/\*\*

\* Author: Narendra Pershad

\* Date: January 17, 2022

\* Filename: Demo02.cpp

\*/

#include <iostream>

#include "pizza.hpp"

#include "parent.hpp"

#include "child.hpp"

void test\_pizza();

void test\_parent();

int main()

{

//test\_pizza();

test\_parent();

}

void test\_pizza()

{ Demo02::Pizza{ "small" };//anonymous object

Demo02::Pizza p1{ "medium" };//another object

std::cout << p1 << '\n';

Demo02::Pizza p2 = Demo02::Pizza{ "large" };//another way

p2.add\_topping("pineapple");

std::cout << p2 << '\n';

Demo02::Pizza p3 = p2;

p3.add\_topping("green olive");

std::cout << p3 << '\n';

p2.add\_topping("hot pepper");

std::cout << p2 << '\n';

}

void test\_parent()

{

//Week02::Parent p1{};

//std::cout << p1;

//Week02::Parent p2 = p1;

Week02::Child c1{};

Week02::Child c2 = c1;

std::cout << c1 << '\n';

std::cout << c2 << '\n';

}

/\*\*

\* Author: Narendra Pershad

\* Date: January 17, 2022

\* Filename: pizza.hpp

\*/

#pragma once

#include <string>

#include <vector>

#include <map>

namespace Demo02

{

class Pizza

{

public:

Pizza(std::string, std::string topping = "cheese"); //constructor with a default argument

~Pizza(); //destructor

Pizza(const Pizza&); //copy constructor

void add\_topping(std::string);

void remove\_topping(std::string);

friend std::ostream& operator<<(std::ostream&, const Pizza&);

private:

std::string size\_;

std::vector<std::string> toppings\_;

const std::map<std::string, double> prices\_ =

{

{"small", 6.99},

{"medium", 8.99},

{"large", 9.99},

};

};

}

/\*\*

\* Author: Narendra Pershad

\* Date: January 17, 2022

\* Filename: pizza.cpp

\*/

#include "pizza.hpp"

#include <iostream>

namespace Demo02

{

Pizza::Pizza(std::string size, std::string topping)

: size\_{ size }, toppings\_{ std::vector<std::string>{topping} }

{

std::cout << "\n\*\*\*Constructor\n";

}

Pizza::Pizza(const Pizza& other)

: size\_{other.size\_}, toppings\_{other.toppings\_}

{

std::cout << "\n\*\*\*Copy constructor\n";

}

Pizza::~Pizza()

{

std::cout << "\n\*\*\*Destructor\n";

}

void Pizza::add\_topping(std::string topping)

{

toppings\_.push\_back(topping);

}

void Pizza::remove\_topping(std::string topping)

{

//search for item

auto item = std::find(

toppings\_.begin(), //where to start searching

toppings\_.end(), //where to end the searching

topping); //what to search for

//delete the item

if (item != toppings\_.end())

{

toppings\_.erase(item);

}

}

std::ostream& operator<<(std::ostream& os, const Pizza& pizza)

{

std::string toppings;

for (std::string item : pizza.toppings\_)

{

toppings += item + ", ";

}

float price =

pizza.prices\_.at(pizza.size\_) //lookup the price for this size

+ pizza.toppings\_.size() \* 0.25; //each toppins if .25

os << pizza.size\_ << " toppings: " << toppings << " @" << price;

return os;

}

}