

به نام خدا

گزارش تمرین سری پنجم AP

محمد مهدی مالموردی

9723079

استاد جهانشاهی

توابع پیاده سازی شده در ingredient.h به صورت زیر می باشد:

```
include > C ingredient.h
1  #ifndef INGREDIENT_H
2  #define INGREDIENT_H
3
4  class Ingredient
5  {
6  public:
7      double get_price_unit(){return price_unit;}
8      size_t get_units(){return units;}
9      std::string get_name(){return name;}
10
11      double price(){return price_unit*units;}
12
13
14  protected:
15      Ingredient(double price_unit, size_t units) : price_unit{price_unit},units{units}
16      {
17      }
18
19
20      double price_unit;
21      size_t units;
22      std::string name;
23  };
24
25
26  #endif // INGREDIENT_H
```

فراخوانی و تعریف توابع در ingredient.h

```

include > C sub_ingredients.h
1  #ifndef SUB_INGREDIENTS_H
2  #define SUB_INGREDIENTS_H
3
4  #include "ingredient.h"
5
6  /*
7  class Cinnamon : public Ingredient
8  {
9  public:
10     Cinnamon(size_t units) : Ingredient(5, units)
11     {
12         |   this->name = "Cinnamon";
13     }
14
15     virtual std::string get_name() {return this->name;}
16 };
17
18 class Chocolate : public Ingredient
19 {
20 public:
21     Chocolate(size_t units) : Ingredient(5, units)
22     {
23         |   this->name = "Chocolate";
24     }
25
26     virtual std::string get_name() {return this->name;}
27 };
28
29 class Sugar : public Ingredient
30 {
31 public:
32     Sugar(size_t units) : Ingredient(1, units)
33     {
34         |   this->name = "Sugar";
35     }
36
37     virtual std::string get_name() {return this->name;}
38 };
39
40 class Cookie : public Ingredient
41 {
42 public:
43     Cookie(size_t units) : Ingredient(10, units)
44     {
45         |   this->name = "Cookie";
46     }
47
48     virtual std::string get_name() {return this->name;}
49 };
50
51 class Espresso : public Ingredient
52 {
53 public:
54     Espresso(size_t units) : Ingredient(15, units)
55     {
56         |   this->name = "Espresso";
57     }
58
59     virtual std::string get_name() {return this->name;}
60 };
61
62 class Milk : public Ingredient
63 {
64 public:
65     Milk(size_t units) : Ingredient(10, units)
66     {
67         |   this->name = "Milk";
68     }
69
70     virtual std::string get_name() {return this->name;}
71 };
72
73 class MilkFoam : public Ingredient
74 {
75 public:
76     MilkFoam(size_t units) : Ingredient(5, units)
77     {
78         |   this->name = "MilkFoam";
79     }
80
81     virtual std::string get_name() {return this->name;}
82 };

```

```

82 };
83
84 class Water : public Ingredient
85 {
86 public:
87     Water(size_t units) : Ingredient(1, units)
88     {
89         this->name = "Water";
90     }
91
92     virtual std::string get_name() {return this->name;}
93 };
94 */
95
96 #define DEFCLASS(NAME, PRICE) \
97     class NAME : public Ingredient \
98     { \
99     public: \
100         NAME(size_t units) \
101             : Ingredient(PRICE, units) \
102         { \
103             name = #NAME; \
104         } \
105         virtual std::string get_name() { return name; } \
106     };
107
108 DEFCLASS(Cinnamon, 5);
109 DEFCLASS(Chocolate, 5);
110 DEFCLASS(Sugar, 1);
111 DEFCLASS(Cookie, 10);
112 DEFCLASS(Espresso, 15);
113 DEFCLASS(Milk, 10);
114 DEFCLASS(MilkFoam, 5);
115 DEFCLASS(Water, 1);
116
117
118
119 #endif // SUB_INGREDIENTS_H

```

فراخوانی و تعریف توابع در challenge + sub\_ingredients.h

```

include > C espresso_based.h
1  #ifndef ESPRESSO_BASED_H
2  #define ESPRESSO_BASED_H
3
4  #include <string>
5  #include <vector>
6  #include "ingredient.h"
7
8  class EspressoBased
9  {
10 public:
11     virtual std::string get_name() = 0;
12     virtual double price() = 0;
13
14     void brew(){}
15     std::vector<Ingredient*> get_ingredients();
16
17     virtual ~EspressoBased();
18
19 protected:
20     EspressoBased();
21     EspressoBased(const EspressoBased& esp);
22     void operator=(const EspressoBased& esp);
23
24     std::vector<Ingredient*> ingredients;
25     std::string name;
26
27 };
28
29
30 #endif // ESPRESSO_BASED_H

```

فراخوانی توابع در espresso\_based.h

```

src > espresso_based.cpp
1  #include "espresso_based.h"
2
3
4  EspressoBased::EspressoBased(const EspressoBased& esp){
5
6
7      name = esp.name;
8      ingredients = esp.ingredients;
9
10 }
11
12 EspressoBased::EspressoBased(){
13     ingredients.clear();
14     name = "";
15 }
16 };
17
18 std::vector<Ingredient*>& EspressoBased::get_ingredients(){return ingredients;}
19
20 void EspressoBased::operator=(const EspressoBased& esp){
21     name = esp.name;
22     ingredients = esp.ingredients;
23 }
24
25
26 EspressoBased::~EspressoBased()
27 {
28     for(auto& i : ingredients){
29         i = nullptr;
30         delete i;}
31     ingredients.clear();
32 }
33

```

تعریف توابع در espresso\_based.cpp

```

include > C cappuccino.h
1  #ifndef CAPPUCCINO
2  #define CAPPUCCINO
3
4  #include <string>
5  #include <vector>
6  #include "sub_ingredients.h"
7  #include "espresso_based.h"
8
9  class Cappuccino : public EspressoBased
10 {
11 public:
12     Cappuccino();
13     Cappuccino(const Cappuccino& cap);
14     ~Cappuccino();
15     void operator=(const Cappuccino& cap);
16
17     virtual std::string get_name();
18     virtual double price();
19
20     void add_side_item(Ingredient* side);
21     std::vector<Ingredient*> get_side_items();
22
23 private:
24     std::vector<Ingredient*> side_items;
25
26 };
27
28 #endif // CAPPUCCINO

```

فراخوانی توابع در cappuccino.h

```

src > G+ cappuccino.cpp
1  #include "cappuccino.h"
2
3
4  Cappuccino::Cappuccino() : EspressoBased()
5  {
6      side_items.clear();
7      name = "Cappuccino";
8      ingredients.push_back(new Espresso(2));
9      ingredients.push_back(new Milk(2));
10     ingredients.push_back(new MilkFoam(1));
11 }
12 Cappuccino::Cappuccino(const Cappuccino& cap){
13     side_items = cap.side_items;
14     name = cap.name;
15     ingredients = cap.ingredients;
16 }
17
18 void Cappuccino::operator=(const Cappuccino& cap){
19     side_items = cap.side_items;
20     name = cap.name;
21     ingredients = cap.ingredients;
22 }
23
24 std::string Cappuccino::get_name(){return name;}
25 double Cappuccino::price(){
26     double sum{0};
27
28     for(auto i : ingredients){
29         sum += i->price();
30     }
31     for(auto i : side_items){
32         sum += i->price();
33     }
34     return sum;
35 }
36
37
38 void Cappuccino::add_side_item(Ingredient* side){
39     side_items.push_back(side);
40 };
41 std::vector<Ingredient*> Cappuccino::get_side_items(){return side_items;}
42
43
44 std::vector<Ingredient*> Cappuccino::get_side_items(){return side_items;}
45
46 Cappuccino::~Cappuccino()
47 {
48     for(auto& i : side_items){
49         i = nullptr;
50         delete i;}
51     side_items.clear();
52 }

```

تعریف توابع در cappuccino.cpp



```

include > C mocha.h
1  #ifndef MOCHA_H
2  #define MOCHA_H
3
4
5  #include <string>
6  #include <vector>
7  #include "sub_ingredients.h"
8  #include "espresso_based.h"
9
10 class Mocha : public EspressoBased
11 {
12 public:
13     Mocha();
14     Mocha(const Mocha& moc);
15     ~Mocha();
16     void operator=(const Mocha& moc);
17
18     virtual std::string get_name();
19     virtual double price();
20
21     void add_side_item(Ingredient* side);
22     std::vector<Ingredient*> get_side_items();
23
24 private:
25     std::vector<Ingredient*> side_items;
26
27 };
28
29 #endif // MOCHA_H

```

فراخوانی توابع در mocha.h

```

src > G mocha.cpp
1  #include "mocha.h"
2
3  Mocha::Mocha() : EspressoBased()
4  {
5      side_items.clear();
6      name = "Mocha";
7      ingredients.push_back(new Espresso{2});
8      ingredients.push_back(new Milk{2});
9      ingredients.push_back(new MilkFoam{1});
10     ingredients.push_back(new Chocolate{1});
11 }
12 Mocha::Mocha(const Mocha& moc){
13     side_items = moc.side_items;
14     name = moc.name;
15     ingredients = moc.ingredients;
16 }
17
18 void Mocha::operator=(const Mocha& moc){
19     side_items = moc.side_items;
20     name = moc.name;
21     ingredients = moc.ingredients;
22 }
23
24 std::string Mocha::get_name(){return name;}
25 double Mocha::price(){
26     double sum{0};
27
28     for(auto i : ingredients){
29         sum += i->price();
30     }
31     for(auto i : side_items){
32         sum += i->price();
33     }
34     return sum;
35 }
36
37 void Mocha::add_side_item(Ingredient* side){
38     side_items.push_back(side);
39 };
40 std::vector<Ingredient*> Mocha::get_side_items(){return side_items;}
41 std::vector<Ingredient*> Mocha::get_side_items(){return side_items;}
42
43 Mocha::~Mocha()
44 {
45     for(auto& i : side_items){
46         i = nullptr;
47         delete i;
48     }
49     side_items.clear();
50 }

```

تعریف توابع در mocha.cpp

کلاس ingredient و توابع آن به مانند هر کلاس ساده و توابع معمول آن نوشته شدند و چالش خاصی نداشتند.

کلاس sub\_ingredients را ابتدا به صورت کپی کردن از نمونه نوشته شده در صورت تمرین نوشتم و برنامه درست کار کرد. در انتها برای انجام چالش با کمی جستجو، کلاس قالبی به کمک ماکرو ها تعریف کرده و از کپی پیست کردن های بیهوده جلوگیری کردیم.

```
#define DEFCLASS(NAME, PRICE) \
    class NAME : public Ingredient \
    { \
    public: \
        NAME(size_t units) \
            : Ingredient{PRICE, units} \
        { \
            name = #NAME; \
        } \
        virtual std::string get_name() { return name; } \
    };

DEFCLASS(Cinnamon, 5);
DEFCLASS(Chocolate, 5);
DEFCLASS(Sugar, 1);
DEFCLASS(Cookie, 10);
DEFCLASS(Espresso, 15);
DEFCLASS(Milk, 10);
DEFCLASS(MilkFoam, 5);
DEFCLASS(Water, 1);
```

باید دقت کنیم که چه در sub\_ingredients و چه در بقیه کلاس های چایلد و مشتق شده، ارث بری ها به صورت پابلیک اند که دسترسی ها مختل نشود. همچنین هم در مقابل تعریف کلاس و هم کانستراکترش باید این ارث بری اعلان شود.

همچنین مجبور شدم که دیستراکتور کلاس EspressoBased را نیز virtual کنم تا تست های شش و ده به درستی کار کنند.

اگر EspressoBased~ در قسمت protected قرار می گرفت به ارور زیر بر می خوردیم:

```
/usr/src/app/src/unit_test.cpp:68:12: error: 'EspressoBased::~EspressoBased()' is protected within this context
68 |     delete esp;
    |           ^
In file included from /usr/src/app/src/unit_test.cpp:5:
/usr/src/app/include/espresso_based.h:23:5: note: declared protected here
23 |     ~EspressoBased();
    |     ^
/usr/src/app/src/unit_test.cpp: In member function 'virtual void Hm5Test_TEST10_Test::TestBody()':
/usr/src/app/src/unit_test.cpp:111:12: error: 'EspressoBased::~EspressoBased()' is protected within this context
111 |     delete esp;
    |           ^
```

که چون حافظه دینامیک داریم و در تست ها از دستور delete استفاده می کنیم، این دسترسی ما را محدود می کند و نمی توانیم دیگر به همان شکل از دستوراتی مثل delete استفاده کنیم. برای همین آن را در قسمت پابلیک می گذاریم.

باید حواسمان باشد که برای محاسبه price در کاپوچینو و موکا، قیمت side\_items را هم به قیمت بقیه بخش ها اضافه کنیم.

همچنین دستورات دیستراکتور ها را کمی باید تغییر می دادیم و قبل دیلیت کردن حافظه دینامیک، آن را برابر با nullptr نیز قرار می دادیم.

لینک گیت هاب:

<https://github.com/mahdimld/AP-HW05>