FH-OÖ Hagenberg/HSD
SDP3, WS 2019
Übung 7

Name(1):	Daniel Weyrer	Abgabetermin:	07.01.20
Name(2):	Viktoria Streibl	Punkte:	22,5
Übungsgruppe:	Gruppe 1	korrigiert:	THP
Geschätzter Aufwand in Ph:	6 | 6	Effektiver Aufwand in Ph:	6 | 5

Beispiel 1 (24 Punkte) Kaffeeautomat: Entwerfen Sie aus der nachfolgenden Spezifikation ein Klassendiagramm, instanzieren Sie dieses und implementieren Sie die Funktionalität entsprechend. Verwenden Sie dabei das Decorator-Pattern:

Ein Kaffeeautomat bietet verschiedene Kaffeesorten (Verlängerter, Espresso, Koffeinfrei) mit entsprechenden Zutaten (Zucker, Milch u. Schlagobers) an. Die Kaffeesorten und Zutaten haben jeweils unterschiedliche Preise und eine entsprechende Beschreibung. Eine Methode `GetCost()` liefert den Gesamtpreis des ausgewählten Kaffees und die Methode `GetDescription()` liefert dazu die entsprechende Beschreibung als `std::string` um z.B. folgende Ausgaben auf `std::cout` zu ermöglichen:

```
Espresso: Zucker, Schlagobers 2.89 Euro
Verlängerter: Zucker, Milch 2.93 Euro
Koffeinfrei: Milch, Milch, Schlagobers 3.15 Euro
```

Die Beschreibung und die Preise werden in einer separaten Preisliste (Konstanten in Header, Klasse, oder Namespace) festgelegt. Zutaten können mehrfach gewählt werden!

Achten Sie beim Design darauf, dass zusätzliche Kaffeesorten und Zutaten hinzugefügt werden können, ohne die bereits bestehenden Klassen verändern zu müssen. Beweisen Sie dies durch das Hinzufügen der Kaffeesorte "Mocca" und der Zutat "Sojamilch".

Implementieren Sie einen Testtreiber der verschiedene Kaffees mit unterschiedlichen Zutaten erzeugt, alle Methoden ausreichend testet und anschließend deren Beschreibung auf `std::cout` ausgibt.

Implementieren Sie weiters eine Klasse `CoffeePreparation` die nach dem FIFO-Prinzip arbeitet und folgende Schnittstelle aufweist:

```
void Prepare(/*Coffee*/);        //adds and prepares a coffee
void Display(std::ostream& os);  //outputs all coffees in preparation
/*Coffee*/ Finished();                //removes the prepared coffee
```

Testen Sie die Klasse ebenfalls ausführlich im Testtreiber!

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit Kommentaren! Testen Sie ihre Implementierungen ausführlich! Geben Sie den **Testoutput** mit ab!

SDP - Exercise 07

winter semester 2019/20

Viktoria Streibl - S1810306013
Daniel Weyrer - S1820306044

January 7, 2020

Contents

1 Organizational

1.1 Team

- Viktoria Streibl - S1810306013
- Daniel Weyrer - S1820306044

1.2 Roles and responsibilities

1.2.1 Jointly

- Planning
- Documentation
- Systemdocumentation

1.2.2 Viktoria Streibl

- Object
- Pricelist
- Coffee Sorts
- Ingredient Sorts

1.2.3 Daniel Weyrer

- Coffeemachine
- TestDriver
- CoffeePreparation
- Ingredient

1.3 Effort

1.3.1 Viktoria Streibl

- estimated: 6 ph
- actually: 5 ph

1.3.2 Daniel Weyrer

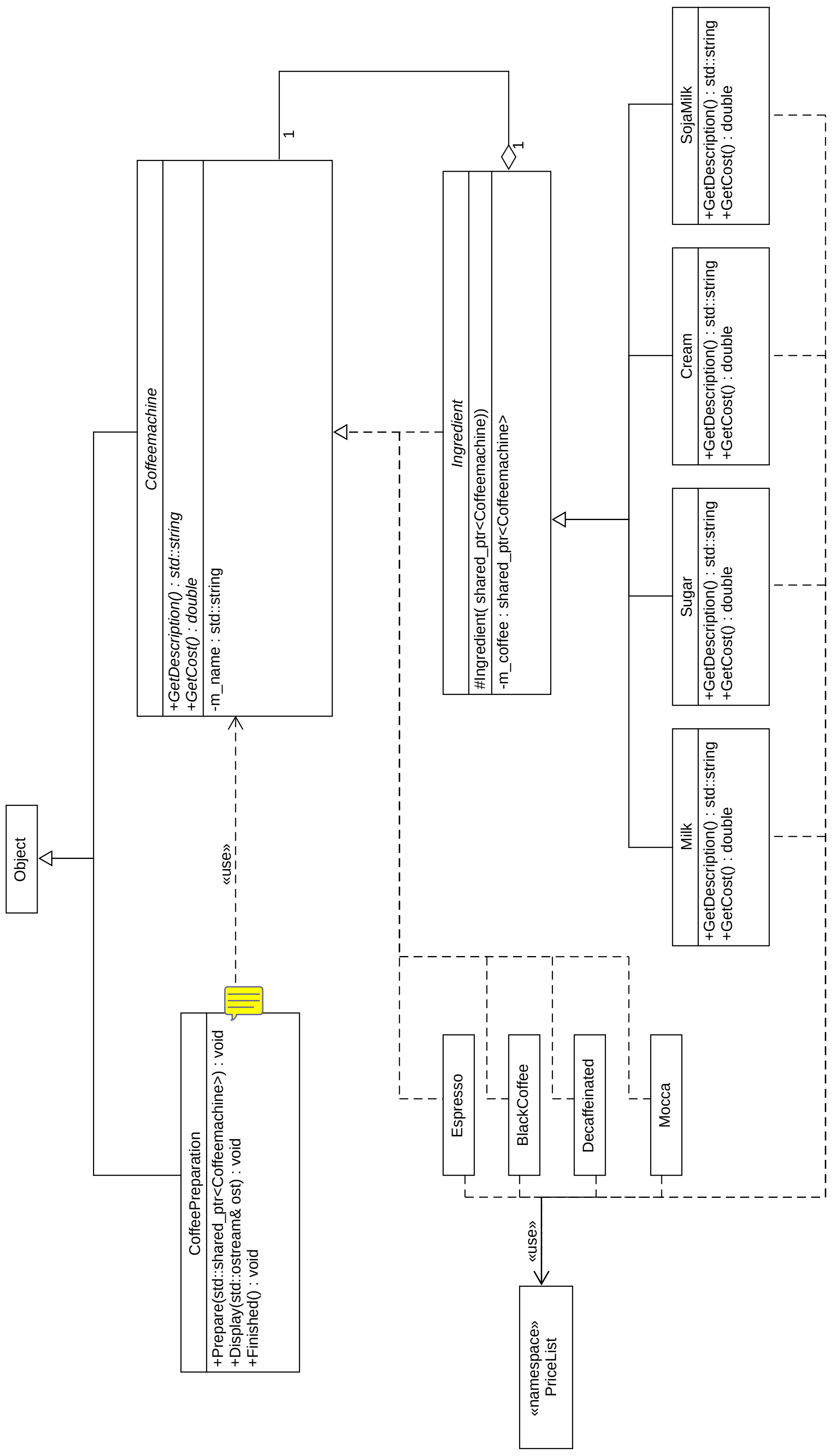
- estimated: 6 ph
- actually: 6 ph

2 Requirenment Definition(System Specification)

This Coffeemachine should work like a normal Coffeemachine. It is a simulation to order different sort of coffee and add several ingredients. Depending on the selection the price will be displayed. The Client is implemented as the PrepareCoffee - Class and uses the Coffeemachine to create those coffees (which basically means to calculate the price for the first item added with its added ingredients) and removes the printed coffee from the list.

3 System Design

3.1 Classdiagram


Coffeemachine
+GetDescription() : std::string
+GetCost() : double
-m_name : std::string
1
1
Ingredient
#Ingredient(shared_ptr<Coffeemachine>)
-m_coffee : shared_ptr<Coffeemachine>
SojaMilk
+GetDescription() : std::string
+GetCost() : double
Cream
+GetDescription() : std::string
+GetCost() : double
Sugar
+GetDescription() : std::string
+GetCost() : double
Milk
+GetDescription() : std::string
+GetCost() : double
Object
«use»
CoffeePreparation
+Prepare(std::shared_ptr<Coffeemachine>) : void
+Display(std::ostream& ost) : void
+Finished() : void
Espresso
BlackCoffee
Decaffeinated
Mocca
«use»
«namespace»
PriceList

3.2 Design Decisions

3.2.1 PriceList

We decided to use an extra file and seperate namespaces for coffees and ingredients to manage the different prices. This makes it easier to add change the prices afterwards.

We did not use another base class for all coffees as it would have been completely empty (the main difference between ingredient and coffees are the GetCost and GetDescription Functions)

4 Component Design

4.1 CoffeePreparation

It contains following Methods:

- Prepare
- Display
- Finished

Prepare adds a shared ptr to a coffee to a list. When "Finished" is being called, the last item in the (first one added) is being printed and deleted from the list straight afterwards.

We used a list, as we‘re adding items in the front of the container (which is not possible with e.g. a vector).

4.2 Coffeemachine

It contains following Methods:

- GetDescription
- GetCost

4.3 Pricelist

It contains a namespace where the prices of the different ingredients are declared.

4.4 Ingredient

Just extends the base class by a shared pointer to the concrete component and its constructor! Base Class for all Ingredients.

4.5 Coffee Sorts

There are different kinds of coffee, the following are implemented:

- Espresso
- Black Coffee
- Decaffeinated
- Mocca

4.6 Ingredient Sorts

There are different kinds of coffee, the following are implemented:

- Milk
- Sugar
- Cream
- Soja Milk

They also have following methods:

- GetDescription
- GetCost

Both of the Methods are calling they‘re predecessors and add their value (which they get from the constants in the namespace), before returning it!

5 Source Code

5.1 CoffeePreparation

5.1.1 CoffeePreparation.h

```cpp
/* -----------------------------------------------------------------------
| Workfile : CoffeePreparation.h
| Description : [ HEADER ]
| Name : Daniel Weyrer      PKZ : S1820306044
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */


#ifndef COFFEEPREPARATION_H
#define COFFEEPREPARATION_H

#include "Object.h"

#include "Coffeemachine.h"
#include "Espresso.h"
#include "BlackCoffee.h"
#include "Decaffeinated.h"

#include "Milk.h"
#include "Mocca.h"
#include "SojaMilk.h"
#include "Sugar.h"
#include "Cream.h"

#include <memory>
#include <list>
#include <iostream>

class CoffeePreparation : public Object {

public:
  void Prepare(Coffeemachine::SPtr const& coffee);
  void Display(std::ostream& os);

  //Prints prepared coffee with all ingredients and the price!
  void Finished();

private:
  std::list<Coffeemachine::SPtr> m_Ingredients;
};

#endif //!COFFEEPREPARATION_H
```

5.1.2 CoffeePreparation.cpp

```cpp
/* -----------------------------------------------------------------------
| Workfile : CoffeePreparation.cpp
| Description : [ SOURCE ]
| Name : Daniel Weyrer      PKZ : S1820306044
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#include "CoffeePreparation.h"

void CoffeePreparation::Prepare(Coffeemachine::SPtr const& coffee) {
  if (coffee != nullptr) {
    m_Ingredients.emplace_front(coffee);
  }
  else {
    std::cerr << "Please check given Parameter - nullptr detected!" << std::endl;
  }
}

void CoffeePreparation::Display(std::ostream& os) {
  if (os.good()) {
    for (auto elem : m_Ingredients) {
      os << elem->GetDescription() << std::endl;
    }
  }
}

void CoffeePreparation::Finished() {
  if (!m_Ingredients.empty()) {
    auto temp = m_Ingredients.back();
    std::cout << temp->GetDescription() << " = " << temp->GetCost() << std::endl;
    m_Ingredients.pop_back();
  }
  else {
    std::cerr << "No Coffee in the list!" << std::endl;
  }
}
```

5.2 Coffeemachine

5.2.1 Coffeemachine.h

```cpp
/* -----------------------------------------------------------------------
| Workfile : Coffeemachine.h
| Description : [ HEADER ]
| Name : Daniel Weyrer      PKZ : S1820306044
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#ifndef COFFEEMACHINE_H
#define COFFEEMACHINE_H

#include "Object.h"

#include <string>
#include <memory>

class Coffeemachine : public Object{
public:

  void GetDisplay();
  virtual std::string GetDescription() = 0;
  virtual double GetCost() = 0;

  using SPtr = std::shared_ptr<Coffeemachine>;

private:
  std::string m_name;

};


#endif //!COFFEEMACHINE_H
```

5.2.2 Coffeemachine.cpp

```cpp
/* -----------------------------------------------------------------------
| Workfile : Coffeemachine.cpp
| Description : [ SOURCE ]
| Name : Daniel Weyrer      PKZ : S1820306044
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#include <iostream>
#include "Coffeemachine.h"



void Coffeemachine::GetDisplay() {
}
```

5.3 Espresso

5.3.1 Espresso.h

```
/* ----------------------------------------------------------------------
| Workfile : Espresso.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */


#ifndef ESPRESSO_H
#define ESPRESSO_H

#include <string>

#include "Pricelist.h"
#include "Coffeemachine.h"

class Espresso : public Coffeemachine {

  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!ESPRESSO_H
```

5.3.2 Espresso.cpp

```
/* ----------------------------------------------------------------------
| Workfile : Espresso.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */

#include "Espresso.h"

using namespace pricelist;

std::string Espresso::GetDescription() {
  return "Espresso:";
}

double Espresso::GetCost() {
  double price = coffee::espresso;

  return price;
}
```

5.4 BlackCoffee

5.4.1 BlackCoffee.h

```
/* ----------------------------------------------------------------------
| Workfile : BlackCoffee.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */


#ifndef BLACKCOFFEE_H
#define BLACKCOFFEE_H

#include <string>

#include "Pricelist.h"
#include "Coffeemachine.h"

class BlackCoffee : public Coffeemachine{
  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!BLACKCOFFEE_H
```

5.4.2 BlackCoffee.cpp

```
/* ----------------------------------------------------------------------
| Workfile : BlackCoffee.h
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */

#include "BlackCoffee.h"

using namespace pricelist;

std::string BlackCoffee::GetDescription() {
  return "Black Coffee: ";
}

double BlackCoffee::GetCost() {
  double price = coffee::blackcoffee;

  return price;
}
```

5.5 Decaffeinated

5.5.1 Decaffeinated.h

```cpp
/* -----------------------------------------------------------------------
| Workfile : Decaffeinated.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */


#ifndef DECAFFEINATED_H
#define DECAFFEINATED_H

#include <string>

#include "Pricelist.h"
#include "Coffeemachine.h"

class Decaffeinated : public Coffeemachine {

  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!DECAFFEINATED_H
```

5.5.2 Decaffeinated.cpp

```cpp
/* -----------------------------------------------------------------------
| Workfile : Decaffeinated.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#include "Decaffeinated.h"

using namespace pricelist;

std::string Decaffeinated::GetDescription() {
  return "Decaffeinated: ";
}

double Decaffeinated::GetCost() {
  double price = coffee::decaffeinated;

  return price;
}
```

5.6 Mocca

5.6.1 Mocca.h

```
/* -----------------------------------------------------------------------
| Workfile : Mocca.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */


#ifndef COFFEE_H
#define COFFEE_H

#include <string>

#include "Pricelist.h"
#include "Coffeemachine.h"

class Mocca : public Coffeemachine{

  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!COFFEE_H
```

5.6.2 Mocca.cpp

```
/* -----------------------------------------------------------------------
| Workfile : Mocca.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#include "Mocca.h"

using namespace pricelist;

std::string Mocca::GetDescription() {
  return "Mocca: ";
}

double Mocca::GetCost() {
  double price = coffee::mocca;

  return price;
}
```

5.7 Ingredient

5.7.1 Ingredient.h

```
/* -----------------------------------------------------------------------
| Workfile : Ingredient.h
| Description : [ HEADER ]
| Name : Daniel Weyrer      PKZ : S1820306044
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */


#ifndef INGREDIENT_H
#define INGREDIENT_H

#include "Coffeemachine.h"

#include <vector>
#include <memory>
#include <string>


class Ingredient : public Coffeemachine {
public:
  virtual std::string GetDescription();
  virtual double GetCost();

protected:
  Ingredient(Coffeemachine::SPtr const& currCoffee);

private:
  std::shared_ptr<Coffeemachine> m_coffee;
};

#endif //!INGREDIENT_H
```

5.7.2 Ingredient.cpp

```
/* -----------------------------------------------------------------------
| Workfile : Ingredient.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#include "Ingredient.h"
#include <iostream>

std::string Ingredient::GetDescription() {
  return m_coffee->GetDescription();
}

double Ingredient::GetCost() {
  return m_coffee->GetCost();
}

Ingredient::Ingredient(Coffeemachine::SPtr const& currCoffee) {
  try {
    if (currCoffee == nullptr) {
      throw("Null-pointer!");
    }

    m_coffee = currCoffee;
  }
  catch (std::exception const& ex) {
    std::cerr << "Exception in Ingredient CTor!" << ex.what() << std::endl;
  }
}
```

5.8 Milk

5.8.1 Milk.h

```
/* ----------------------------------------------------------------------
| Workfile : Milk.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */


#ifndef MILK_H
#define MILK_H

#include <string>
#include <memory>

#include "Pricelist.h"
#include "Ingredient.h"

class Milk : public Ingredient {
public:
  Milk(std::shared_ptr<Coffeemachine> const& currCoffee) : Ingredient{ currCoffee } {}

  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!MILK_H
```

5.8.2 Milk.cpp

```
/* ----------------------------------------------------------------------
| Workfile : Milk.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */

#include "Milk.h"

#include <iostream>

using namespace pricelist;

std::string Milk::GetDescription() {
  return Ingredient::GetDescription() + "Mil[illegible]";
}


double Milk::GetCost() {
  return Ingredient::GetCost() + ingredients::milk;
}
```

5.9 Sugar

5.9.1 Sugar.h

```
/* ----------------------------------------------------------------------
| Workfile : Sugar.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */


#ifndef SUGAR_H
#define SUGAR_H

#include <string>
#include <memory>

#include "Pricelist.h"
#include "Ingredient.h"

class Sugar : public Ingredient {
public:
  Sugar(std::shared_ptr<Coffeemachine> const& currCoffee) : Ingredient{ currCoffee } {}

  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!SUGAR_H
```

5.9.2 Sugar.cpp

```
/* ----------------------------------------------------------------------
| Workfile : Sugar.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */

#include "Sugar.h"
#include <iostream>

using namespace pricelist;

std::string Sugar::GetDescription() {
  return Ingredient::GetDescription() + " Sugar ";
}
double Sugar::GetCost() {
  return Ingredient::GetCost() + ingredients::sugar;
}
```

5.10 Cream

5.10.1 Cream.h

```cpp
/* ----------------------------------------------------------------------
| Workfile : Cream.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */


#ifndef CREAM_H
#define CREAM_H

#include <string>

#include "Pricelist.h"
#include "Ingredient.h"

class Cream : public Ingredient {
public:
  Cream(std::shared_ptr<Coffeemachine> const& currCoffee) : Ingredient{ currCoffee } {}

  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!CREAM_H
```

5.10.2 Cream.cpp

```cpp
/* ----------------------------------------------------------------------
| Workfile : Cream.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */

#include "Cream.h"
#include <iostream>

using namespace pricelist;

std::string Cream::GetDescription() {
  return Ingredient::GetDescription() + " Cream ";
}
double Cream::GetCost() {
  return Ingredient::GetCost() + ingredients::cream;
}
```

5.11 SojaMilk

5.11.1 SojaMilk.h

```
/* -----------------------------------------------------------------------
| Workfile : SojaMilk.h
| Description : [ HEADER ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */


#ifndef SOJAMILK_H
#define SOJAMILK_H

#include <string>
#include <memory>

#include "Pricelist.h"
#include "Ingredient.h"

class SojaMilk : public Ingredient {

public:
  SojaMilk(std::shared_ptr<Coffeemachine> const& currCoffee) : Ingredient{ currCoffee } {}

  std::string GetDescription() override;
  double GetCost() override;
};

#endif //!SOJAMILK_H
```

5.11.2 SojaMilk.cpp

```
/* -----------------------------------------------------------------------
| Workfile : SojaMilk.cpp
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#include "SojaMilk.h"

#include <iostream>

using namespace pricelist;


std::string SojaMilk::GetDescription() {
  return Ingredient::GetDescription() + " SojaMilk ";
}

double SojaMilk::GetCost() {
  return Ingredient::GetCost() + ingredients::sojaMilk;
}
```

5.12 PriceList

5.12.1 Pricelist.h

```
/* ----------------------------------------------------------------------
| Workfile : Pricelist.h
| Description : [ SOURCE ]
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 06.01.20
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */

#ifndef PRICELIST_H
#define PRICELIST_H

namespace pricelist {

  namespace coffee {
    const double blackcoffee = 1;
    const double mocca = 1.20;
    const double espresso = 1;
    const double decaffeinated = 0.8;
  }

  namespace ingredients {
    const double milk = 0.25;
    const double sugar = 0.1;
    const double sojaMilk = 0.3;
    const double cream = 0.5;
  }
}

#endif //!PRICELIST_H
```

5.13 TestDriver

```cpp
#include <iostream>

#include "CoffeePreparation.h"


#include <memory>

using namespace std;

int main() {
  CoffeePreparation prep;

  Coffeemachine::SPtr c1{ make_shared<Espresso>() };
  Coffeemachine::SPtr c2{ make_shared<Mocca>() };

  prep.Prepare(make_shared<SojaMilk>(make_shared<Sugar>(c1)));
  prep.Prepare(make_shared<Sugar>(make_shared<Cream>(make_shared<SojaMilk>(c2))));

  prep.Display(cout);
  prep.Finished();
  prep.Finished();

}
```

5.14 Output

```
Mocca:  SojaMilk  Cream  Sugar
Espresso: Sugar  SojaMilk
Espresso: Sugar  SojaMilk  = 1.4
Mocca:  SojaMilk  Cream  Sugar  = 2.1
```