Диаграмма классов:

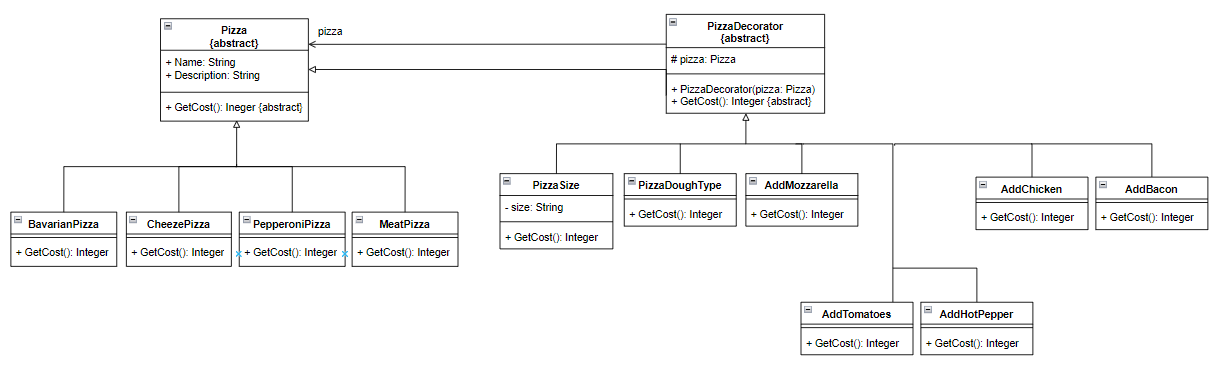
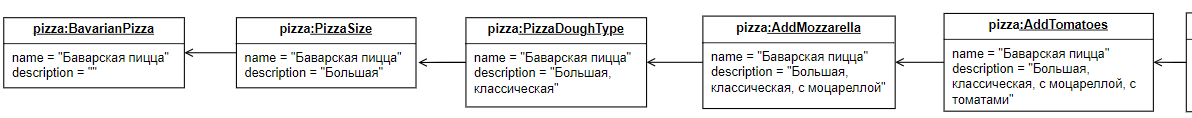
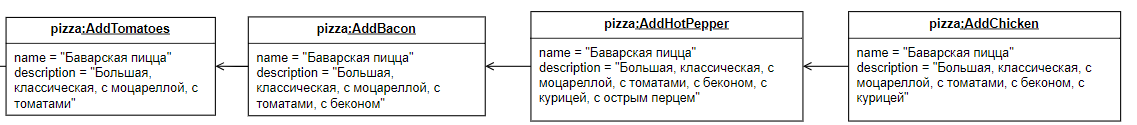
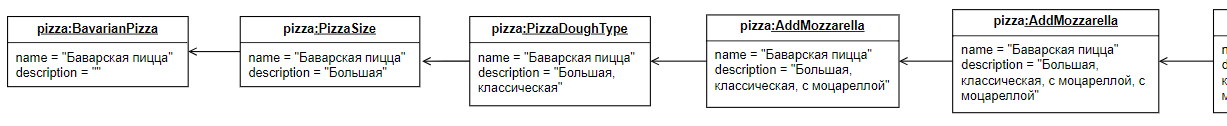
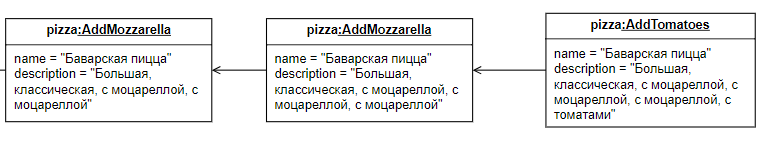


Диаграмма объектов:









Паттерн применяется при добавлении ингредиентов к пицце, и помогает избежать создания огромного количества классов под каждый тип пиццы: отдельные классы под каждую «чистую» пиццу (без доп. ингредиентов), отдельные под пиццу с добавлением только сыра, только помидоров, их различные комбинации и т. д.

Декоратор динамически определяет количество добавок и повышение стоимости пиццы в зависимости от них.

Поэтому код значительно упрощается:

// В файле MainForm.cs

public Pizza pizza;

// В файле, отвечающем за добавление ингредиентов

mainForm.pizza = new PizzaSize(sizeComboBox.Text, mainForm.pizza);

mainForm.pizza = new PizzaDoughType(doughTypeComboBox.Text, mainForm.pizza);

if (mozzarellaCheckBox.Checked) mainForm.pizza = new AddMozzarella(mainForm.pizza);

if (baconCheckBox.Checked) mainForm.pizza = new AddBacon(mainForm.pizza);

if (chickenCheckBox.Checked) mainForm.pizza = new AddChicken(mainForm.pizza);

if (tomatoesСheckBox.Checked) mainForm.pizza = new AddTomatoes(mainForm.pizza);

if (hotPepperCheckBox.Checked) mainForm.pizza = new AddHotPepper(mainForm.pizza);

Без паттерна тут могла быть куча ифов, которые бы проверяли, какой размер у пиццы и какие у неё ингридиеты, чтобы создать экземпляр нужного класса.

Пример кода без паттерна:

// В файле Pizzas.cs

public abstract class Pizza

{

public string Name'

public string Description;

public abstract int GetCost();

}

public class BigClassicBavarianPizza : Pizza

{

public BavarianPizza()

{

Name = "Баварская пицца";

Description = "Большая, классическая";

}

public override int GetCost()

{

return 900;

}

}

public class BigClassicBavarianPizzaWithTomatoes : Pizza

{

public BavarianPizza()

{

Name = "Баварская пицца";

Description = "Большая, классическая, с помидорами";

}

public override int GetCost()

{

return 960;

}

}

public class BigClassicBavarianPizzaWithTomatoesAndBacon : Pizza

{

public BavarianPizza()

{

Name = "Баварская пицца";

Description = "Большая, классическая, с помидорами, с беконом";

}

public override int GetCost()

{

return 1020;

}

}

// И т.д. все возможные комбинации пицц

// В файле AdditiveForm.cs

private void button1\_Click(object sender, EventArgs e)

{

mainForm.pizza = new PizzaSize(sizeComboBox.Text, mainForm.pizza);

mainForm.pizza = new PizzaDoughType(doughTypeComboBox.Text, mainForm.pizza);

if (sizeComboBox.Text == "Большая")

{

if (doughTypeComboBox.Text == "Классическая")

{

if (tomatoesСheckBox.Checked)

{

if (baconCheckBox.Checked)

{

if (/\*......\*/)

{

//

// ..........

//

}

else mainForm.pizza = new BigClassicBavarianPizzaWithTomatoesAndBacon();

}

//

// ..........

//

else mainForm.pizza = new BigClassicBavarianPizzaWithTomatoes();

}

if (/\*......\*/)

{

//

// ..........

//

}

//

// ..........

//

else mainForm.pizza = new BigClassicBavarianPizza();

}

else

{

//

// ..........

//

}

}

if (sizeComboBox.Text == "Средняя")

{

//

// ..........

//

}

else

{

//

// ..........

//

}

mainForm.chequeGridView.Rows.Add(mainForm.pizza.Name, mainForm.pizza.GetCost(), mainForm.pizza.Description);

this.Close();

}