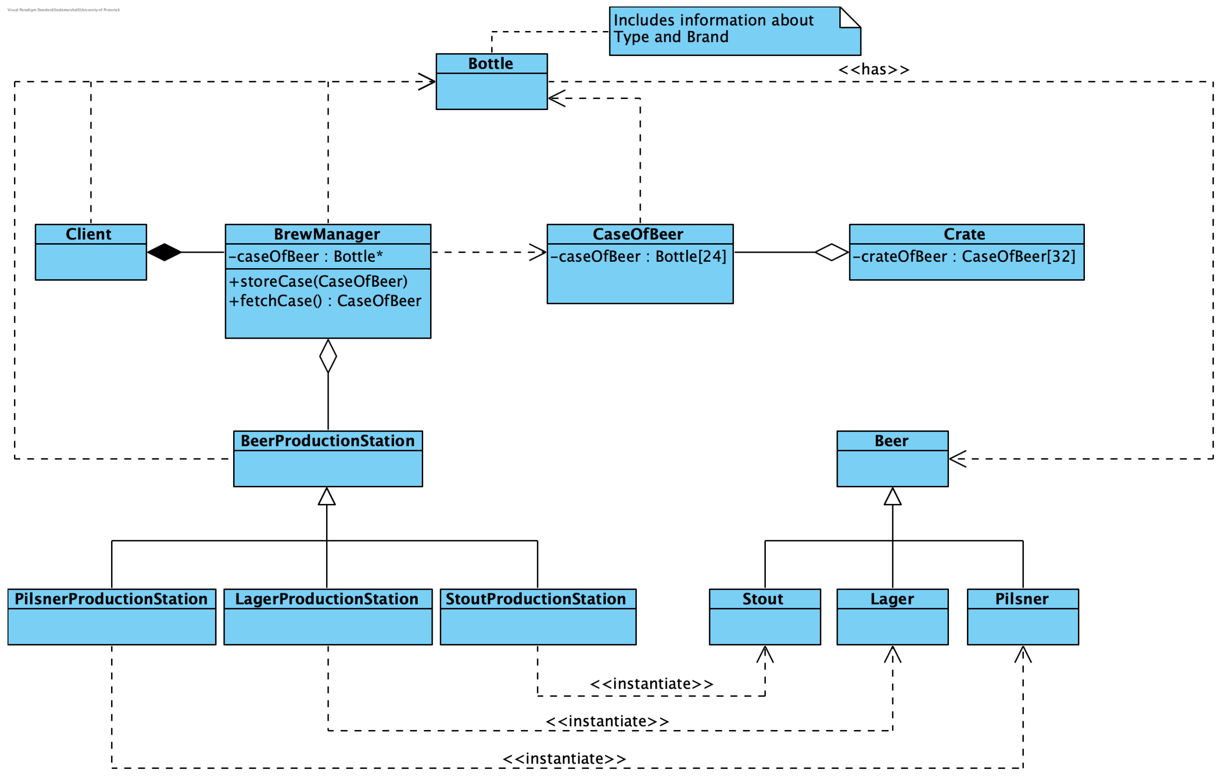
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS 214 Class Test 2 - L02 to L05

- This test takes place on **17th August 2020**.
- The maximum duration of this test is **40 minutes**.
- This test consists of **5 questions** for a total of **45 marks**.

After visiting the local private brewery and sitting through an hour tutorial on how their processes work, you quickly sketch out the following high level design for your design team and programming team to review.



Question 1 ....(5 marks)

1.1 Identify the *client* of the Factory Method pattern. (1)

1.2 Which class represents the Memento participant? (1)

1.3 In which class will the factory method be defined? (1)

1.4 Is the `Bottle` class a participant of a pattern in the class diagram? (1)

1.5 From the information presented in the class diagram, is it possible to say that a Template Method design pattern (other than the one potentially in the Factory Method pattern) is to be implemented? (1)

Question 2 ....(21 marks)

Consider the following definition of the `BrewManager` class and answer the questions that follow.

```
class BrewManager {
```

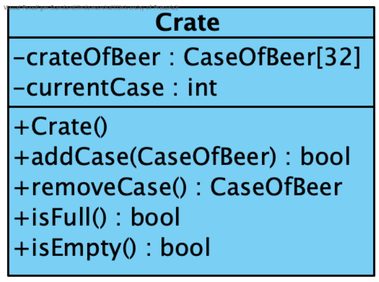
```
public:
  BrewManager();
  void setupBrewingStation(int);
  bool brewBeer(string);
  void cleanBrewingStation();
  vector<Bottle*> shipCrate();

protected:
  bool storeCase(Bottle*  CaseOfBeer);

private:
  Bottle* caseOfBeer;
  BeerProductionStation* beerProductionStation;
  Crate* crate;
};
```

2.1 Which operations represent the `createMemento` and `setMemento` operations of the Memento pattern? You may assume that there are getters and setters defined in the `CaseOfBeer` class. Note, a handle to memento is not held by the Originator in this case. The Originator, creates the memento, populates it and then places it into storage. When the stored mementos are required (either a full or partially filled crate), the Originator will fetch the crate and pass it back to its client. (2)

2.2 The `setupBrewingStation` operation determines which ConcreteCreator is used. `brewBeer` calls the appropriate `produce` operation. Write the code for `brewBeer`. (3)

2.3 Why are the operations of the Memento participant defined as protected? (2)

2.4 The UML class diagram for the `Crate` class is given by: (14)


Crate
-crateOfBeer : CaseOfBeer[32]
-currentCase : int
+Crate()
+addCase(CaseOfBeer) : bool
+removeCase() : CaseOfBeer
+isFull() : bool
+isEmpty() : bool

- The constructor initialises `currentCase` to indicate that the crate is empty on construction.
- If the crate is not full, a case of beer is inserted into the crate and `true` is returned.
- A case of beer is removed from the crate if the crate is not empty and `true` is returned.

Assume that the class definition is given in the file `Crate.h`, provide the implementations that will be placed in the file `Crate.cpp`.

Question 3 .... (3 marks)

The `Bottle` class is defined as follows:

```
class Bottle {
public:
  Bottle();
  void setBrand(string);
  void setType(string);
  string getLabel();
  string getCap();
private:
  string brand;
  string type;
};
```

For each bottle of beer, the type and brand of the beer contained in the bottle needs to be set. Assume an object of Bottle has been created as follows:

```
Bottle* bottle = new Bottle();
bottle->setBrand("Hansa");
bottle->setType("Pilsner");
```

The `getLabel` operation always includes both the brand and the type while the `getCap` operation only displays the brand. Draw the object diagram immediately after these three statements have executed.

Question 4 .... (12 marks)

Consider the following incomplete main program and answer the questions that follow:

```
#include <iostream>
#include <vector>

using namespace std;

int main(){
  BrewManager* brewManager = new BrewManager();
  int choice;
  string brand;
  string type;
  bool crateFull;

  do {
    cout << "Which type of Beer would you like to brew?" << endl;
    cout << "1. Pilsner" << endl;
    cout << "2. Lager" << endl;
    cout << "3. Stout" << endl;
    cout << "4. I'm done brewing" << endl;

    cin >> choice;

    if ((choice < 4) && (choice > 0)) {
      cout << "What is the brand? - ";
      cin >> brand;

      // include code to set up a brewing station
      //   brew the beer and clean the brewing station here

    }

  } while ((choice != 4) && (!crateFull));

  cout << "Either stopped brewing or crate is full" << endl;

  // Check the crate
  vector<Bottle*> crate =  // get the crate of beer, whether full or not.

  for (std::vector<Bottle*>::iterator it = crate.begin() ; it != crate.end(); ++it) {
    cout << "Beers in crate " << *it << endl;
    Bottle* box = *it;
    cout << "Beers in box " << box << endl;
    for (int i = 0; i < 24; i++) {
      // print the label on the bottle.
    }
```

```
        cout << endl;
    }

    delete brewManager;

    return 0;
}
```

4.1 Provide the code to include all necessary header files. (4)

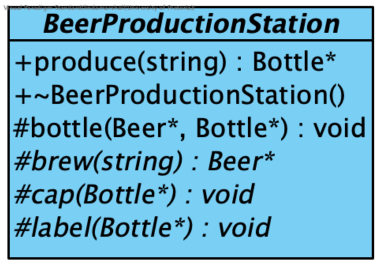
4.2 Write the code to set up a brewing station, brew the beer and clean the brewing station. (6)

4.3 Complete the statement to receive the crate of beer that is either partially full or completely full. (1)

4.4 Write a statement that will display the label on the current bottle from the case. (1)

Question 5 ....(4 marks)

The `BeerProductionClass` is modelled in UML as follows:



5.1 Which participant of the Factory Method pattern does the `BeerProductionStation` represent? (1)

5.2 The class is abstract, how is beer produced? (1)

5.3 Fill in the blanks.
The `produce` operation defined in in the `BeerProductionStation`, is referred to as:

a) the __________ operation in the Factory Method design pattern. (1)

b) the __________ operation in the Template Method design pattern. (1)

Werner Graaff u18050362 COS214 Class test 2

Question 1

1.1. BrewManager
1.2. CaseOfBeer
1.3. BeerProductionStation
1.4. No
1.5. No

Question 2

2.1. Creatememento: bool storeCase(Bottle* CaseOfBeer);

setMemento: vector<Bottle*> shipCrate();

2.2 bool BrewManager::brewBeer(string type)

```
 {
     caseOfBeer = beerProductionStation->produce(brand);
     return storeCase(caseOfBeer);
 }
```

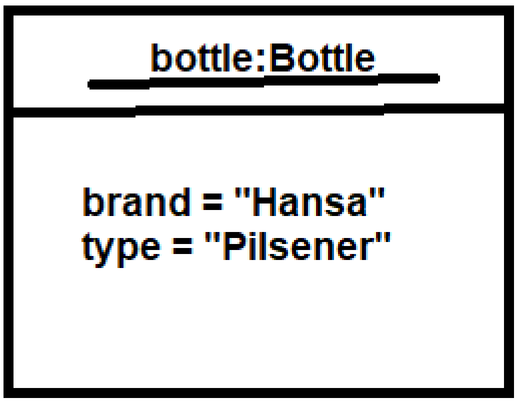
2.3 Wide interface between originator and memento enforcing narrow interface between memento and any other class.

2.4.

```
#include "Crate.h"
Crate::Crate()
{
   this->currentCase = -1;
}
bool Crate::addCase(caseofBeer* caseOB)
{
   if(!isFull())
   {
         currentCase++;
         crateOfBeer[currentCase] = c;
         return true;
   }
   return false;
}
```

```
CaseOfBeer Crate::removeCase()
{
   if(!isEmpty())
   {
           currentCase--;
           return crateOfBeer[currentCase+1];
   }
   return false;
}
bool Crate::isFull()
{
     If(currentCase == 32)
      return true;
}
bool Crate::isEmpty()
{
    If(currentCase == -1)
      return true;
}
```

Question 3



Question 4

4.1. #include “Crate.h”

     #include “CaseOfBeer.h”

4.2.

```
brewManager->setupBrewingStation(choice);

crateFull = brewManager->brewBeer(brand);

Bottle* bottle = bps->produce(choice);

brewManager->cleanBrewingStation();
```

4.3. `brewManager->shipCrate();`

4.4. `cout<<box[i]->getLabel();`

Question 5

5.1. Creator

5.2. By calling the derived classes that implement the virtual functions.

5.3. a) anOperator()

b) Template

COS 214 Class Test 3 - L06 to L10

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

- 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.

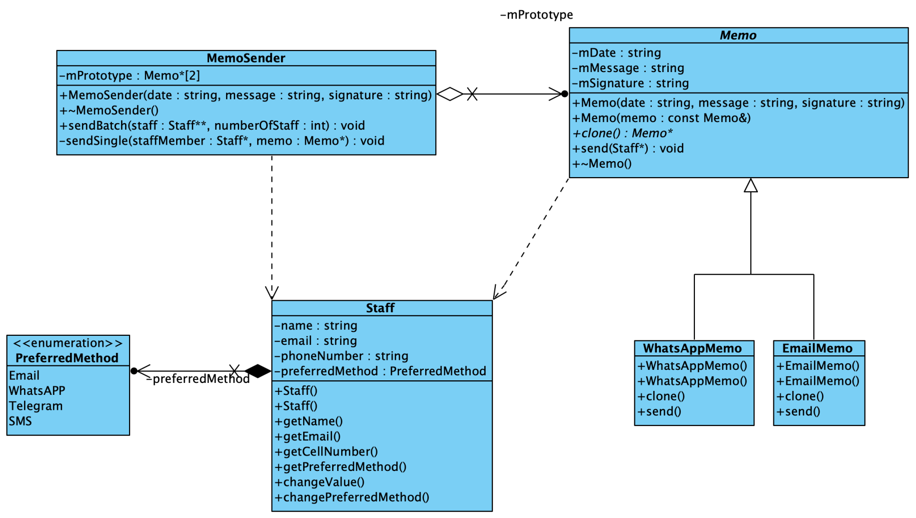
1.1 Provides a hierarchy that encapsulates many possible “platforms”, and the construction of a suite of “products”. (1)

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;

}
```

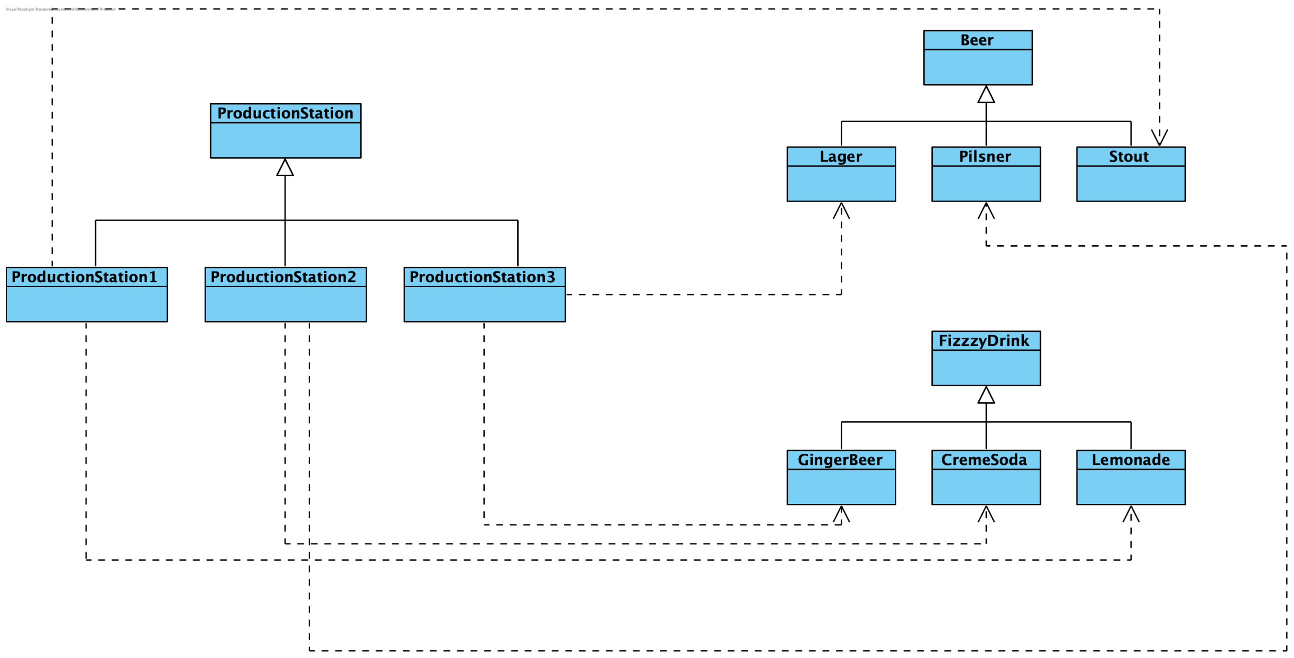
Is this an implementation of the Template Method design pattern?

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)

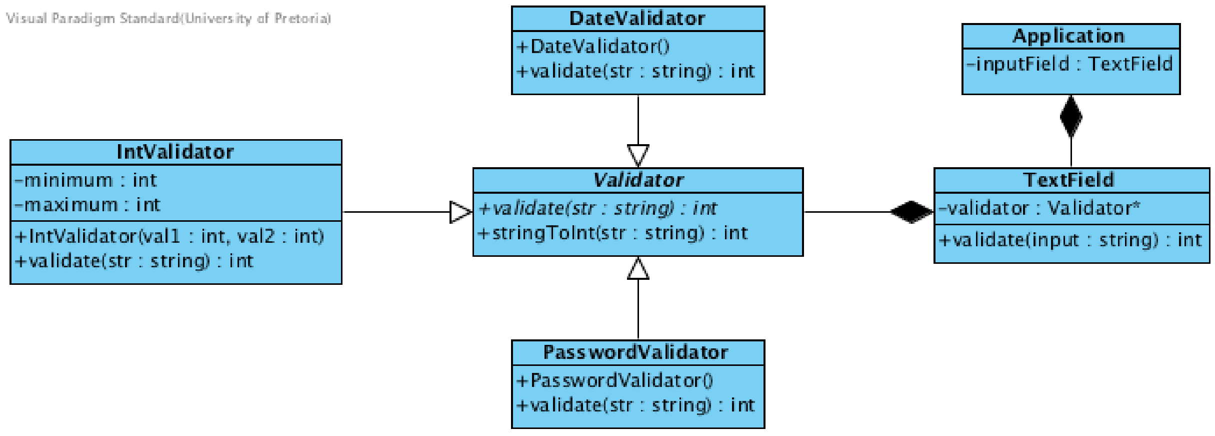
3.3 The pattern shown in the class diagram manages the creation process of the products. Using only this pattern may result in different types of product being produced at the same time. The designers of the system decide that the State design pattern can be used to manage the production of the same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

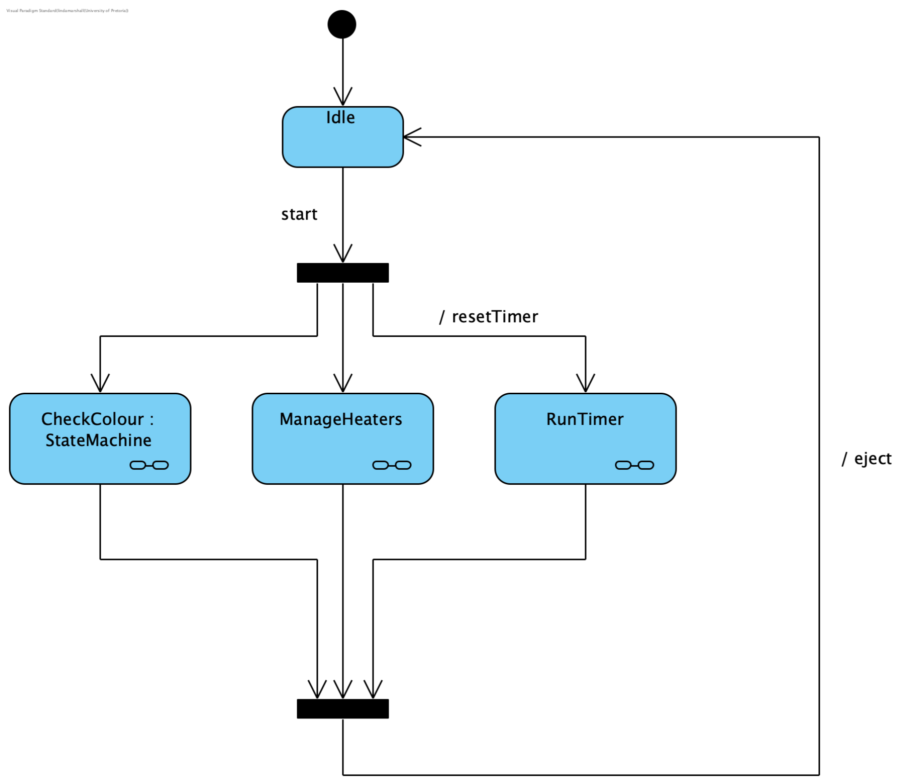
iii Concrete Strategies

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

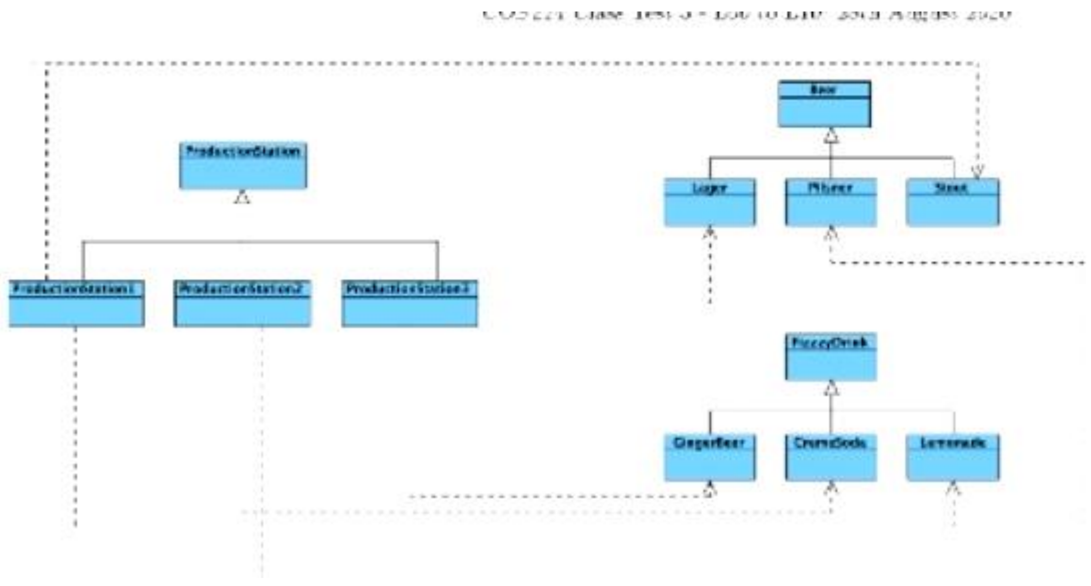
AbstractProduct: Beer, FizzyDrink

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

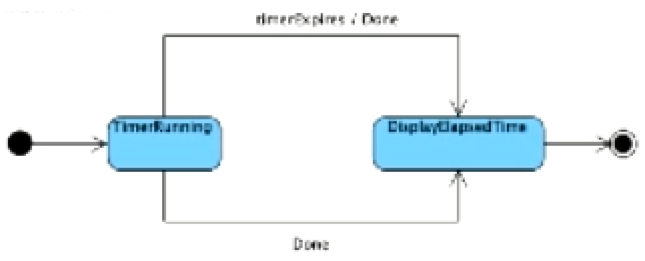
iv: validate(input: string) : int

4.2

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

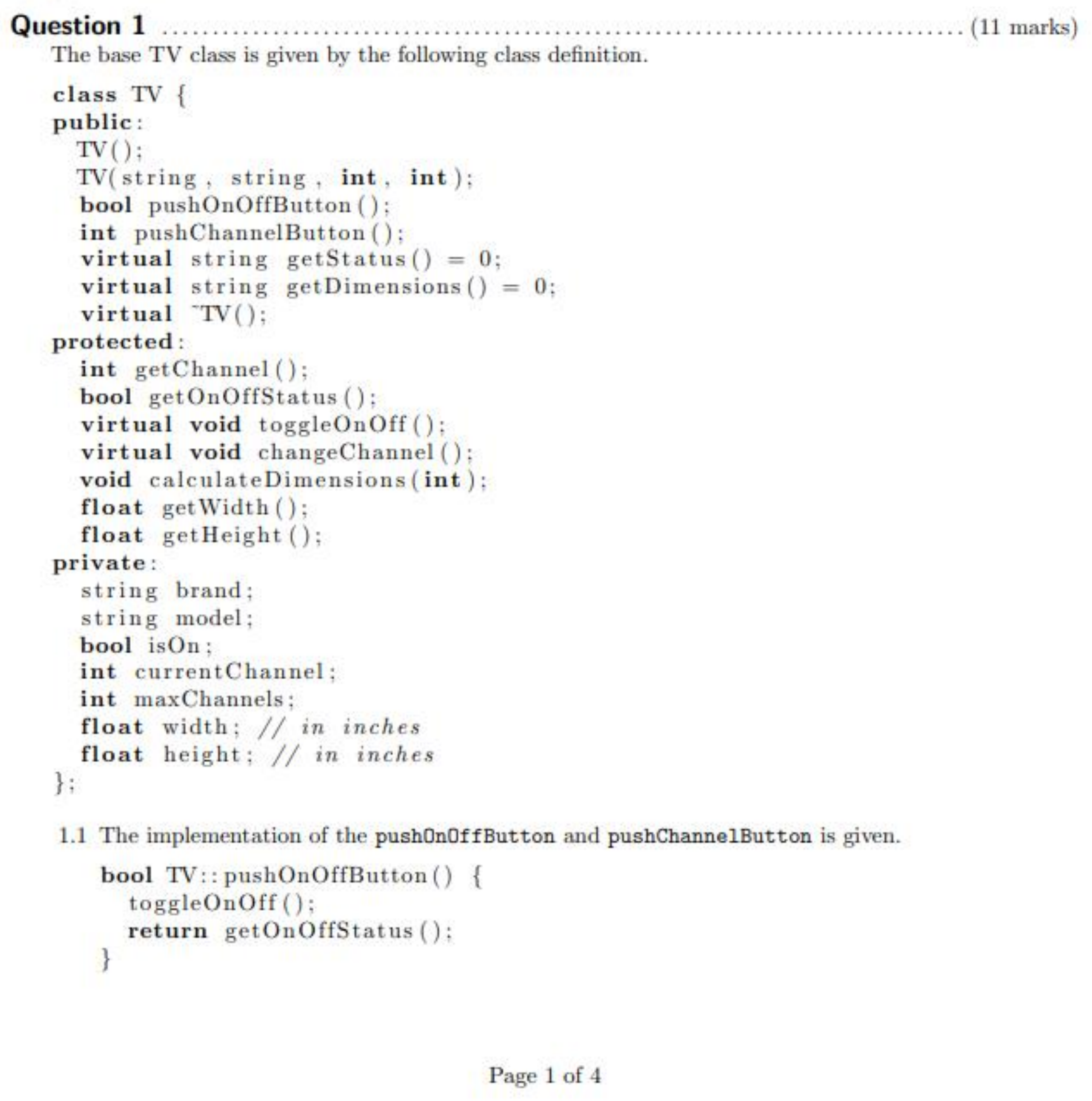
Question 5

ution:



- Initial state - 1 mark

Class test 4 Answers



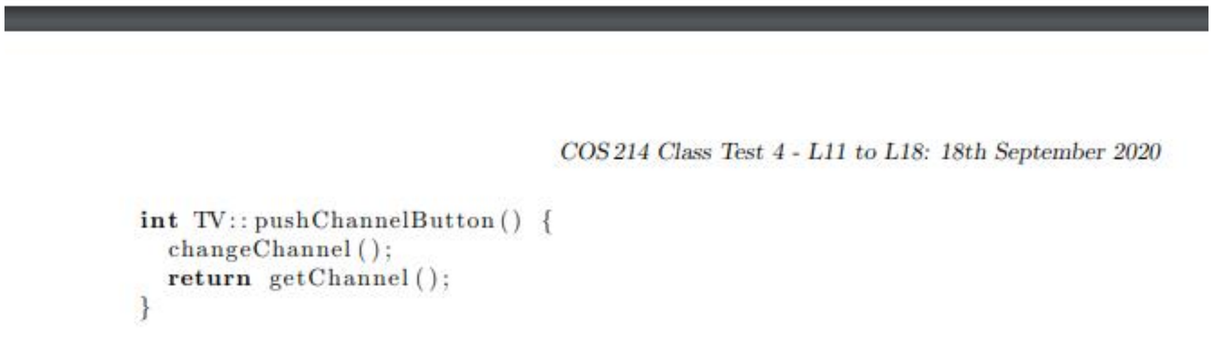
Question 1 .... (11 marks)

The base TV class is given by the following class definition.

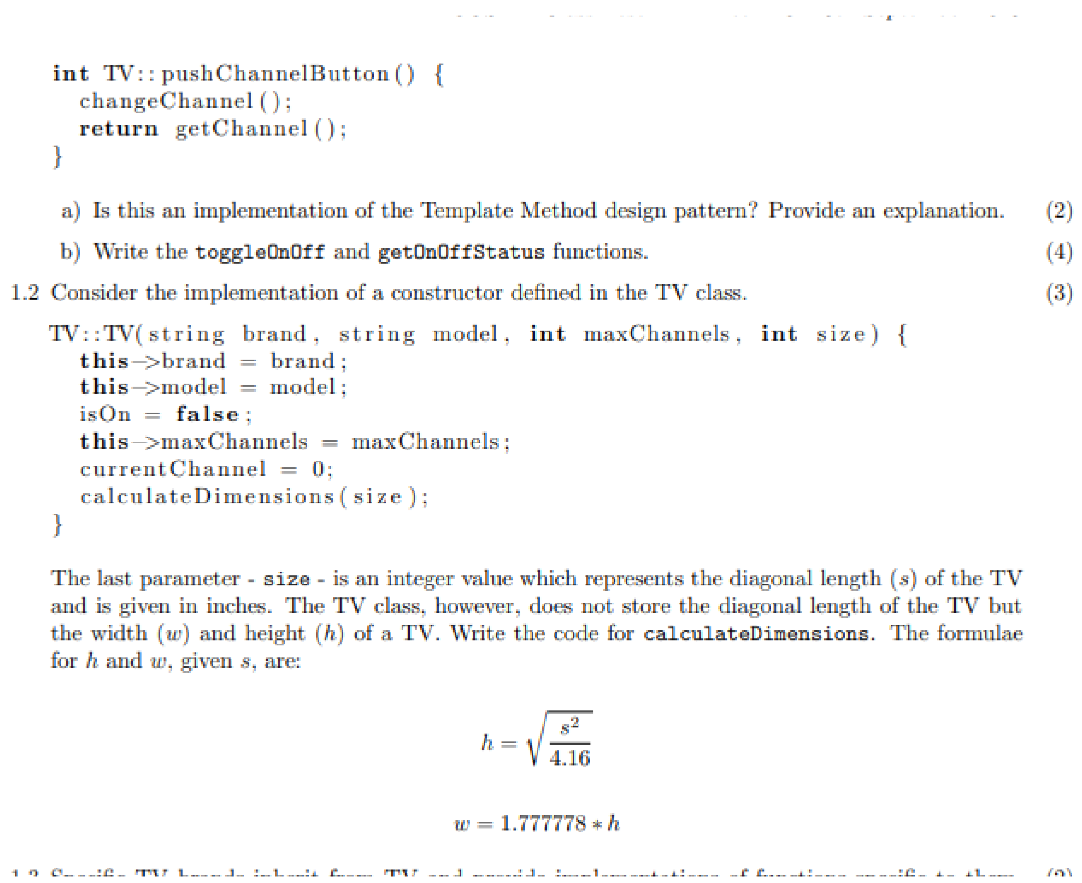
```
class TV {
public:
  TV();
  TV(string, string, int, int);
  bool pushOnOffButton();
  int pushChannelButton();
  virtual string getStatus() = 0;
  virtual string getDimensions() = 0;
  virtual ~TV();
protected:
  int getChannel();
  bool getOnOffStatus();
  virtual void toggleOnOff();
  virtual void changeChannel();
  void calculateDimensions(int);
  float getWidth();
  float getHeight();
private:
  string brand;
  string model;
  bool isOn;
  int currentChannel;
  int maxChannels;
  float width; // in inches
  float height; // in inches
};
```

1.1 The implementation of the `pushOnOffButton` and `pushChannelButton` is given.

```
bool TV::pushOnOffButton() {
  toggleOnOff();
  return getOnOffStatus();
}
```



```
int TV::pushChannelButton() {
  changeChannel();
  return getChannel();
}
```



```
int TV::pushChannelButton() {
  changeChannel();
  return getChannel();
}
```

a) Is this an implementation of the Template Method design pattern? Provide an explanation. (2)

b) Write the `toggleOnOff` and `getOnOffStatus` functions. (4)

1.2 Consider the implementation of a constructor defined in the TV class. (3)

```
TV::TV(string brand, string model, int maxChannels, int size) {
  this->brand = brand;
  this->model = model;
  isOn = false;
  this->maxChannels = maxChannels;
  currentChannel = 0;
  calculateDimensions(size);
}
```

The last parameter - `size` - is an integer value which represents the diagonal length (s) of the TV and is given in inches. The TV class, however, does not store the diagonal length of the TV but the width (w) and height (h) of a TV. Write the code for `calculateDimensions`. The formulae for h and w, given s, are:

$$h = \sqrt{\frac{s^2}{4.16}}$$

$$w = 1.777778 * h$$

1.1.a No, because all the functions that are called are not virtual and are implemented in TV class

1.1 b) void TV::toggleOnOff(){
this->isOn = !isOn; }

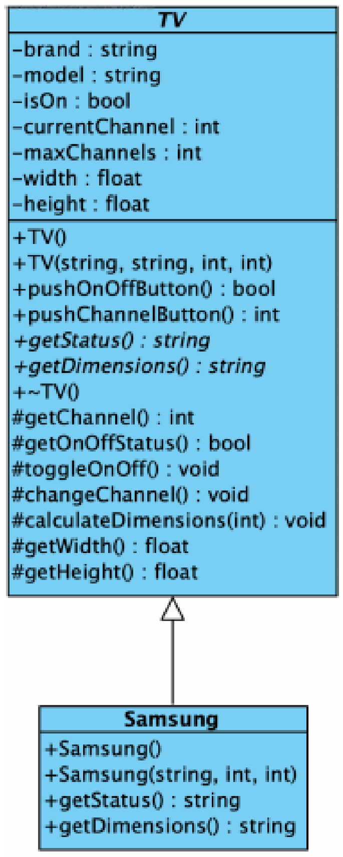
bool TV::getOnOffStatus(){
return this->isOn; }

1.2 void TV::calculateDimensions(int s){

this->height = sqrt(pow(s,2)/4.16));

this->width = 1.777778 * this->height; }

1.3 string getStatus()
string getDimensions()



How would you apply the Composite design pattern to model the wall of TV's? To answer this question, answer the questions that follow.

2.1 For the Composite design pattern, which participants of the pattern do `TV` and `Samsung` represent? (2)

2.2 You may assume that a function exists to calculate the optimal number of TV's required to cover the wall. The function, `calculateWallDimensions`, accepts as a parameter the diagonal length (`size`) of the wall in inches and returns a triple. The first element of the triple is the number of TV's that make up the width of the wall TV's, the second is the number of TV's making up the height. The final element of the triple is the optimal TV size required, defined in inches. These three values must be stored by an object of class `Wall`. (6)

When a wall of TV's is defined, no TV's are assigned to the wall until a member function, `createWall`, is called. This function, returns a boolean if the creation was successful. You may assume, for now, the wall is made of Samsung TV's.

Design a class, call it `Wall`, that when included with the above class diagram, will complete the implementation of the Composite design pattern. Provide only the class definition.

Make use of the `tuple` defined in C++11 for the return type of the `calculateWallDimensions` function. The `tuple` must take 3 `int` values as template parameters, that is `tuple<int,int,int>`.

2.3 Does it make sense to program the Iterator design pattern to iterate through the wall of TV's? Provide reasons for your answer. (2)

2.1 TV: Component

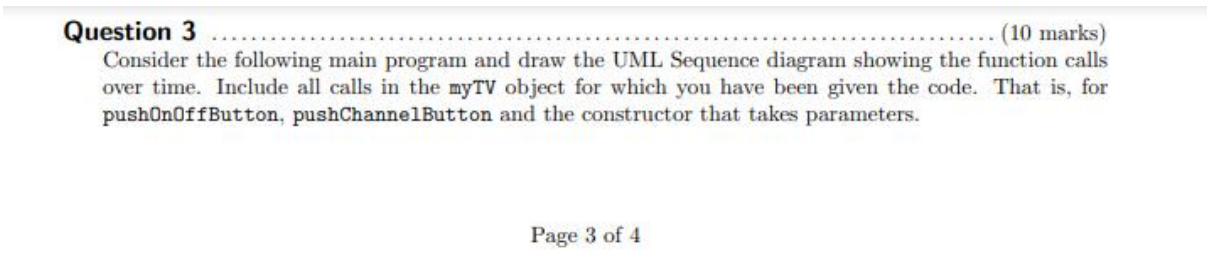
Samsung: Leaf

2.2.

```
class Wall : public TV
```

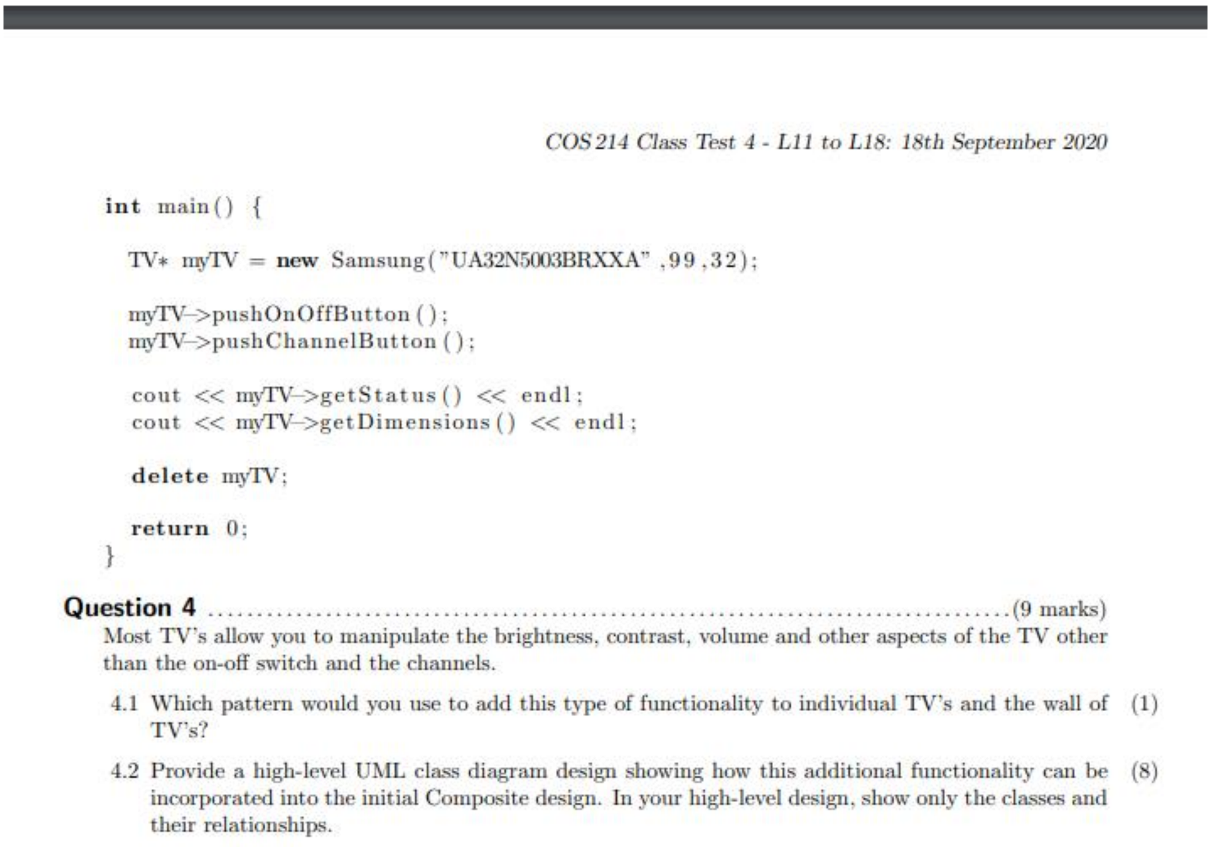
```
{
private:
   int numTVsWidth;
   int numTVsHeight;
   int optimalTVSize;
   TV *tvs[];
public:
   Wall();
   Wall(string huntMethod, string speciality);
getSize();
getHeight();
getWidth();
   ~Wall();
   tuple<int, int, int> calculateWallDimensions(int inches);
   bool createWall(TV tvs[]);
```

2.3. Yes, as it would provide a way to iterate through the TV objects outside of the Wall class, making it easier for the TVs to be traversed in a uniform manner.



Question 3 .... (10 marks)
Consider the following main program and draw the UML Sequence diagram showing the function calls over time. Include all calls in the `myTV` object for which you have been given the code. That is, for `pushOnOffButton`, `pushChannelButton` and the constructor that takes parameters.

Page 3 of 4



COS 214 Class Test 4 - L11 to L18: 18th September 2020

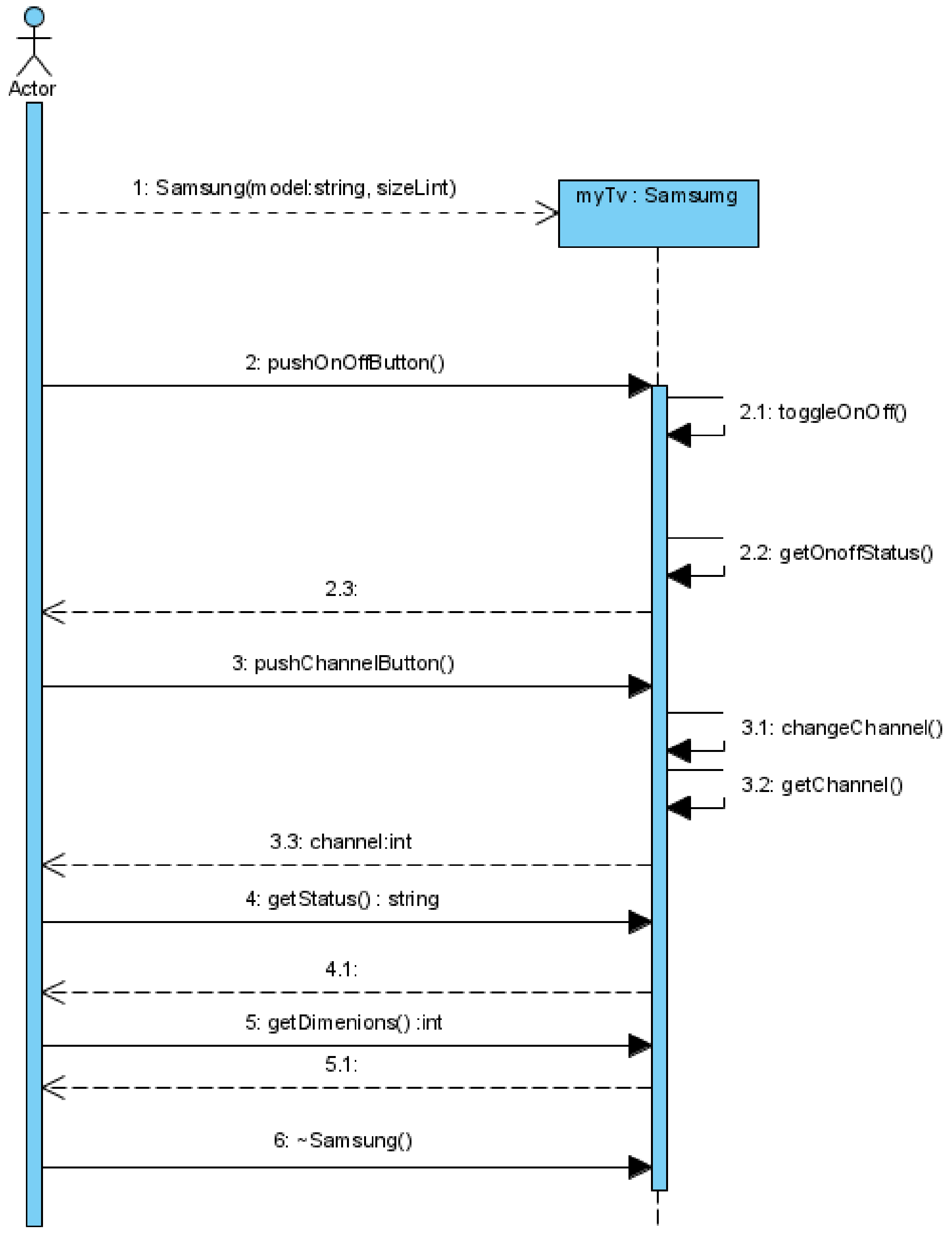
```
int main() {

  TV* myTV = new Samsung("UA32N5003BRXXA",99,32);

  myTV->pushOnOffButton();
  myTV->pushChannelButton();

  cout << myTV->getStatus() << endl;
  cout << myTV->getDimensions() << endl;

  delete myTV;

  return 0;
}
```

Question 4 .... (9 marks)
Most TV's allow you to manipulate the brightness, contrast, volume and other aspects of the TV other than the on-off switch and the channels.

4.1 Which pattern would you use to add this type of functionality to individual TV's and the wall of TV's? (1)

4.2 Provide a high-level UML class diagram design showing how this additional functionality can be incorporated into the initial Composite design. In your high-level design, show only the classes and their relationships. (8)

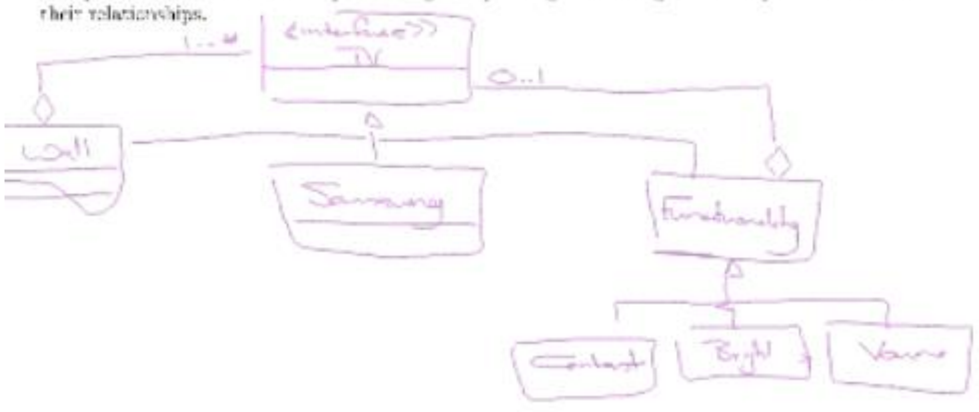
3.



4.1. Decorator

4.2.

incorporated into the initial Composite design. In your high-level design, show only the classes and their relationships.



5.2 Observer

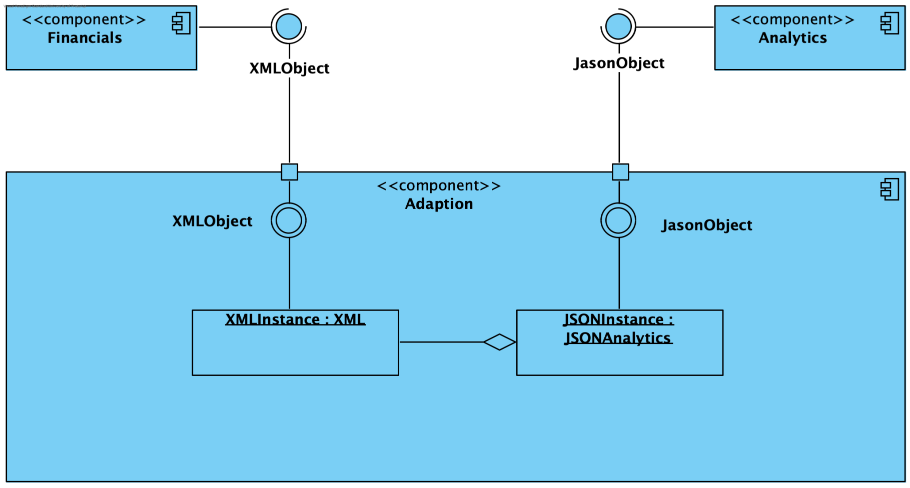
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS 214 Class Test 5 - L19 to L21

- This test takes place on **6th October 2020**.
- The maximum duration of this test is **40 minutes**.
- This test consists of **3 questions** for a total of **36 marks**.

Your client is currently using a financial system that makes use of XML to exchange data between components of their system. They have invested large quantities of money into this system and have a vast amount of data linked to this system which is stored in a data lake. Due to this investment, in the existing system, they are not ready to move over to a new system just yet. They have however purchased a system that will move them to the cutting edge again in terms of data analytics on the data they have in their data lake. This new system uses JSON to exchange data between its components.

Below is an architectural view of the systems and how they are to be adapted so that they can work together.



This is a UML Component diagram. The components (Financials, Analytics and Adaption) comprise of multiple classes. The lollipops (blue circles), represent the provided interface between components and the surrounding circle (socket), the required interface. `XML` *and* `JSONAnalytics` *are classes defined within the Adaption component.*

Effectively, communication can take place between the Finances component and the Analytics component and vice versa. We will only be considering communication from the Finances to the Analytics for the majority of the questions.

Question 1 .... (13 marks)

You decide to apply the Object Adapter design pattern to adapt the XML data format to JSON.

1.1 The classes that exist in the Adaption component are `XMLInterface`, `XML`, `JSONInterface` and `JSONAnalytics`. `XMLInterface` and `JSONInterface` are both abstract classes. (5)

Draw the UML Class diagram for the Object Adapter for this scenario. Show only the classes, whether they are abstract and the relationships between the classes. *Do NOT include any features (functions and attributes)*

1.2 Identify the corresponding classes for the participants of the Adapter design pattern. (3)

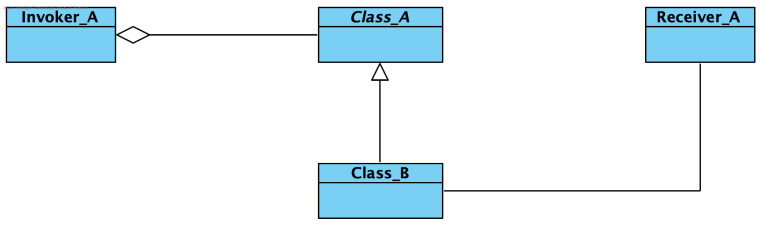
1.3 The `JSONInterface` class is given by: (5)

```
class JSONInterface {
  public:
    virtual string convert(string) = 0;
};
```

Provide a definition for the `JSONAnalytics` class.

Question 2 .... (11 marks)

Rather than sending XML data from the Financials system to the Adaption system as text strings, you decide to send the data '*encapsulated*' as a command. The following is an incomplete representation of the Command design pattern.



2.1 In which components will the classes representing the Invoker and Receiver participants of the command pattern be defined? (2)

2.2 Provide apt names, taking the scenario into account, for the classes labelled `Class_A` and `Class_B`. (2)

2.3 Which class, defined in Question 1, should have a relationship with `Class_B` and what type of relationship should this be? (2)

2.4 If commands were to be sent from the `Analytics` component to the `Adaption` component, how would the Command UML Class diagram given above change? (2)

2.5 Propose a definition for `Class_A`. (3)

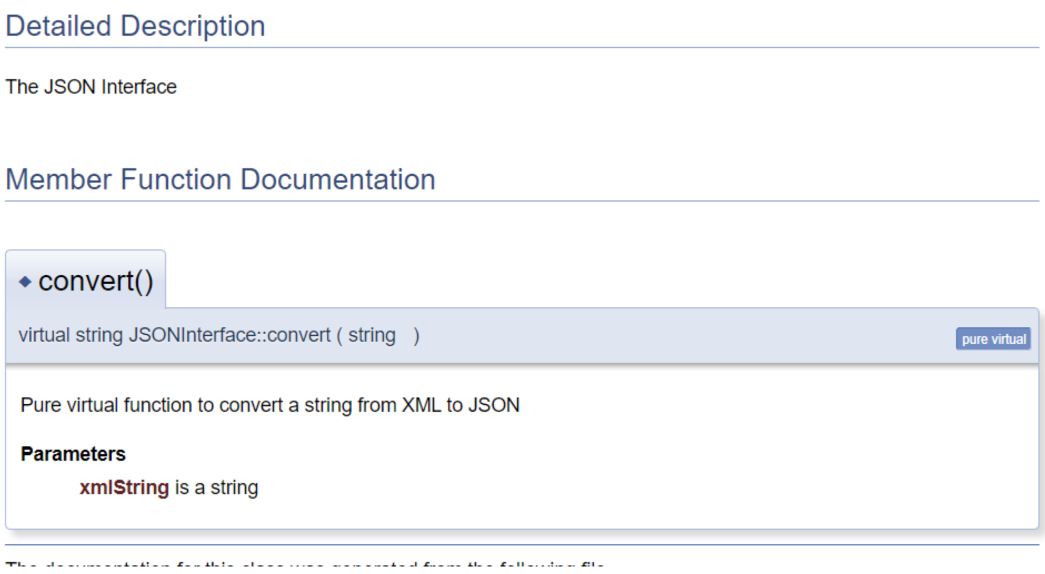
Question 3 .... (12 marks)

3.1 What is the main purpose of each of the following? (2)

a) Doxygen

b) GitHub

3.2 The following section of a web page was generated using Doxygen. The code that was used to generate the page is the JSONInterface class definition.


Detailed Description

The JSON Interface

Member Function Documentation

◆ convert()

virtual string JSONInterface::convert (string)

Pure virtual function to convert a string from XML to JSON

Parameters

xmlString is a string

a) How are multi-line comments specified in Doxygen? (1)

b) Where do you think the text used to populate the *Detailed Description* section is placed in the code file? (1)

c) What text on the web page was produced using a tag and what tag was it that was used to produce the text? (2)

d) The return type of `convert()` is not shown in the documentation. How would you include it? (2)

3.3 What advantages does using a Git repository hold over not using one? (4)

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

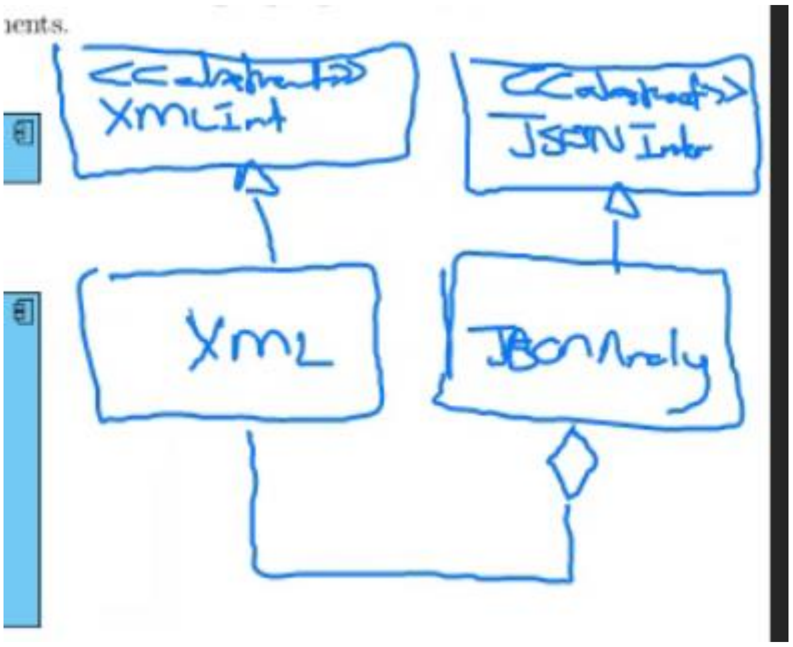
COS 214 Class Test 5 – L19 to L21

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

Cell: 0716407441
Email: u18050362@tuks.co.za

Question 1

1.1



1.2

Adapter: JSONAnalytics

Adaptee: XML

Target: JSONInterface

1.3

```
Class JSONAnalytics : public JSONInterface
{
        Public:
        JSONAnalytics();
        ~JSONAnalytics();
        JSONAnalytics(XML);
        Virtual string convert(string);
Private:
        XML* adapter;
}
```

Question 2

2.1 Invoker: Financials

Receiver: Adaption

2.2 Class_A : Command

Class_B: XMLCommand

2.3 XMLInterface, xml should adhere to command

2.4 Add JSON command inheriting from class A

2.5

```
Virtual execute() = 0;
```

Question 3

3.1 a) To generate documentation for programs in order to make them more understandable.

3.1 b) To enable teams to work together on large projects.

3.2 a) /** Insert comment here */

3.2 b) At the beginning of the JSON interface class definition.

3.2 c) “xmlString” , the tag that was used is @param

3.2 d) By adding a @return tag

3.3 It enables you to have well documented version control, ensuring that everyone on the team are constantly working with the latest versions of the files. It makes it easy to determine which feature should be rolled back should it cause any problems with the rest of the code. GitHub also allows for branching, which makes it easier to work off of previous work if you found a better way to do something. It also allows you to easily track changes that was made by either yourself or other members of the team.

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

COS 214 Class Test 6 - L22 to L26

- This test takes place on **16th October 2020**.
- The maximum duration of this test is **40 minutes**.
- This test consists of **4 questions** for a total of **20 marks**.

Question 1 ....(5 marks)

Consider the following code for the Adaptee, Target and Adapter participants of the Adapter design pattern.

```cpp
class InsaneWrapper {
public:
  InsaneWrapper(int const& _value);
  InsaneWrapper& operator = (int _value) ;
  InsaneWrapper& operator = (InsaneWrapper &_wrapper) ;
  operator int const () const;
  InsaneWrapper& operator ++ ();
  InsaneWrapper& operator ++ (int);
  InsaneWrapper& operator -- ();
  InsaneWrapper& operator -- (int);
  bool operator == (InsaneWrapper const& _wrapper);
private:
  int value;
};

class Wrapper {
public:
  virtual void print(ostream&) = 0;  // print the object
  virtual void increment() = 0;  // increment the wrapped object
  virtual void decrement() = 0; // decrement the wrapped object
  virtual void update(Wrapper*) = 0; // update the wrapped object
  virtual ~Wrapper() {};
};

class SaneWrapper : public Wrapper {
public:
  SaneWrapper();
  SaneWrapper(int);
  virtual void print(ostream&);
  virtual void increment();
  virtual void decrement();
  virtual void update(Wrapper*);
  virtual ~SaneWrapper();
protected:
  InsaneWrapper* object;
};
```

1.1 The Adapter participant in the given code is which of the following classes? (1)

A. `InsaneWrapper`

B. `Wrapper`

C. `SaneWrapper`

D. None of the above.

1.2 `InsaneWrapper` is which participant of the Adapter design pattern? (1)

A. `Adaptee`

B. `Adapter`

C. Client

D. `Target`

1.3 Which of the following class definitions will result in the given code being an implementation of a Class Adapter rather than an Object Adapter. (1)

A. The following line remains the same, **class** SaneWrapper : **public** Wrapper

B. **class** ClassSaneWrapper : **public** Wrapper

C. **class** SaneWrapper : **public** Wrapper, InsaneWrapper

D. **class** SaneWrapper : **private** Wrapper, InsaneWrapper

E. **class** SaneWrapper : **public** Wrapper, **private** InsaneWrapper

F. **class** SaneWrapper : **private** Wrapper, **public** InsaneWrapper

1.4 Which lines need to be removed from the given `SaneWrapper` implementation? (1)

A. Line 23

B. Line 25

C. Line 34

D. Line 35

E. Lines 23 and 25

F. Lines 34 and 35

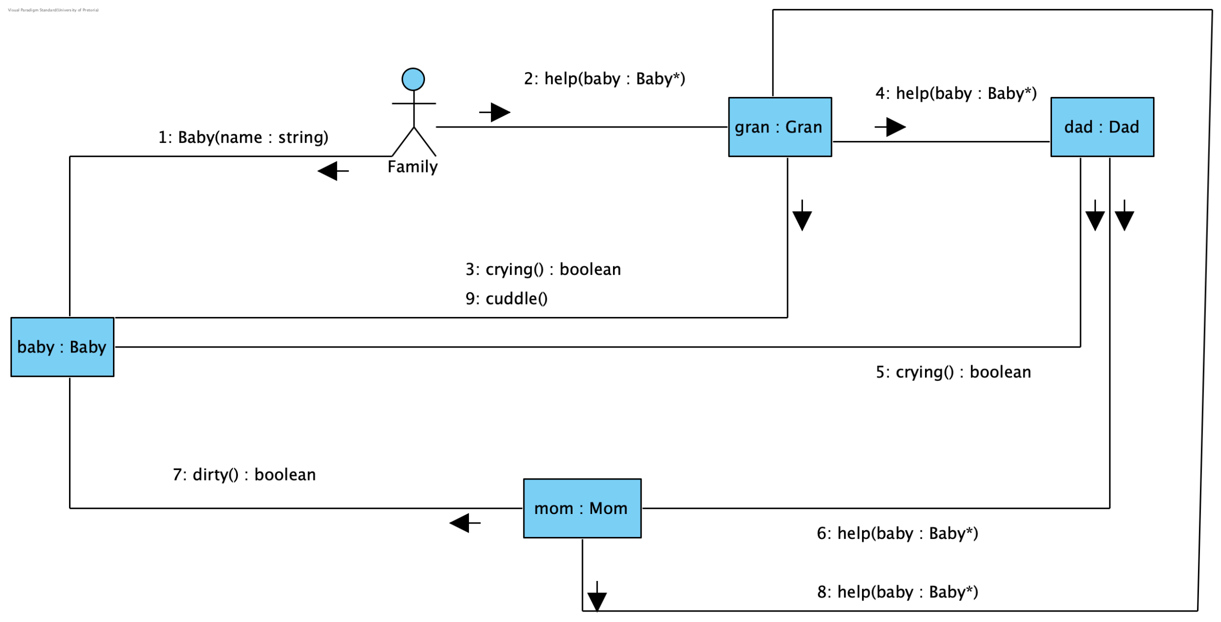
1.5 "An implementation of the destructor is necessary for the Class Adapter implementation of the `SaneWrapper`". (1)

A. True

B. False

Question 2 .... (6 marks)

Consider the following UML Communication diagram and answer the questions that follow.



Assume that all objects, other than `baby`, are instantiations of classes which inherit from the class `Adult`. Class `Adult` implements an association which results in the `Adult` hierarchy implementing *recursive composition*. The design pattern which gave rise to the communication diagram above is the *Chain of responsibility*.

2.1 Which of the following functions will be defined in class `Baby`? (3)

A. `crying`

B. `cuddle`

C. `dirty`

D. `help`

2.2 *Recursive composition* in terms of the `Adults` class means that: (1)

A. an aggregation exists between class `Adult` and class `Gran`

B. at least a composition association exists from class `Adult` to class `Adult`.

C. a chain of composition associations exist from the client to the `Gran` to `Dad` to `Mom` classes.

D. a chain of composition associations exist from the client to the `Baby` to `Gran` to `Dad` to `Mom` classes.

2.3 If you were to explain to a fellow student how the chain works you would in all probability draw a representation of the chain. Which representation of the chain would you use for your explanation? (1)

A. dad : Dad → mom : Mom → gran : Gran

B. dad : Dad → gran : Gran → mom : Mom

C. gran : Gran → dad : Dad → mom : Mom

D. gran : Gran → mom : Mom → dad : Dad

E. mom : Mom → gran : Gran → dad : Dad

F. mom : Mom → dad : Dad → gran : Gran

2.4 Which function represents the *handleRequest* function of the *Chain of responsibility* design pattern? (1)

A. `crying`

B. `cuddle`

C. `dirty`

D. `help`

Question 3 ....(4 marks)

You have been given the following list of classes and told that they are all participants of the Builder design pattern.

- `HousePlan` - an abstract class
- `House` - a specialisation of class `HousePlan`
- `Home` - another abstract class
- `Igloo` and `Tipi` - specialisations of `Home`, each hold a handle to a `House` object.
- `CivilEngineer` - has an aggregation association with Home.

For each of the following participants of the Builder design pattern, identify the corresponding classes.

3.1 Builder (1)

A. `HousePlan`

B. `Home`

C. `Igloo`

D. `CivilEngineer`

3.2 ConcreteBuilder (1)

A. `House`

B. `Home`

C. `Tipi`

D. `CivilEngineer`

3.3 Director (1)

A. `House`

B. `Home`

C. `Igloo`

D. `CivilEngineer`

3.4 Product (1)

A. `House`

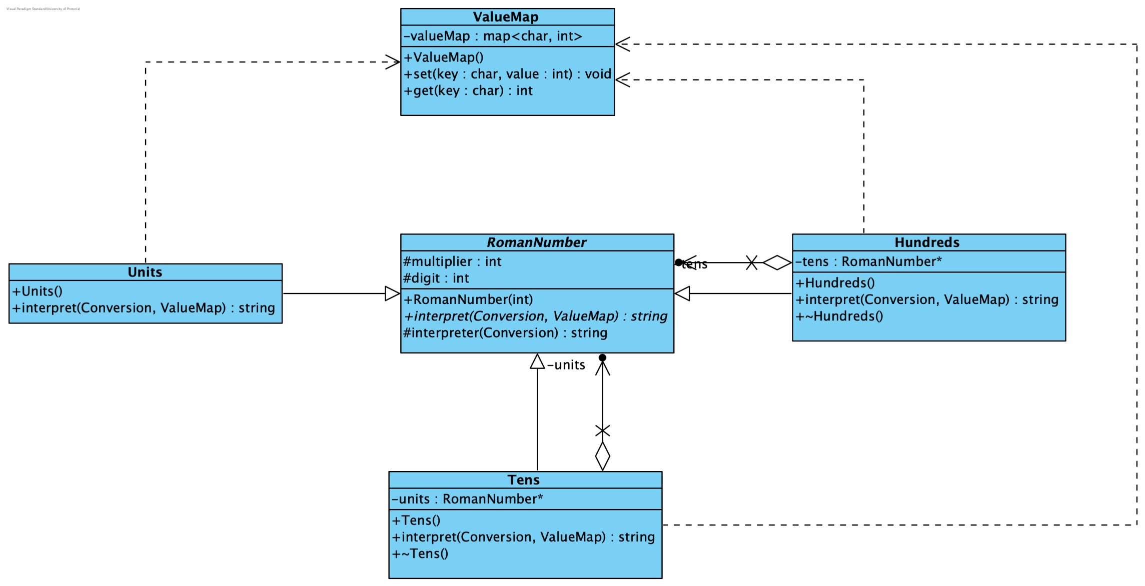
B. `Home`

C. `Tipi`

D. `CivilEngineer`

Question 4 ....(5 marks)

Consider the following UML class diagram and answer the questions that follow.


ValueMap
-valueMap : map<char, int>
+ValueMap()
+set(key : char, value : int) : void
+get(key : char) : int
RomanNumber
#multiplier : int
#digit : int
+RomanNumber(int)
+interpret(Conversion, ValueMap) : string
#interpreter(Conversion) : string
Units
+Units()
+interpret(Conversion, ValueMap) : string
Hundreds
-tens : RomanNumber*
+Hundreds()
+interpret(Conversion, ValueMap) : string
+~Hundreds()
-tens
-units
Tens
-units : RomanNumber*
+Tens()
+interpret(Conversion, ValueMap) : string
+~Tens()

4.1 Which pattern is shown in the given UML class diagram? (1)

A. Adapter

B. Builder

C. Composite

D. Decorator

E. Interpreter

4.2 The `Hundreds` class is referred to as the ________ participant of the pattern. (1)

A. Composite

B. Leaf

C. Nonterminal

D. Terminal

4.3 Which class in the diagram represents the `Context` participant of the pattern? (1)

A. Hundreds

B. RomanNumber

C. Tens

D. Units

E. ValueMap

4.4 What is the multiplicity of the aggregate relationship between the `Tens` and the `RomanNumbers` classes? (1)

A. 0

B. 1

C. 0..1

D. 0..n

E. 1..*

4.5 What is the largest integer value that can be converted to a Roman numeral when the design given in the above class diagram is implemented? The BNF for the conversion from integer to Roman numerals, on which the class diagram is based, is given by: (1)

```
RomanNumber ::= Hundreds Tens Units
Hundreds ::= LowHundreds | CD | D LowHundreds
LowHundreds ::= Empty | LowHundreds C
Tens ::= LowTens | XL | L LowTens | XC
LowTens ::= Empty | LowTens X
Units ::= LowUnits | IV | V LowUnits | IX
LowUnits ::= Empty | LowUnits I
```

A. 99

B. 199

C. 599

D. 899

E. 999

F. 1000

COS 214 Class Test 6 – L22 to L26

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

Cell: 0716407441
Email: u18050362@tuks.co.za

Question 1

1.1 C

1.2 A

1.3 E

1.4 F

1.5 B

Question 2

2.1 A

2.2 B

2.3 C

2.4 D

Question 3

3.1 B

3.2 C

3.3 D

3.4 A

Question 4

4.1 E

4.2 C

4.3 E

4.4 B

4.5 D

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

COS 214 Class Test 7 -(Optional) L27 to L32

- This test takes place on **3rd November 2020**.
- The maximum duration of this test is **40 minutes**.
- This test consists of **5 questions** for a total of **22 marks**.

Question 1 ....(5 marks)

Match the intent of the pattern to the pattern name.

1.1 Ensure a class has only one instance and provide a global point of access to it. (1)

A. Prototype

B. Singleton

C. Flyweight

1.2 Provide a surrogate or placeholder for another object to control access to it. (1)

A. Memento

B. Proxy

C. Strategy

1.3 Represent an operation to be performed on the elements of an object structure. The pattern lets you define a new operation without changing the classes of the elements on which it operates. (1)

A. Iterator

B. Visitor

C. None of the above

1.4 Without violating encapsulation, capture and externalise an object's internal state so that the object can be restored to this state later. (1)

A. Flyweight

B. Memento

C. State

1.5 Creating a website in steps. First the header, then the content, then the footer, and finally assembling the parts into a working website. (1)

A. Prototype

B. Template Method

C. Abstract Factory

D. Builder

Question 2 ....(4 marks)

For each of the following, indicate whether the statement is True or False.

2.1 The Builder pattern allows one to vary a product's internal representation. (1)

2.2 Typically, the Visitor design pattern is used in conjunction with the Adapter design pattern. (1)

2.3 The Singleton design pattern makes it easy to create an unlimited number of object instantiations. (1)

2.4 A Proxy design pattern is ideally suited for managing access to resources. (1)

Question 3 ....(3 marks)

The following code presents a template definition of the Singleton design pattern. Study the definition and answer the questions that follow.

```
template<typename T>
class Singleton {
public:
    static T& instance();
protected:
    Singleton() {}
    virtual ~Singleton() {}
private:
    Singleton(const Singleton&);
};
```

3.1 Which of the following is the best implementation for the public function? (1)

A.

```
T& Singleton::instance() {
    static T theInstance;
    return theInstance;
}
```

B.

```
T& Singleton::instance() {
    if (theInstance == 0)
        theInstance = new Singleton();
    return theInstance;
}
```

C.

```
template<typename T>
T& Singleton<T>::instance() {
    static T theInstance;
    return theInstance;
}
```

D.

```
template<typename T>
T& Singleton<T>::instance() {
    if (theInstance == 0)
        theInstance = new Singleton();
    return theInstance;
}
```

3.2 Consider the class definition and implementation, called `MyClass`, that makes use of the Singleton template class. (2)

```
class MyClass : public Singleton<MyClass> {
    friend class Singleton<MyClass>;
public:
    void setValue(int n) { x = n; }
    int getValue() const { return x; }
protected:
    MyClass() { x = 0; }
private:
    int x;
};
```

Which of the following statements would be the best way to set the value of attribute `x` of `MyClass`?

- **Option I.**

```
MyClass &m = MyClass::instance();
m.setValue(10);
```

- **Option II.**

```
MyClass::instance().setValue(10);
```

- **Option III.**

```
Singleton<MyClass>::instance().setValue(10);
```

A. Only Option I.

B. Only Option II.

C. Only Option III.

D. Options I. and II.

E. Options II. and III.

F. Options I. and III.

Question 4 ....(3 marks)

The instantiation of objects of a class `X` is extremely expensive. Clients need to be placed under the impression that they are dealing with objects of type `X` but due to memory constraints their creation should be delayed until absolutely necessary.

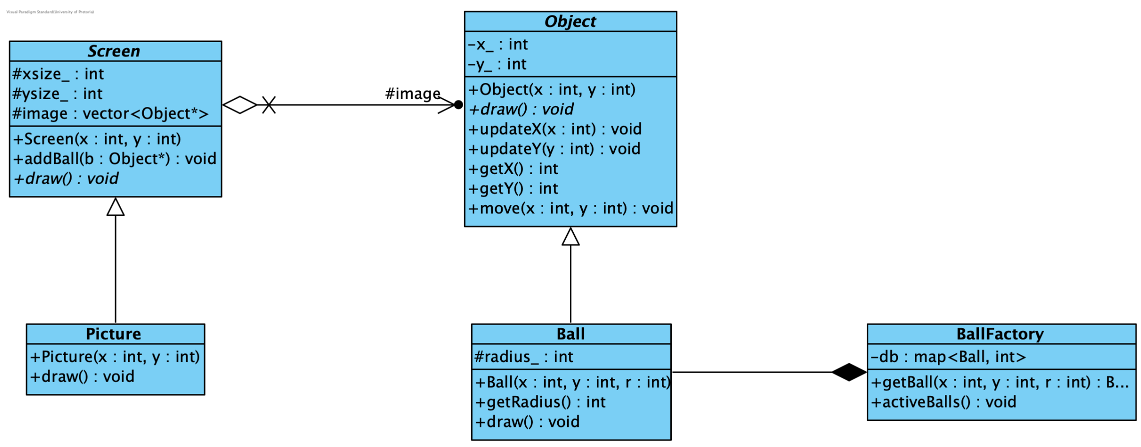
4.1 Which pattern is being described? (1)

4.2 Which participant does class `X` represent? (1)

4.3 The superclass of class `X` is referred to as the _____ participant of the pattern. (1)

Question 5 ....(7 marks)

Consider the following UML Class diagram and answer the questions that follow.



5.1 Which pattern is shown in the diagram? (1)

5.2 The class `Ball` identifies as which participant of the pattern? (1)

5.3 When a ball is placed in a picture, the BallFactory will apply reference counting for each specific ball. Will the following code implement the reference counting adequately? (1)

```
Ball* BallFactory::getBall(int x, int y, int r){
     Ball temp(x,y,r);
     db[temp]++;
     map<Ball, int>::iterator iter = db.find(temp);
     return const_cast<Ball*>(&(iter->first));
}
```

5.4 What will the output of the following code segment be? (3)

```
     BallFactory ballFactory;
     Picture* p;

     p  = new Picture(200,200);
     p->addBall(ballFactory.getBall(1,1,1));
     p->addBall(ballFactory.getBall(2,2,2));
     p->addBall(ballFactory.getBall(3,3,3));
     p->addBall(ballFactory.getBall(1,1,1));

     ballFactory.activeBalls();
```

If `activeBalls` is defined as follows:

```
void BallFactory::activeBalls(){
    map<Ball, int>::iterator iter;
    cout<<"The map:"<<endl;
    for (iter = db.begin(); iter != db.end(); iter++) {
        cout << "("<< iter->first << " : " << iter->second << ")" << endl;
    }
}
```

5.5 Consider the following implementation of the `draw()` function. (1)

```
void Ball::draw() {
     cout << *this << endl;
}

void Picture::draw(){
     cout<<"Picture screen: "<< endl;
     for_each(image.begin(), image.end(), mem_fun(&Object::draw));
     cout<<endl;
}
```

Provide the code to draw a picture. You may use the code segment given previously and draw that picture.

COS 214 Class test 7

Question 1

1.1 B
1.2 B
1.3 B
1.4 B
1.5 D

Question 2

2.1 True (internal representation = state)

2.2 False (visitor doesn't use an aggregate)

2.3 False

2.4 True

Question 3

3.1 C

3.2 E (can create multiple template singletons, but only ever one per data type)

Question 4

4.1 Proxy

4.2 RealSubject

4.3 Subject

Question 5

5.1 Flyweight

5.2 Concrete Flyweight

5.3 Yes

5.4 Ball(1,1,1) : 2

Ball(2,2,2) : 1

Ball(3,3,3) : 1

5.5 p->draw()