|  |  |
| --- | --- |
| **Assignment Case** |  |
| COMP6122 | COMP6122001  Framework Layer Architecture |
| **Computer Science** | **Example Case** |
| ***Valid on*** *-* | **Revision 00** |

## Soal

*Case*

**NomNom Co.**

**Criteria**:

1. **Factory Method**

First, you need **factory method pattern** to define an interface or abstract class for creating an object but let the subclassed decide which class to instantiate. In other words, subclasses are responsible to create the instance of the class.

1. **Singleton**

Next, you need to create **singleton pattern** to restrict the instantiation of an object, so the user does not instantiate more than 1 object while the program runs.

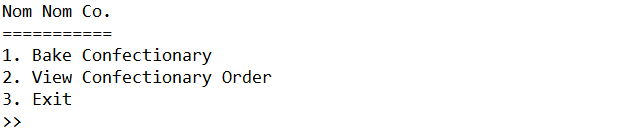
1. **Adapter**

Last, you need the **adapter pattern** to convert an interface to another interface. This pattern lets different classes from different interfaces to work together without conflict.

**NomNom Co.** is a small confectionary shop which recently got famous because some well-known people love the confectionary very much. Because of that, its order got sky-rocketed and renders the company fails to keep up the order manually. Fortunately, the Co-Founder of NomNom Co.reached out to you and ask you to build a system that could handle all of the order. You need to build a program based on the following criteria:

1. **Menu**

When the system runs, display the menu to user. User can choose whether they want to **bake confectionary**, **view confectionary order**, or **exit** the program.



*Figure 1Menu*

1. **Bake Confectionary**

If user chooses to **menu 1** (**Bake Confectionary**), prompt user for the confectionary detail with following criteria:

* + - First, prompt user to input **confectionary type**. Validate the input is **case sensitive** and can only be filled with **either** “**Cupcake**” or “**Tart**”.



*Figure 2Confectionary Type Input*

* + - Next, prompt user to input **confectionary name**. Validate the name length **must be** **between 5 and 15** (**inclusive**).



*Figure 3Confectionary Name Input*

* + - Next, prompt user to input **confectionary softness**. Validate the input is **case sensitive** and can only be filled with **either** “**Fluffy**”, “**Medium**”, or “**Hard**”.



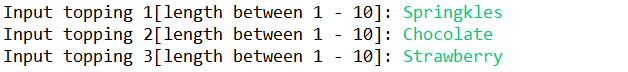
*Figure 4Confectionary Softness Input*

* + - After that, prompt user to input info regarding **topping**.Validate the input is **case sensitive** and can only be filled with **either** “**Y**” or “**N**”.



*Figure 5Adding Topping Input*

* If the input is “**Y**”, prompt user to input **3 topping**. Validate each input length **must be between 1 and 10** (**inclusive**).



*Figure 6Topping Input*

* **Otherwise**, proceed to the **price** input.
  + - Then, prompt user to input **confectionary price**. Validate the price must be **between** **10.0 and 50.0** (**inclusive**).



*Figure 7 Confectionary Price Input*

* + - After that, prompt user to input **payment type**. Validate the input is **case sensitive** and can only be filled with **either** “**Cash**”, ” **Transfer**”, and “**Crypto**”.



*Figure 8Payment Type Input*

* + - After that, **convert** **the price** based on the following formula

|  |  |  |
| --- | --- | --- |
| **Cash** | **Transfer** | **Crypto** |
| Price \* 1 | Price \* 1.1 | Price / 2 |

* + - Lastly, **display message** clarifying that the confectionary has been made. Then, bake the requested confectionary and **add the confectionary** to the **database**. Ensure that **th**ere can only be **1 database instance** created in this program.

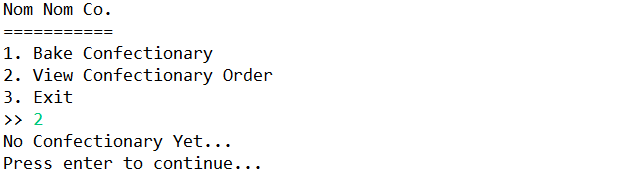


*Figure 9Success Message*

1. **View Arrangement Package**

If user choose **menu 2** (**View Confectionary Order**), **display data** from **database** based on the following criteria:

* + - If the database is **empty**, then **inform** user with **a message**.



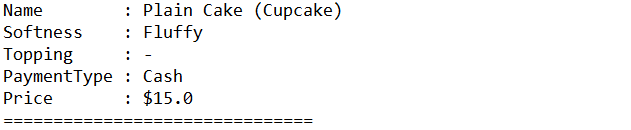
*Figure 10Empty Database*

* + - **Otherwise**, **display** **all baked confectionary** from the database with the required information regarding:
* **Confectionary Name** and **Confectionary Type**
* **Confectionary Softness**
* **Topping**
* If there’s **no additional topping**, **display** “**-**” for the topping section.
* **Otherwise**, display **all** **topping**.
* **Price**
* If the price is in **transfer** format, generate a **random account number** with the **length of 10** based on the following formula:

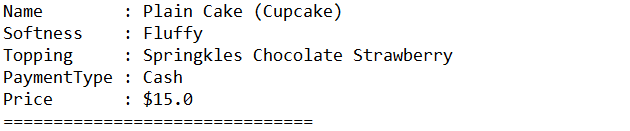
|  |
| --- |
| **XXXXXXXXXX |** X can be **filled with [0-9]** |

* If the price is in **crypto** format, generate a **random crypto address** with the **length of 12** based on the following formula:

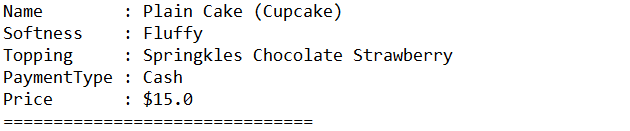
|  |
| --- |
| **0xXXXXXXXXXX** **|** X can be **filled with [a-z0-9]** |



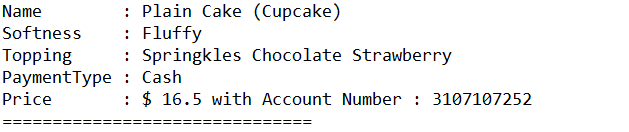
*Figure 11Confectionary without Topping View*



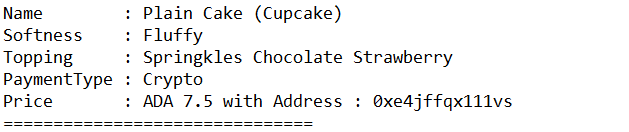
*Figure 12Confectionary with Topping View*



*Figure 13View*



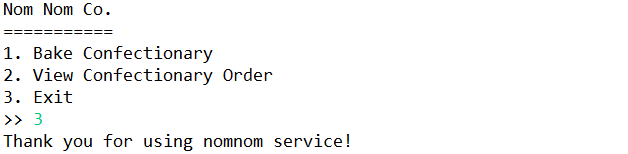
*Figure 14View*



*Figure 15View*

1. **Exit**

If user chooses **menu 3** (**Exit**), then show a **goodbye message** and **exit** the program.



*Figure 16*