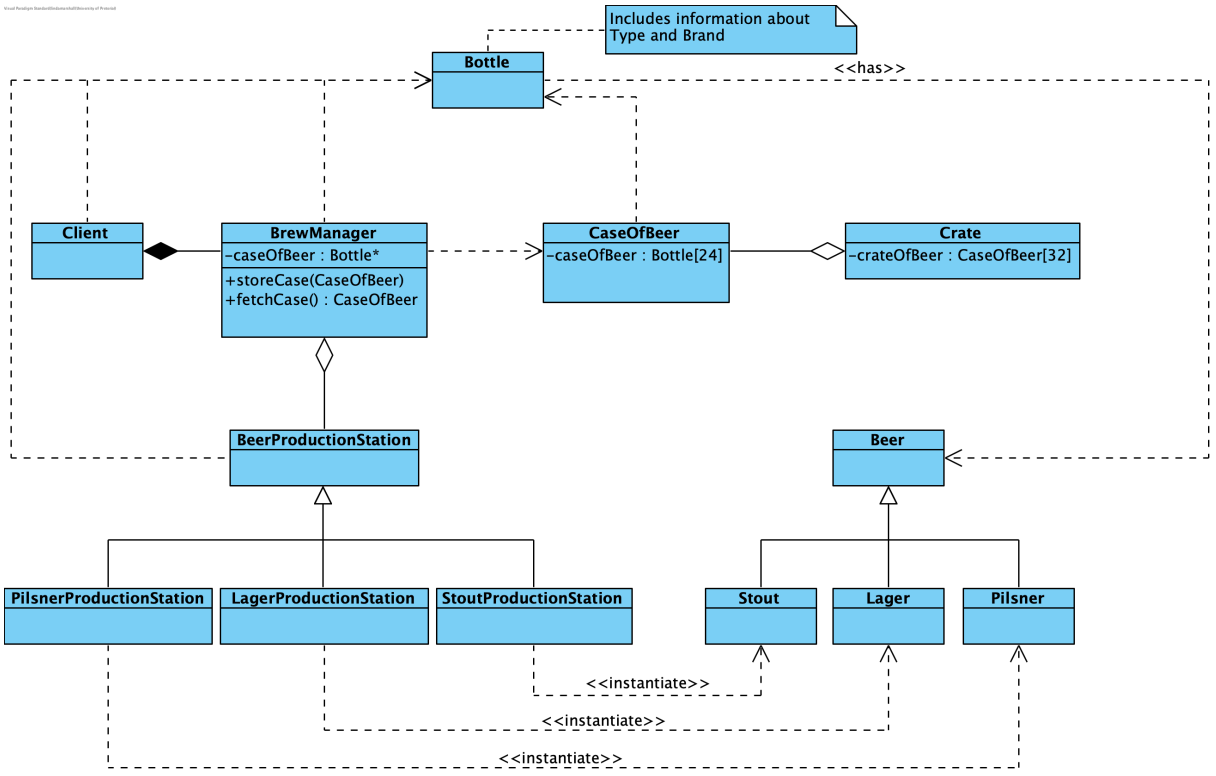
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

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 {
```

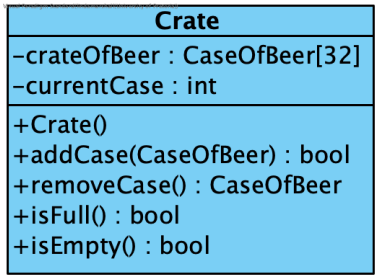
```
public:
  BrewManager();
  void setupBrewingStation(int);
  bool brewBeer(string);
  void cleanBrewingStation();
  vector<Bottle*> shipCrate();

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)



- 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)

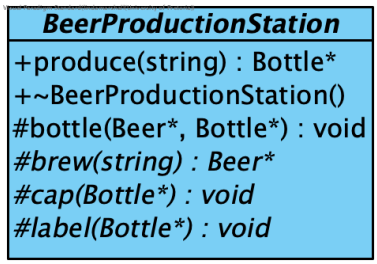
4.2 Write the code to set up a brewing station, brew the beer and clean the brewing station. (6)

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);
}
```

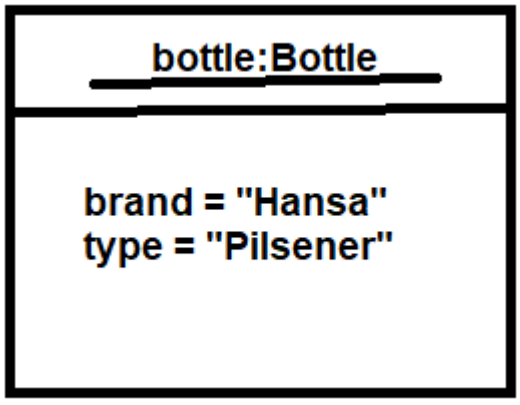
2.3 Wide interface between originator and memento enforcing narrow interface between memento and any other class.

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