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

COS 214 Class Test 3 - L06 to L10

- This test takes place on **28th August 2020**.
- The maximum duration of this test is **40 minutes**.
- This test consists of **5 questions** for a total of **47 marks**.

Question 1 .... (3 marks)

For each of the following statements, identify the pattern that best matches the statement.

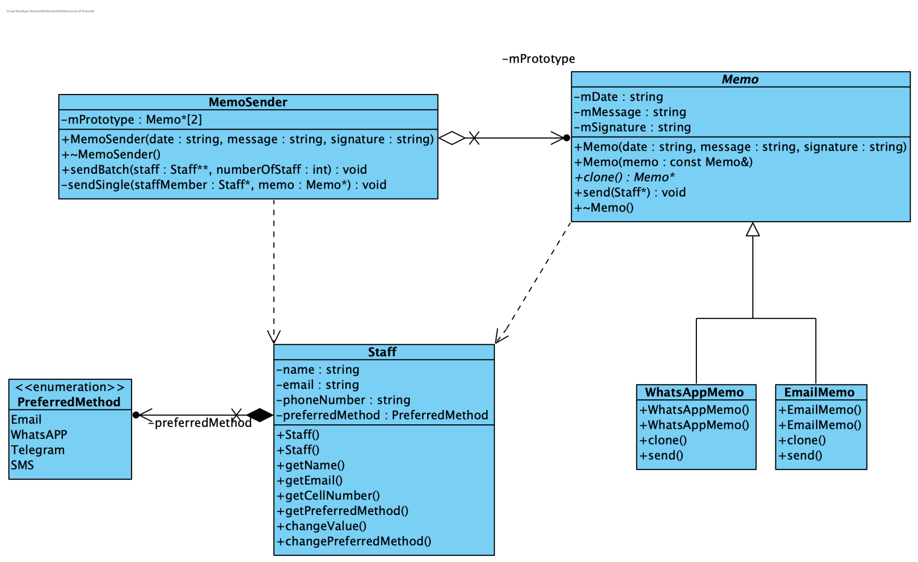
1.1 Provides a hierarchy that encapsulates many possible "platforms", and the construction of a suite of "products". (1)

1.2 Used in situations where it is necessary to dynamically swap out algorithms. (1)

1.3 Creates an object by making a copy of an existing object. (1)

Question 2 .... (11 marks)

Consider the following design for the memo sender example and answer the questions that follow.



2.1 Identify the participants of the design pattern shown in the class diagram. (4)

2.2 Consider the following implementation of the `sendBatch` function.

```
void MemoSender::sendBatch(Staff** staff, int numberOfStaff) {
  cout << "Starting␣to␣send␣a␣batch␣of␣messages␣..." << endl;
  for(int i = 0; i < numberOfStaff; i++) {
    switch (staff[i]->getPreferredMethod()) {
      case Email :
        sendSingle(staff[i], mPrototype[0]->clone());
        break;
      case WhatsAPP :
        sendSingle(staff[i], mPrototype[1]->clone());
        break;
      default : cout << "Error␣—␣preferred␣method␣not␣available" << endl;
    }
  }
  cout << "All␣messages␣were␣sent!" << endl;
}
```

a) Provide an implementation for the `clone` function of the `EmailMemo` class. (3)

b) The `send` function for sending email is defined by: (1)

```
void EmailMemo::send(Staff* staff) {
  Memo::send(staff);
  cout << "Email" << endl;

}
```

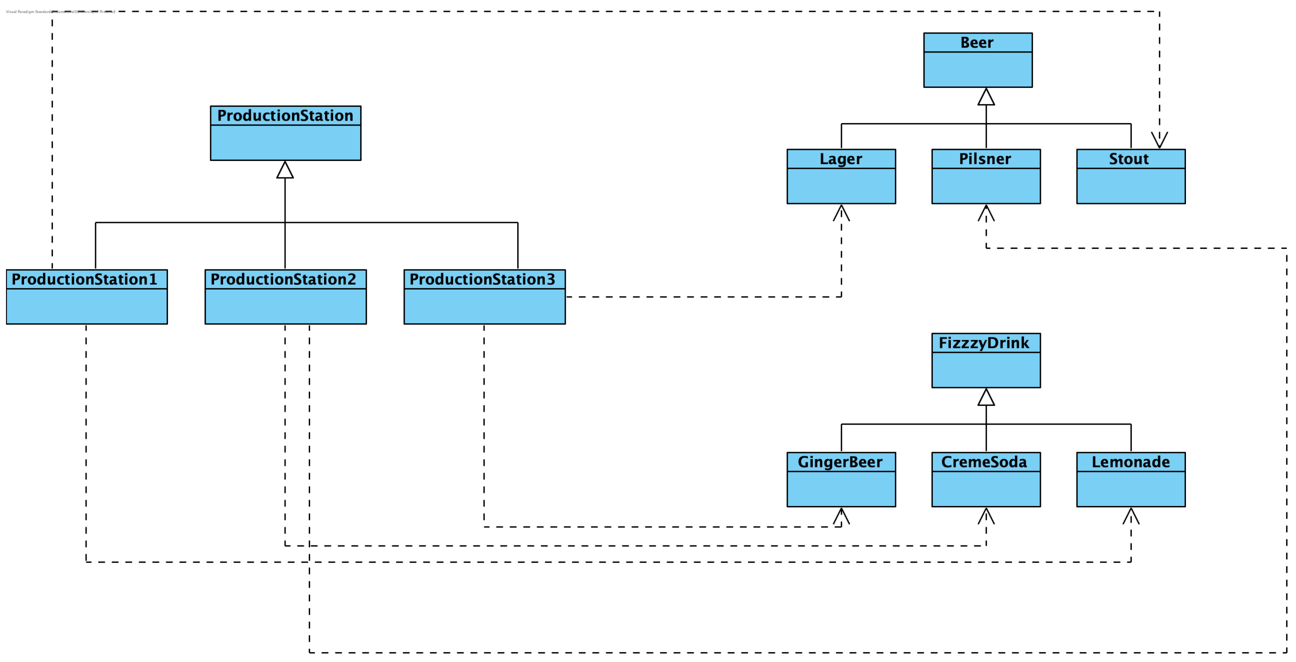
Is this an implementation of the Template Method design pattern?

2.3 Complete the following main program (3)

```
int main() {
  ---[i]--- sender("August␣18,␣2020", "Please␣remember␣the␣online␣meeting␣tomorrow

  ---[ii]---** staffList = new ---[ii]---*[4];

  staffList[0] = new Staff("Paul␣Dool","p.dool@cos214.ac.za",
             "012␣004␣9936",Email);
  staffList[1] = new Staff("Kathy␣Hope","k.hope@cos214.ac.za",
             "022␣804␣9936",WhatsAPP);
  staffList[2] = new Staff("Roger␣Reed","r.reed@cos214.ac.za",
             "084␣999␣5055",WhatsAPP);
  staffList[3] = new Staff("Lebo␣Nkosi","l.nkosi@cos214.ac.za",
             "081␣111␣9033",Telegram);

  ---[iii]--- // Send the message to the staff members

  return 0;
}
```

Question 3 .... (21 marks)

The brewery has been repurposed even further. It now produces and bottles beer, fizzy drinks and sparkling wine. It never produces combinations of beer, fizzy drinks or sparkling wine at the same time. That is, the process in charge of managing what was originally brewing will only produce product from one grouping at a time. The following is a class diagram showing the classes in the system enabling the production of beer and fizzy drinks.



3.1 Identify the participants of the design pattern shown in the class diagram. (8)

3.2 What needs to be added to the given class diagram to enable the production of sparkling wine? (4)

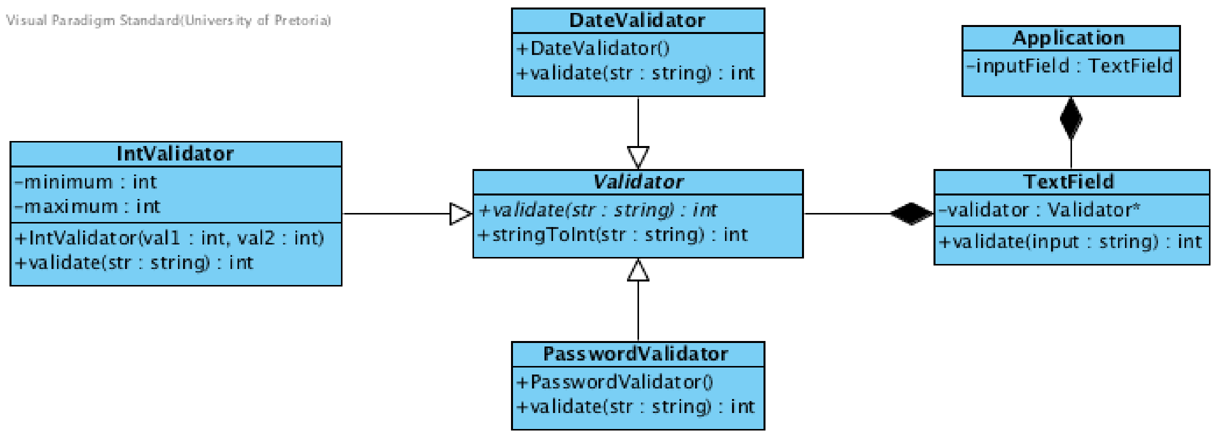
3.3 The pattern shown in the class diagram manages the creation process of the products. Using only this pattern may result in different types of product being produced at the same time. The designers of the system decide that the State design pattern can be used to manage the production of the same type of product at any one time. That is alter the behaviour of the the production process.

a) As it stands, the State design pattern cannot be added. What class needs to included in the design to enable the implementation of the pattern? Include all relationships with this new class and any relationship changes. Draw a UML Class diagram showing the additions and changes. (5)

b) Identify the participants of the State design pattern in the updated design incorporating the State design pattern. (4)

Question 4 .... (6 marks)

The following diagram is the class diagram of the design of a generic input field used by a client program. It applies the Strategy design pattern.


Visual Paradigm Standard(University of Pretoria)
DateValidator
+DateValidator()
+validate(str : string) : int
Application
-inputField : TextField
IntValidator
-minimum : int
-maximum : int
+IntValidator(val1 : int, val2 : int)
+validate(str : string) : int
Validator
+validate(str : string) : int
+stringToInt(str : string) : int
TextField
-validator : Validator*
+validate(input : string) : int
PasswordValidator
+PasswordValidator()
+validate(str : string) : int

4.1 Complete the following list of participants and features as applied to the pattern. (4)

i Context

ii Strategy

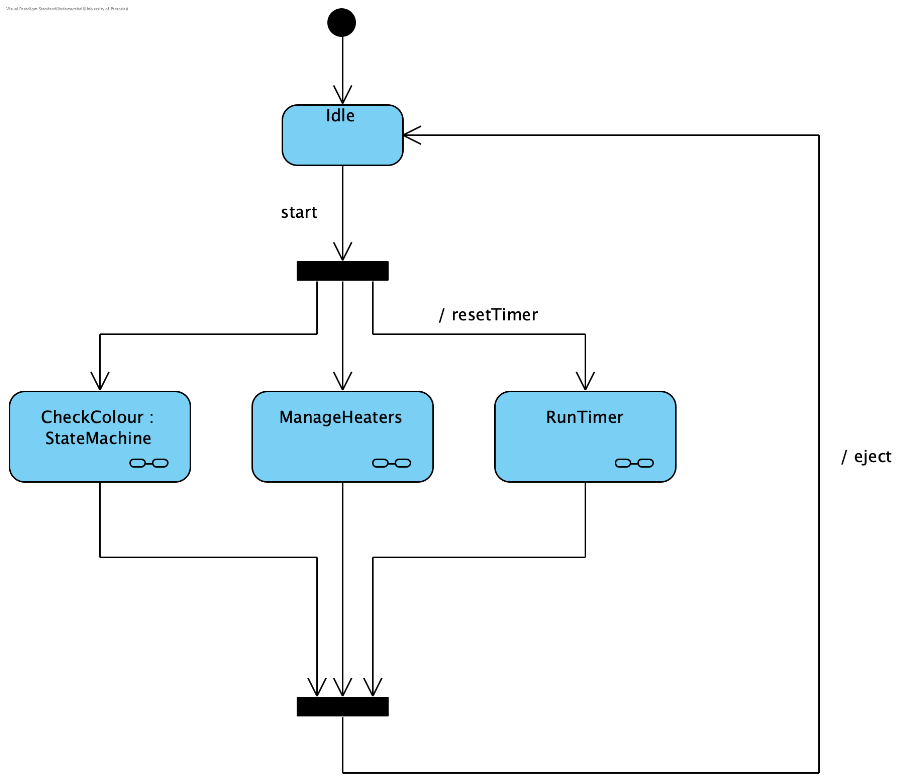
iii Concrete Strategies

iv algorithm

4.2 Write the implementation of the `validate()` method of the `TextField` class as it should appear in `TextField.cpp`. (2)

Question 5 .... (6 marks)

Consider the UML State machine diagram for the toaster example (discussed during Lecture 10 and Tutorial 2, for which the details for the `CheckColour` and `ManageHeaters` states have been designed) given in the figure below.



Design the `RunTimer` substate for the toaster with the following functionality.

- When entering the substate, the timer begins to run and the system enters the `TimerRunning` state.
- Once running, the timer can either expire or be stopped.
  - A timer expires when the set time has elapsed. This results in a `Done` action being issued.
  - The timer is stopped when a `Done` event is triggered from another part of the system.
- Before leaving the substate, the time elapsed is displayed.

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

COS 214 Class Test 3 – L06 to L10

Student Number: u18050362
First Name: Werner
Last Name: Graaff

Cell: 0716407441
Email: u18050362

Question 1

1.1

Abstract factory.

1.2

Strategy method.

1.3

Prototype method.

Question 2

2.1

Prototype: Memo

Concrete prototypes: WhatsAppMemo and EmailMemo

Client: Memosender

2.2 (a)

```
Memo* EmailMemo::clone()
{
	return new EmailMemo(*this);
}
```

(b)

No, because no primitive operations are called inside of send().

2.3

I MemoSender

Ii Staff

Iii sender.sendBatch(staffList, 4);

Question 3

3.1

AbstractFactory: ProductionStation

ConcreteFactory: ProductionStation1, ProductionStation2, ProductionStation3

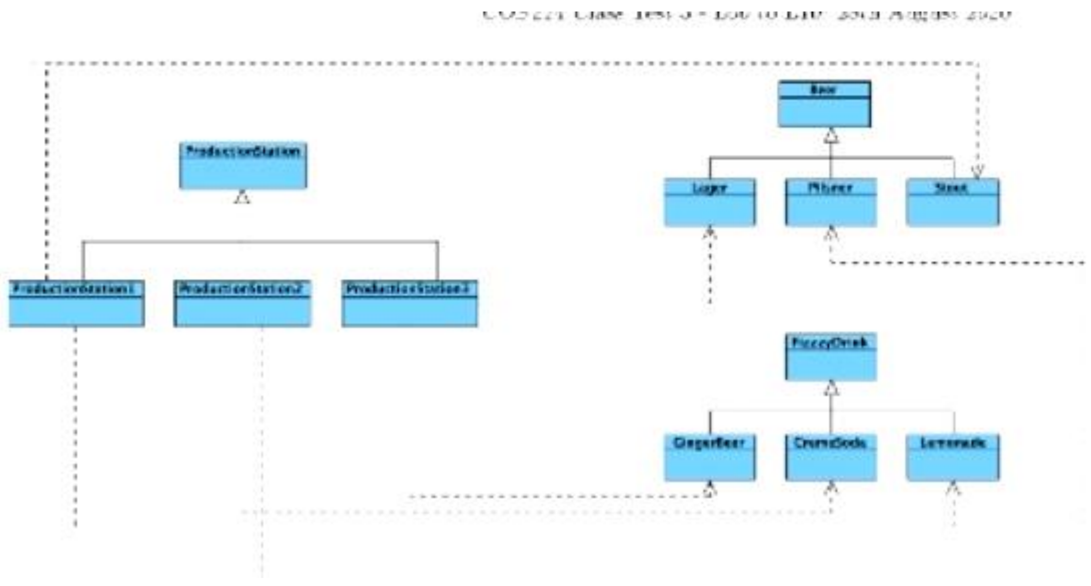
AbstractProduct: Beer, FizzyDrink

ConcreteProduct: Lager, Pilsner, Stout, GingerBeer, CremeSoda, Lemonade

3.2

A new Concrete factory will need to be added below ProductionStation. A new AbstractProduct “Wine” will need to be added with a concreteProduct of “Sparkling”.

3.3 (a)



(b)

State: State

Context: context

Concrete states: PS1state, PS2State, PS3State

Question 4

4.1

i: TextField

ii: Validator

Iii: IntValidator, DateValidator, PasswordValidator

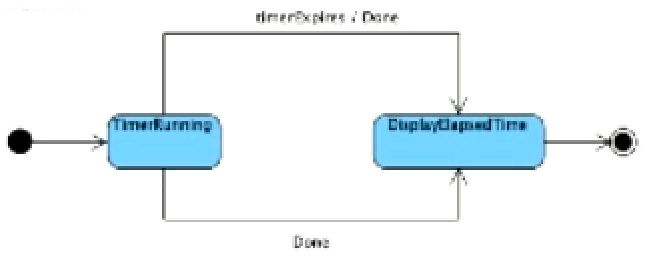
iv: validate(input: string) : int

4.2

```
Int TextField::Validate(string input)
{
        return validator->validate(input)
}
```

Question 5

ution:



- Initial state - 1 mark