OU2 - Design Patterns

HT2023

2023-12-11

1. Introduction

This assignment is composed of four smaller assignments in which the task is to create programs or implement functions using different types of design patterns. All of these programs are created and compiled in Java 17.

Theodor Jonsson
ens18trn@cs.umu.se

2. Builder

The Builder design pattern is a creational design pattern that allows you to construct objects step by step. It allows you to produce different types of an object using the same code. The builder design pattern is also useful when an object has multiple attributes that aren't always needed, this is similar to the Factory pattern but in Factory you are forced to send all of the parameters and the optional ones you don't want need to be sent as null.

This task is to create a burger with the attributes meatType, a list of vegetables and a list of sauces and a cost. We want to be able to instantiate it with any combination of meat and toppings without deciding it directly at the time of creation.

To accomplish this an abstract class called Burger is created which the three different types of burgers will inherit from. Then creating a BurgerBuilder class which has the methods to specify meattype and adding vegetables and sauces and calculating the cost for these. Then when the customer is satisfied with its order the user can call the build method in the BurgerBuilder class which will return the created Burger. This means the user only has to interact with the BurgerBuilder and the abstract class of Burger. Figure 1 shows the UML-diagram for the program showing that the user doesn't need to know the different types of burgers, they only need to input the ingredients to the BurgerBuilder and after building getting back the desired burger.

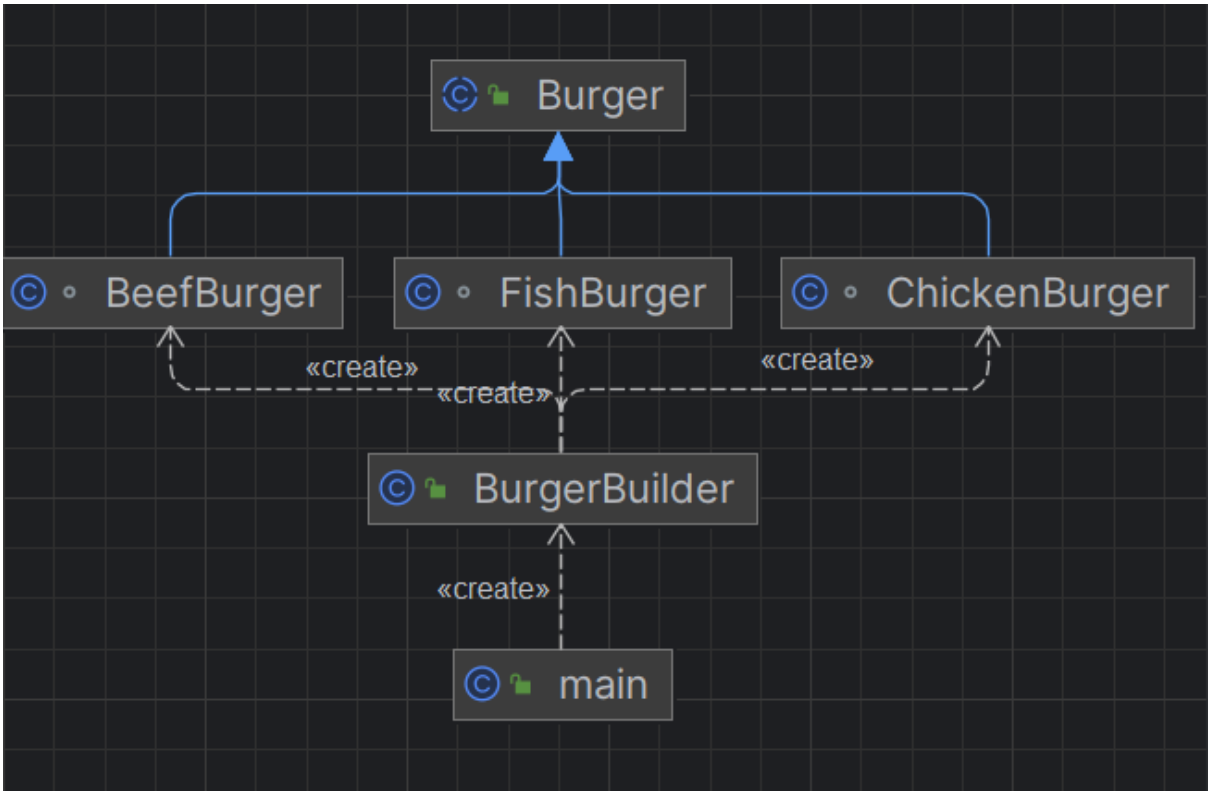

Figure 1. UML-diagram over the Burger Builder program

3. Abstract Factory

Abstract Factory is a creational design pattern that is a sort of factory of factories; it provides an interface for creating families of dependent objects without having to specify their concrete class. Abstract Factory makes it easy to change the type of factory that will actually create the object without breaking the user code.

The task is to expand the burger business and create a menu for the three most common combinations of burgers which only takes one button to create. These items are supposed to be easily swappable.

To achieve this a BurgerFactory interface is created which only has one method which is createBurger. Three different concrete factories which implement the BurgerFactory interface these three factories all use the BurgerBuilder and has preset ingredients to add the builds the burger which in turn then returns the burgers

4. Memento

The Memento Design pattern is a behavioral design pattern which offers a way to implement undoing actions by saving the state of an object and returning the object to the saved state if need be.

This task was to create a text editor that is able to create, edit and remove notes as well as being able to go back to previous states of the notes. When the user types in the text editor the JETextArea will notify all listeners of updates in this case it is the Note class. The Note class will store the text in the JETextArea as a String and the cursor positions as integers; these values are what the state of the text editor is .

There are six buttons that the user can use. New Note creates a new note and saves the one user was on. Save saves the current state of the text editor for when the User uses Back which returns the text editor to the last saved state. Then Select which moves the user to the selected note from the list of notes and Removes deletes the selected note. The last button Show All will show a list of every saved state of that note in a pop up window.

5. Iterator

The iterator design pattern is a design pattern which lets you traverse a collection of elements without exposing the underlying representation. In this task the underlying representation is a doubly linked list.

Here we are tasked to implement the Iterable interface in the linked list to accomplish this we have to create a method that returns an iterator. This iterator in this task is a private inner class that implements the iterator interface and overrides the default methods called next which returns the next element and hasNext which returns true if there is another element false if there isn't. These methods are used in the default implementation of the foreach loop

6. Notes

The code might be sort of strange. It is from my own submission for last year's OU2. I asked Johan if I should hand in the assignment according to this year's specification or last year and I was told to do it according to this year's requirements to make it easier for grading. So not wanting to throw everything i did last year in the trash, I repurposed what i could which means i might have brought in things i am not allowed to.