean algebra is currently under development. The first order of business was to come up with a way to construct and handle Boolean expressions. This stage of development is now complete. The following classes were created, each with a descriptive class name indicating its functionality:

   - `Expression`, which is an abstract class.
   - `BinaryOperator`, which inherits from `Expression`.
   - `Negate`, which inherits from expression.
   - `And` and `Or`, which both inherit from BinaryOperator.
   - `Variable`, which inherits from Expression and represents a single variable.
   - `Constant`, which inherits from Expression.

   The project has now advanced to the second stage where the next order of business is to add functionality so that one may actually use these classes to calculate answers from Boolean expressions in infix notation. Answer the following questions:

i. Which class would be most appropriate to encapsulate a single pointer to an `Expression` object? (1)

Variable

ii. Which participant was not explicitly mentioned for the second stage of development? (1)

Strategy.

iii. Complete the table below by giving the design pattern as well as the role of each of the entities in the pattern: (6)

Not composite

Entity	Role in chosen pattern
The design pattern	Strategy
The `And` class	
The `BinaryOperator` class	
The `Variable` class	
The `Expression` class	
The `Constant` class	

(b) You and a colleague were tasked with developing an application. Each of you received a set of instructions as to what parts of the application you would be responsible for. Unfortunately he was not in the habit of communicating with you properly and he was very unimaginative when it came to naming conventions for his code. He has since resigned and you now have to peruse the code he has written in order to see if there is anything worth salvaging. The following is his code:

```
class A;

class B
{
        friend class A;

        private:
                B(){}

                int coordX;
                int coordY;
                int coordZ;
};

class A
{
        public:

                A(int x_, int y_, int z_, string n);
                void place(int x_, int y_, int z_);
                void output();
                B* checkA();
                void operation(B* b);

        private:
                int x;
```

```
                int y;
                int z;
                string name;
};

A::A(int x_, int y_, int z_, string n)
{
        x = x_;
        y = y_;
        z = z_;
        name = n;
}

void A::place(int x_, int y_, int z_)
{
        x = x_;
        y = y_;
        z = z_;
}

void A::output()
{
        cout<<"My name is "<<name<<" and I am at:
            ["<<x<<','<<y<<','<<z<<']'<<endl;
}

B* A::checkA()
{
        cout<<"returning ["<<x<<','<<y<<','<<z<<']'<<endl;
        B* b = new B();

        b->coordX = x;
        b->coordY = y;
        b->coordZ = z;

        return b;

}

void A::operation(B* b)
{

        this->x = b->coordX;
        this->y = b->coordY;
        this->z = b->coordZ;

        delete b;
}

class C
{
        public:
                C();
                void set(B* b);
                B* get();

        private:
                B* t;
};

C::C()
```

```
{
        t = 0;
}


void C::set(B* b)
{
        if(t) delete t;
        t = b;
}

B* C::get()
{
        return t;
}
```

Answer the following:

i. What functionality was his code supposed to add to the application? (1)

Create an interface for other classes.
Internal state of A to be captured by B.

ii. Write down no more than 6 lines of code (statements) to demonstrate that the code will actually support the functionality from the previous question. (6)

```
B* b = new B();
b->coordX = x;
b->coordY = y;
b->coordZ = z;
return b;
```

iii. Complete the table below by giving the design pattern as well as the role of each of the entities in the pattern: (4)

Entity	Role in chosen pattern
The design pattern	Adapter
The A class	Adapter
The B class	Adaptee
The C class	

Memento
Originator
Memo
Caretaker

(c) You and a colleague were tasked with developing an application. Each of you received a set of instructions as to what parts of the application you would be responsible for. Unfortunately he was not in the habit of communicating with you properly and he was very unimaginative when it came to naming conventions for his code. He has since resigned and you now have to peruse the code he has written in order to see if there is anything worth salvaging. The following is his code:

```
class A
{
        public:
                virtual int disjunct(int,int) = 0;
                virtual int conjunct(int,int) = 0;
                virtual int convert(int) = 0;
};

class B
{
        public:
                B(double f = 0.0, char op = '+');
                char operate(double operand1, double operand2) const;
```

```
                char operator+(const B& operand2) const;
                char operator-(const B& operand2) const;
                char operator=(double other) const;
                void setup(char op);
                char getup() const;

        private:
                char set;
};


class C : public A
{
        public:
                C();
                virtual ~C();
                virtual int conjunct(int l, int r);
                virtual int disjunct(int l, int r);
                virtual int convert(int o);

        protected:
                B* operation;

};

class D
{
        public:
                int operator==(int i);
                int operator=(int i);
                int operator<(int i);
                int operator>(int i);
                int operator+(int i);
                int operator-(int i);
};

class E : public A, private D
{
        public:
                virtual int conjunct(int i, int j);
                virtual int disjunct(int i, int j);
                virtual int convert(int i);
};
```

Unfortunately you were not given any access to the implementation files. Answer the following:

i. What did the client ask for? (1)

Interpretor

ii. Complete the table below by giving the design pattern as well as the role of each of the entities in the pattern: (6)

Entity	Role in chosen pattern
The design pattern	
The A class	
The B class	
The C class	
The D class	
The E class	