Philip Pesic

Week 17

December 11 2022

Week 17 Program

```
// Factory Pattern

// * Add code for a 4th pizza type:  Cheese

// * Modify the pizza_information function, so it also prints the pizza name

// * Comment each function - Explaining what it does

// * Explain why this is a factory pattern

//

//  main.cpp

//  Week 17 Prog

//

//  Created by Pippo Pesic on 12/9/22.

//


//This program is an example of the factory method because we create classes derived from
Pizza, and then create the objects in a function outside of the class. The main function will not
create objects, but rather passes the type of pizza to the create function in order to create the
function.


#include <stdexcept>
#include <iostream>
#include <memory>
```

Philip Pesic

Week 17

December 11 2022

Week 17 Program

```
#include <string>
using namespace std;

class Pizza {
public:
	virtual double getPrice() const = 0;
	virtual ~Pizza() { cout << "Destructor called" << endl; }
};

class PepperoniOlivePizza : public Pizza {
public:
	virtual double getPrice() const { return 8.50; };
	virtual ~PepperoniOlivePizza() {};
};

class DeluxeChickenPizza : public Pizza {
public:
	virtual double getPrice() const { return 10.50; };
	virtual ~DeluxeChickenPizza() {};
};
```

Philip Pesic

Week 17

December 11 2022

Week 17 Program

```
class HawaiianPizza : public Pizza {
public:
	virtual double getPrice() const { return 11.50; };
	virtual ~HawaiianPizza() {};
};


class CheesePizza : public Pizza {
public:
	virtual double getPrice() const { return 7.50; };
	virtual ~CheesePizza() {};
};


class PizzaFactory {
public:
	enum PizzaType {
	PepperoniOlive,
	DeluxeChicken,
	Hawaiian,
	Cheese
	};
```

Philip Pesic

Week 17

December 11 2022

Week 17 Program

```
//Creates a new pizza

        static unique_ptr<Pizza> createPizza(PizzaType pizzaType) {

        switch (pizzaType) {

        case PepperoniOlive:  return make_unique<PepperoniOlivePizza>();

        case DeluxeChicken:  return make_unique<DeluxeChickenPizza>();

        case Hawaiian:  return make_unique<HawaiianPizza>();

        case Cheese:   return make_unique<CheesePizza>();

        }

        throw "invalid pizza type.";

        }

};


// Instantiate all available pizzas and print their prices, and names

void pizza_information(PizzaFactory::PizzaType pizzatype)

{

        string PizzaNames[4] = {"PepperoniOlive", "DeluxeChicken", "Hawaiian", "Cheese"};


        unique_ptr<Pizza> pizza = PizzaFactory::createPizza(pizzatype);

        cout << "Price of " << PizzaNames[pizzatype] << " is " << pizza->getPrice() <<

std::endl;
```

Philip Pesic

Week 17

December 11 2022

Week 17 Program

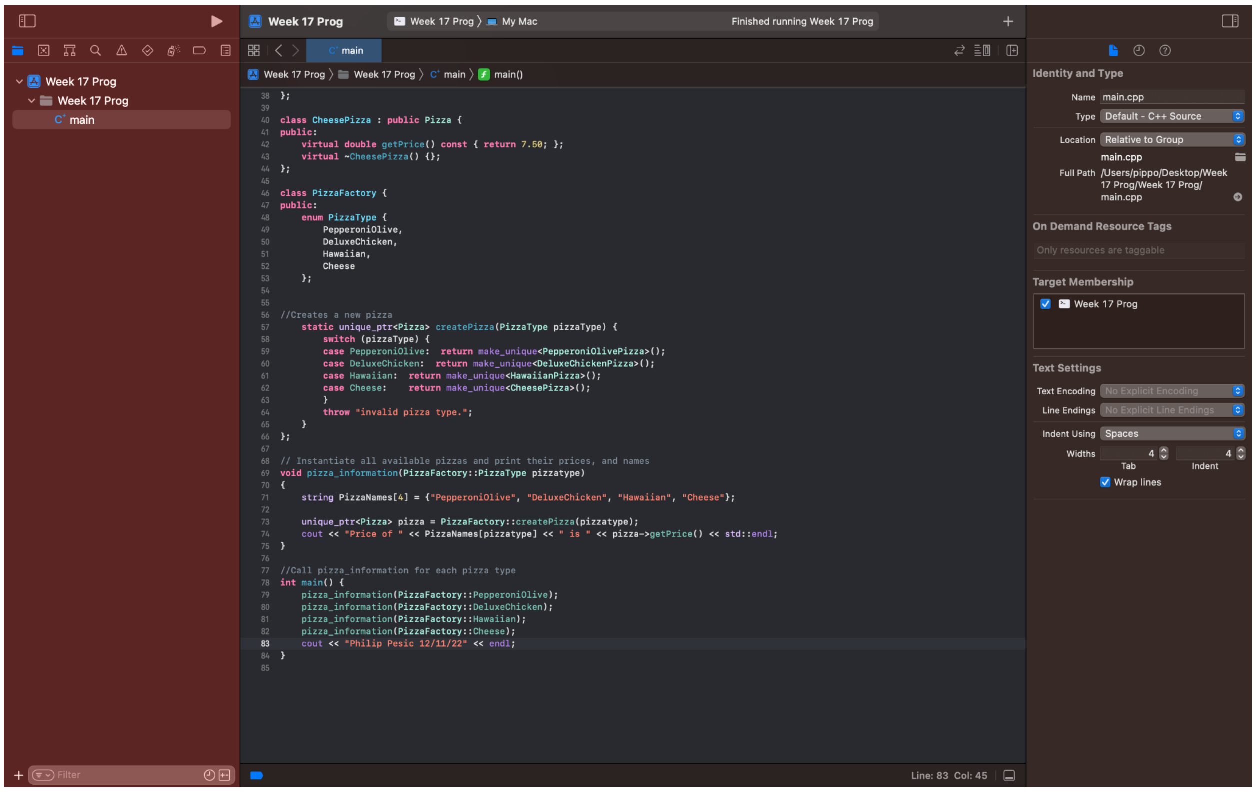
```
}


//Call pizza_information for each pizza type
int main() {
	pizza_information(PizzaFactory::PepperoniOlive);
	pizza_information(PizzaFactory::DeluxeChicken);
	pizza_information(PizzaFactory::Hawaiian);
	pizza_information(PizzaFactory::Cheese);
	cout << "Philip Pesic 12/11/22" << endl;
}
```

Philip Pesic

Week 17

December 11 2022

Week 17 Program



```cpp
};

class CheesePizza : public Pizza {
public:
    virtual double getPrice() const { return 7.50; };
    virtual ~CheesePizza() {};
};

class PizzaFactory {
public:
    enum PizzaType {
        PepperoniOlive,
        DeluxeChicken,
        Hawaiian,
        Cheese
    };


//Creates a new pizza
    static unique_ptr<Pizza> createPizza(PizzaType pizzaType) {
        switch (pizzaType) {
        case PepperoniOlive:  return make_unique<PepperoniOlivePizza>();
        case DeluxeChicken:  return make_unique<DeluxeChickenPizza>();
        case Hawaiian:  return make_unique<HawaiianPizza>();
        case Cheese:    return make_unique<CheesePizza>();
        }
        throw "invalid pizza type.";
    }
};

// Instantiate all available pizzas and print their prices, and names
void pizza_information(PizzaFactory::PizzaType pizzatype)
{
    string PizzaNames[4] = {"PepperoniOlive", "DeluxeChicken", "Hawaiian", "Cheese"};

    unique_ptr<Pizza> pizza = PizzaFactory::createPizza(pizzatype);
    cout << "Price of " << PizzaNames[pizzatype] << " is " << pizza->getPrice() << std::endl;
}

//Call pizza_information for each pizza type
int main() {
    pizza_information(PizzaFactory::PepperoniOlive);
    pizza_information(PizzaFactory::DeluxeChicken);
    pizza_information(PizzaFactory::Hawaiian);
    pizza_information(PizzaFactory::Cheese);
    cout << "Philip Pesic 12/11/22" << endl;
}

```

Week 17 Prog
Week 17 Prog > My Mac
Finished running Week 17 Prog
main
Week 17 Prog > Week 17 Prog > main > main()
Line: 83 Col: 45

Identity and Type
Name: main.cpp
Type: Default - C++ Source
Location: Relative to Group
main.cpp
Full Path: /Users/pippo/Desktop/Week 17 Prog/Week 17 Prog/main.cpp

On Demand Resource Tags
Only resources are taggable

Target Membership
✓ Week 17 Prog

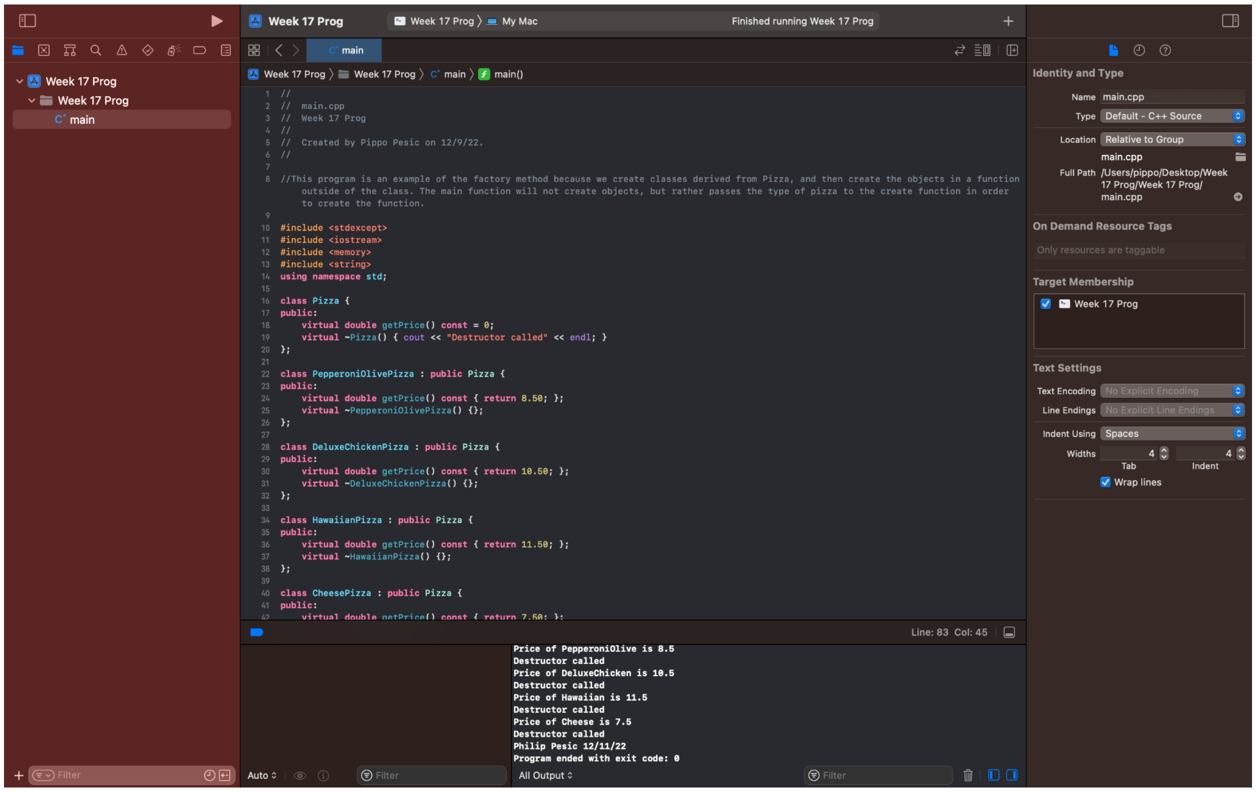
Text Settings
Text Encoding: No Explicit Encoding
Line Endings: No Explicit Line Endings
Indent Using: Spaces
Widths: 4 Tab, 4 Indent
✓ Wrap lines

Philip Pesic

Week 17

December 11 2022

Week 17 Program



Week 17 Prog 〉 My Mac — Finished running Week 17 Prog

Week 17 Prog 〉 Week 17 Prog 〉 main 〉 main()

```cpp
//
//  main.cpp
//  Week 17 Prog
//
//  Created by Pippo Pesic on 12/9/22.
//

//This program is an example of the factory method because we create classes derived from Pizza, and then create the objects in a function
    outside of the class. The main function will not create objects, but rather passes the type of pizza to the create function in order
    to create the function.

#include <stdexcept>
#include <iostream>
#include <memory>
#include <string>
using namespace std;

class Pizza {
public:
    virtual double getPrice() const = 0;
    virtual ~Pizza() { cout << "Destructor called" << endl; }
};

class PepperoniOlivePizza : public Pizza {
public:
    virtual double getPrice() const { return 8.50; };
    virtual ~PepperoniOlivePizza() {};
};

class DeluxeChickenPizza : public Pizza {
public:
    virtual double getPrice() const { return 10.50; };
    virtual ~DeluxeChickenPizza() {};
};

class HawaiianPizza : public Pizza {
public:
    virtual double getPrice() const { return 11.50; };
    virtual ~HawaiianPizza() {};
};

class CheesePizza : public Pizza {
public:
    virtual double getPrice() const { return 7.50; };
```

Line: 83 Col: 45

```
Price of PepperoniOlive is 8.5
Destructor called
Price of DeluxeChicken is 10.5
Destructor called
Price of Hawaiian is 11.5
Destructor called
Price of Cheese is 7.5
Destructor called
Philip Pesic 12/11/22
Program ended with exit code: 0
```

Identity and Type
Name: main.cpp
Type: Default - C++ Source
Location: Relative to Group
main.cpp
Full Path: /Users/pippo/Desktop/Week 17 Prog/Week 17 Prog/main.cpp

On Demand Resource Tags
Only resources are taggable

Target Membership
[✓] Week 17 Prog

Text Settings
Text Encoding: No Explicit Encoding
Line Endings: No Explicit Line Endings
Indent Using: Spaces
Widths: Tab 4, Indent 4
[✓] Wrap lines

I learned: how to use and implement the factory pattern