VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

****

**THE ESSAY OF DESIGN PATTERN**

**DEALING WITH GENERALIZATION**

*Instructor*: **NGUYEN THANH PHUOC**

*Implementer*: **NGUYEN TRUNG TIN – 520H0589**

**LE MINH NHAT – 520H0560**

*Class***: 20H50202**

*Course***: 24**

**HO CHI MINH CITY, 2023**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

****

**ESSAY OF DESIGN PATTERN**

**DEALING WITH GENERALIZATION**

Instructor: **NGUYEN THANH PHUOC**

Implementer: **NGUYEN TRUNG TIN – 520H0589**

**LE MINH NHAT – 520H0560**

Class**: 20H50202**

Course**: 24**

**HO CHI MINH CITY, 2023**

ACKNOWLEDGEMENT

First of all, I would like to thank Ton Duc Thang University for Software Engineering in the curriculum. In particular, I would like to express my deep and sincere gratitude to Mr. **Nguyen Thanh Phuoc** who wholeheartedly guided and guided me during the study and writing of this essay. Thank you for your kind words and enthusiasm, which helped me to complete my thesis.

Thank you to my friends who have accompanied and encouraged me throughout the research process.

Thank you very much.

**PROJECT COMPLETED**

**AT TON ĐUC THANG UNIVERSITY**

I hereby declare that this is our own project and is guided by Mr. **Nguyen Thanh Phuoc**. The research contents and results on this topic are honest and have not been published in any form before. The data in the tables for analysis, comments, and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

In addition, the project also uses a number of comments, assessments as well as data from other authors, other agencies, and organizations, with citations and source annotations.

**If I find any fraud, I will take full responsibility for the content of my project.** Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*Ho Chi Minh City, April 1st 2023*

*Author*

*Nguyen Trung Tin*

*Le Minh Nhat*

TEACHER'S CONFIRMATION AND ASSESSMENT SECTION

**TEACHER'S CONFIRMATION**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh City, April 1st 2023*

(sign and write name)

**TEACHER'S ASSESSMENT**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh City, April 1st 2023*

(sign and write name)

TABLE OF CONTENT

[ACKNOWLEDGEMENT 1](#_Toc29296)

[TEACHER'S CONFIRMATION AND ASSESSMENT SECTION 3](#_Toc22661)

[TABLE OF CONTENT 4](#_Toc32633)

[LIST OF FIGURES, TABLES 9](#_Toc820)

[CONTRIBUTION 10](#_Toc16816)

[SUMMARY 11](#_Toc20196)

[CHAPTER 1: PULL UP FIELD 12](#_Toc7235)

[1.1 Problem 12](#_Toc10341)

[1.2 Solution 12](#_Toc28082)

[1.3 Example 12](#_Toc19808)

[1.3.1 Before 12](#_Toc6532)

[1.3.2 After 13](#_Toc17854)

[1.4 Demo 13](#_Toc22677)

[1.4.1 Source Code Java 13](#_Toc15663)

[1.4.2 Source code C# 13](#_Toc19695)

[CHAPTER 2: PULL UP METHOD 13](#_Toc26163)

[2.1 Problem 13](#_Toc32075)

[2.2 Solution 13](#_Toc6666)

[2.3 Example 14](#_Toc15783)

[2.3.1 Before 14](#_Toc17778)

[2.3.2 After 14](#_Toc30109)

[2.4 Demo 15](#_Toc28710)

[2.4.1 Source Code Java 15](#_Toc31324)

[2.4.2 Source code C# 15](#_Toc23924)

[CHAPTER 3: PULL UP CONSTRUCTOR BODY 15](#_Toc8559)

[3.1 Problem 15](#_Toc30310)

[3.2 Solution 15](#_Toc6839)

[3.3 Example 16](#_Toc29266)

[3.3.1 Before 16](#_Toc21318)

[3.3.2 After 16](#_Toc6351)

[3.4 Demo 16](#_Toc28665)

[3.4.1 Source Code Java 16](#_Toc4187)

[3.4.2 Source code C# 17](#_Toc9735)

[CHAPTER 4: PUSH DOWN METHOD 17](#_Toc1852)

[4.1 Problem 17](#_Toc10463)

[4.2 Solution 17](#_Toc12387)

[4.3 Example 17](#_Toc24578)

[4.3.1 Before 17](#_Toc7251)

[4.3.2 After 18](#_Toc27848)

[4.4 Demo 18](#_Toc30301)

[4.4.1 Source Code Java 18](#_Toc1838)

[4.4.2 Source code C# 19](#_Toc12471)

[CHAPTER 5: PUSH DOWN FIELD 19](#_Toc12825)

[5.1 Problem 19](#_Toc15917)

[5.2 Solution 19](#_Toc2742)

[5.3 Example 19](#_Toc11903)

[5.3.1 Before 19](#_Toc21473)

[5.3.2 After 20](#_Toc27319)

[5.4 Demo 20](#_Toc9653)

[5.4.1 Source Code Java 20](#_Toc30952)

[5.4.2 Source code C# 20](#_Toc10937)

[CHAPTER 6: EXTRACT SUBCLASS 20](#_Toc11068)

[6.1 Problem 20](#_Toc24382)

[6.2 Solution 20](#_Toc31814)

[6.3 Example 21](#_Toc11503)

[6.3.1 Before 21](#_Toc10982)

[6.3.2 After 21](#_Toc19360)

[6.4 Demo 22](#_Toc31297)

[6.4.1 Source Code Java 22](#_Toc23011)

[6.4.2 Source code C# 22](#_Toc24225)

[CHAPTER 7: EXTRACT SUPERCLASS 22](#_Toc29416)

[7.1 Problem 22](#_Toc26173)

[7.2 Solution 22](#_Toc29872)

[7.3 Example 23](#_Toc25901)

[7.3.1 Before 23](#_Toc167)

[7.3.2 After 23](#_Toc26138)

[7.4 Demo 23](#_Toc21007)

[7.4.1 Source Code Java 23](#_Toc11633)

[7.4.2 Source code C# 24](#_Toc8437)

[CHAPTER 8: EXTRACT INTERFACE 24](#_Toc19230)

[8.1 Problem 24](#_Toc15737)

[8.2 Solution 24](#_Toc22291)

[8.3 Example 24](#_Toc11957)

[8.3.1 Before 24](#_Toc5935)

[8.3.2 After 24](#_Toc14732)

[8.4 Demo 25](#_Toc325)

[8.4.1 Source Code Java 25](#_Toc5915)

[8.4.2 Source code C# 25](#_Toc25158)

[CHAPTER 9: COLLAPSE HIERARCHY 25](#_Toc8112)

[9.1 Problem 25](#_Toc31714)

[9.2 Solution 25](#_Toc5740)

[9.3 Example 26](#_Toc13344)

[9.3.1 Before 26](#_Toc25338)

[9.3.2 After 26](#_Toc3769)

[9.4 Demo 26](#_Toc12805)

[9.4.1 Source Code Java 26](#_Toc9783)

[9.4.2 Source code C# 27](#_Toc31313)

[CHAPTER 10: FORM TEMPLATE METHOD 27](#_Toc19241)

[10.1 Problem 27](#_Toc5023)

[10.2 Solution 27](#_Toc25072)

[10.3 Example 27](#_Toc13775)

[10.3.1 Before 27](#_Toc7150)

[10.3.2 After 28](#_Toc10705)

[10.4 Demo 28](#_Toc20246)

[10.4.1 Source Code Java 28](#_Toc873)

[10.4.2 Source code C# 28](#_Toc1850)

[CHAPTER 11: REPLACE INHERITANCE WITH DELEGATION 29](#_Toc12758)

[11.1 Problem 29](#_Toc1671)

[11.2 Solution 29](#_Toc26323)

[11.3 Example 29](#_Toc22279)

[11.3.1 Before 29](#_Toc3729)

[11.3.2 After 29](#_Toc23853)

[11.4 Demo 30](#_Toc10398)

[11.4.1 Source Code Java 30](#_Toc7084)

[11.4.2 Source code C# 30](#_Toc4302)

[CHAPTER 12: REPLACE DELEGATION WITH INHERITANCE 30](#_Toc6960)

[12.1 Problem 30](#_Toc164)

[12.2 Solution 30](#_Toc15535)

[12.3 Example 31](#_Toc5323)

[12.3.1 Before 31](#_Toc13162)

[12.3.2 After 31](#_Toc16293)

[12.4 Demo 31](#_Toc3035)

[12.4.1 Source Code Java 31](#_Toc6391)

[12.4.2 Source code C# 31](#_Toc3868)

[REFERENCE 32](#_Toc15410)

LIST OF FIGURES, TABLES

**LIST OF FIGURES**

[Figure 1 : Before use Pull Up Field example 12](#_Toc12793)

[Figure 2 : After use Pull Up Field example 13](#_Toc1738)

[Figure 3 : Before use Pull Up Method 14](#_Toc32756)

[Figure 4 : After use Pull Up Method 15](#_Toc10564)

[Figure 5 : Before use Pull Up Constructor Body 16](#_Toc2314)

[Figure 6 : After use Pull Up Constructor Body 16](#_Toc30060)

[Figure 7 : Before use Push Down Method 18](#_Toc7455)

[Figure 8 : After use Push Down Method 18](#_Toc9969)

[Figure 9 : Before use Push Down Method 19](#_Toc9951)

[Figure 10 : After use Push Down Field 20](#_Toc23679)

[Figure 11 : Before use Extract Subclass 21](#_Toc3586)

[Figure 12 : After use Extract Subclass 22](#_Toc11304)

[Figure 13 : Before use Extract Superclass 23](#_Toc3790)

[Figure 14 : After use Extract Superclass 23](#_Toc22280)

[Figure 15 : Before use Extract Interface 24](#_Toc10974)

[Figure 16 : After use Extract Interface 25](#_Toc24617)

[Figure 17 : Before use Collapse Hierachy 26](#_Toc5923)

[Figure 18 : After use Collapse Hierachy 26](#_Toc2724)

[Figure 19 : Before use Form Template Method 28](#_Toc19240)

[Figure 20 : After use Form Template Method 28](#_Toc777)

[Figure 21 : Before use Replace Inheritance With Delegation 29](#_Toc5530)

[Figure 22 : After use Replace Inheritance With Delegation 30](#_Toc14337)

[Figure 23 : Before use Replace Delegation With Inheritance 31](#_Toc23047)

[Figure 24 : After use Replace Delegation With Inheritance 31](#_Toc9778)

**LIST OF TABLES**

CONTRIBUTION

|  |  |  |
| --- | --- | --- |
| Member | Assignment | Completion |
| 1. 520H0560 - Le Minh Nhat | Code Java, C#, | 100% |
| 1. 520H0589 – Nguyen Trung Tin | Code Java, Word | 100% |

SUMMARY

In this essay, we learn about "Dealing with generalization" in Design Pattern. We have completed the report and demo in Java and C# . There is source code demo at git under the following 2 links:

- Source code Java:

<https://github.com/nhatlenlh/dealing-with-generalization.git>

- Source code C#:

<https://github.com/nhatlenlh/dealing-with-generalization.git>

CHAPTER 1: PULL UP FIELD

1.1 Problem

Subclasses and developed separately, causing identical or nearly identical fields and methods to appear.

1.2 Solution

- Make sure that uses of the candidate fields to ensure they are used in the same way.

- If the fields do not have the same name, rename the fields and replace all references to the fields in the existing code.

- Create a field with the same name in the superclass. If the fields were private, the superclass field should be protected.

- Remove the fields from the subclasses.

- Compile and test.

- You may use *Self Encapsulate Field* for the new field, and hide it behind access methods.

1.3 Example

1.3.1 Before

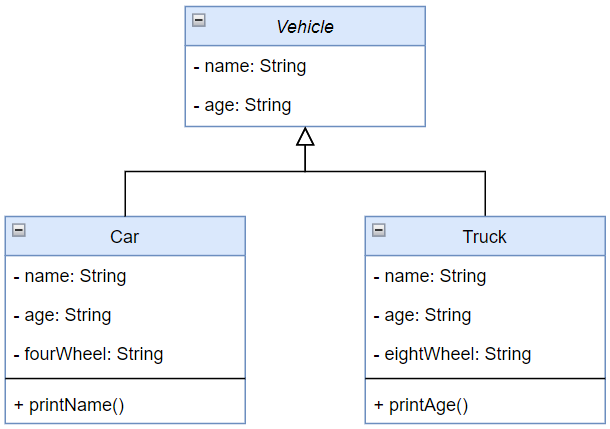


Figure 1: Before use Pull Up Field example

1.3.2 After

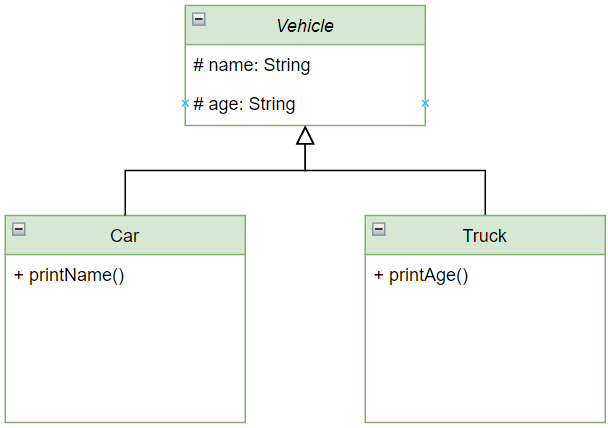


Figure 2: After use Pull Up Field example

1.4 Demo

1.4.1 Source Code Java

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/pullupfields>

**1.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/pullupfield>

**CHAPTER 2: PULL UP METHOD**

2.1 Problem

Subclasses have methods that perform similar work.

2.2 Solution

- Find similar methods in superclasses and format them to match each other.

- If methods use different parameters, put the parameters in the form that you want to see in the superclass.

- Copy the method to the superclass.

- In the superclass, the method code uses fields and methods that exist only in subclasses and therefore aren’t available in the superclass:

- For fields: use either *Pull Up Field* or *Self-Encapsulate Field* to create getters and setters in subclasses; then declare these getters abstractly in the superclass.

- For methods: use either Pull Up Method or declare abstract methods for them in the superclass.

- Remove the methods from the subclasses.

- Check the locations in which the method is called.

2.3 Example

2.3.1 Before

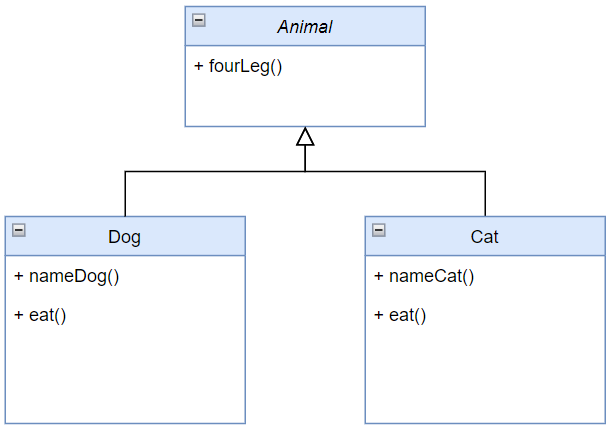


Figure 3: Before use Pull Up Method

2.3.2 After

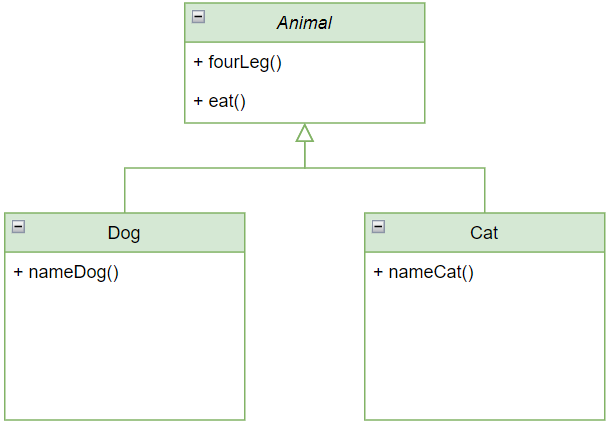


Figure 4: After use Pull Up Method

2.4 Demo

2.4.1 Source Code Java

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/pullupmethod>

**2.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/pullupmethod>

CHAPTER 3: PULL UP CONSTRUCTOR BODY

3.1 Problem

Subclasses have constructors with code that’s mostly identical.

3.2 Solution

- Create a constructor in a superclass.

- Extract the common code from the beginning of the constructor of each subclass to the superclass constructor.

- Place the call for the superclass constructor in the first line in the subclass constructors.

3.3 Example

3.3.1 Before

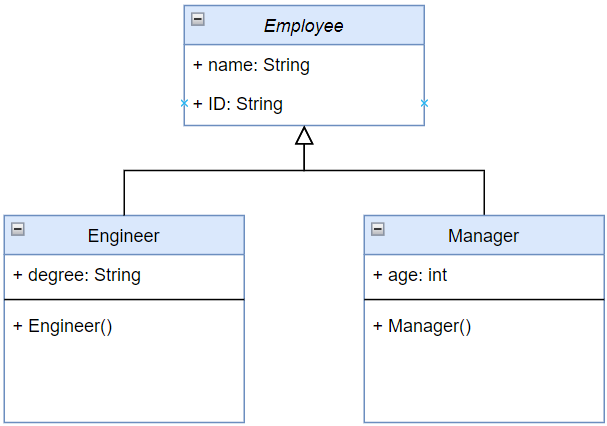


Figure 5: Before use Pull Up Constructor Body

3.3.2 After

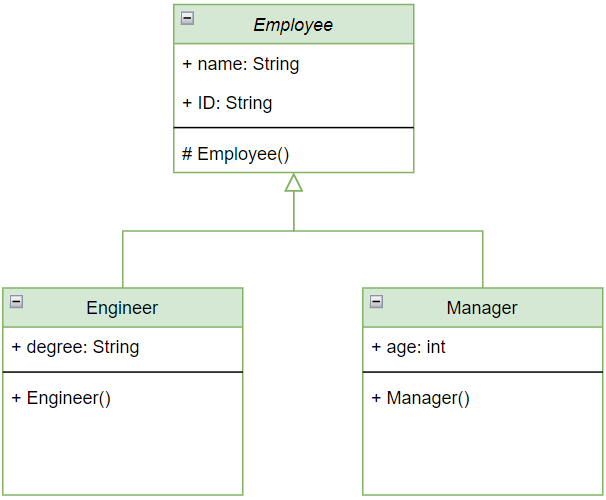


Figure 6: After use Pull Up Constructor Body

3.4 Demo

3.4.1 Source Code Java

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/pullupconstructorbody>

**3.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/pullupconstructorbody>

CHAPTER 4: PUSH DOWN METHOD

4.1 Problem

Behavior implemented in a superclass used by only one or a few subclasses

4.2 Solution

- Declare the method in a subclass and copy its code from the superclass.

- Remove the method from the superclass.

- Find all places where the method is used and verify that it’s called from the necessary subclass.

4.3 Example

4.3.1 Before

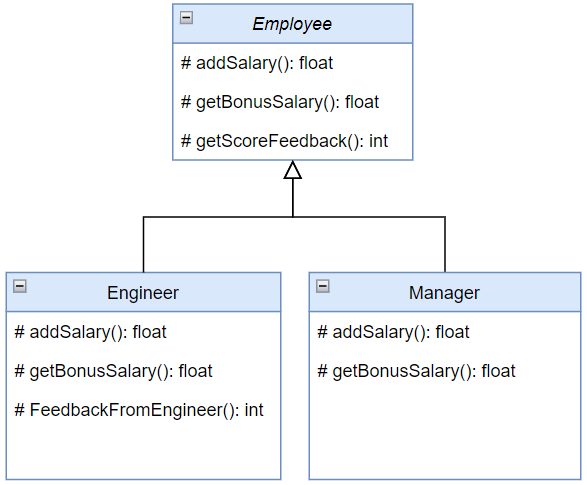


Figure 7: Before use Push Down Method

4.3.2 After

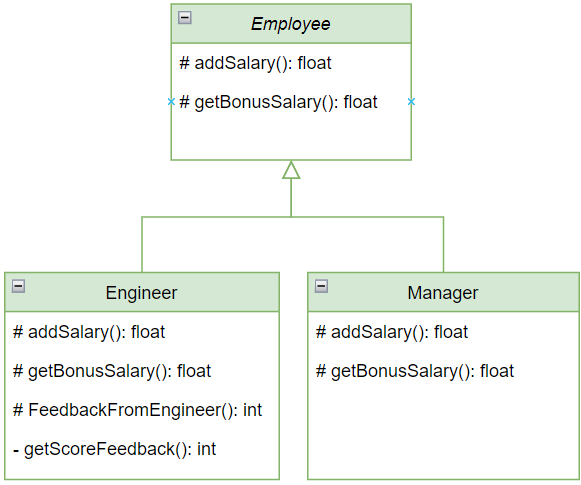


Figure 8: After use Push Down Method

**4.4 Demo**

**4.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/pushdownmethod>

**4.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/pushdownmethod>

CHAPTER 5: PUSH DOWN FIELD

5.1 Problem

A field used only in a few subclasses

5.2 Solution

- Declare a field in all the necessary subclasses.

- Remove the field from the superclass.

5.3 Example

5.3.1 Before

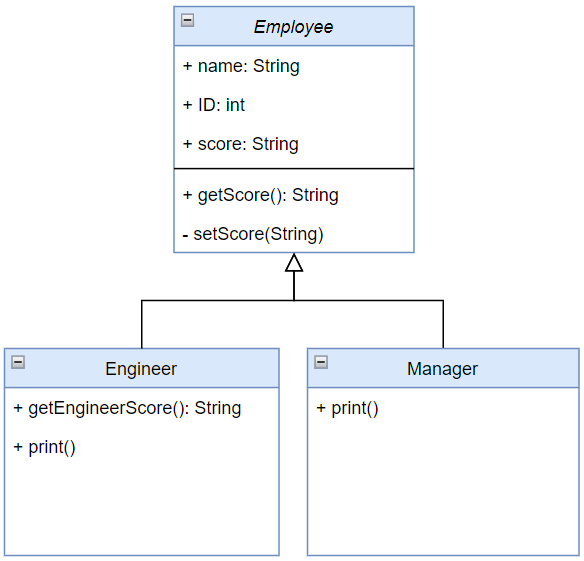


Figure 9: Before use Push Down Method

5.3.2 After

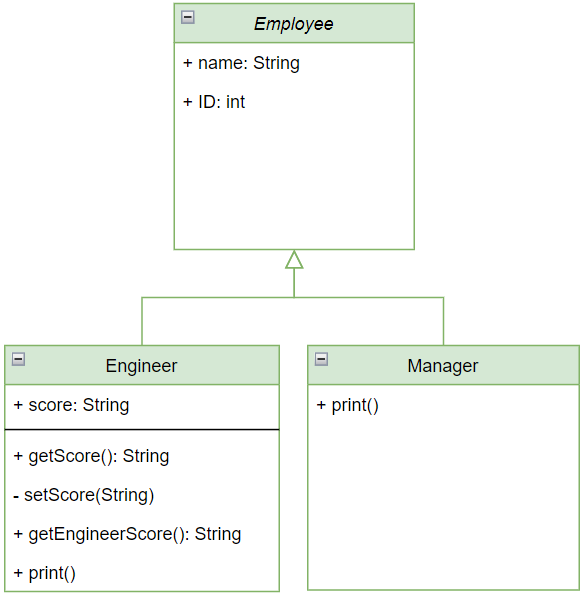


Figure 10: After use Push Down Field

**5.4 Demo**

**5.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/pushdownfield>

**5.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/pushdownfield>

CHAPTER 6: EXTRACT SUBCLASS

6.1 Problem

A class has features that are used only in certain cases.

6.2 Solution

- Create a new subclass.

- If you need additional data to create objects from a subclass, create a constructor and add the necessary parameters to it.

- Find all calls to the constructor of the parent class. When the functionality of a subclass is necessary, replace the parent constructor with the subclass constructor.

- Move the necessary methods and fields from the parent class to the subclass by *Push Down Method* and *Push Down Field*. The fields remain accessible throughout the whole process: from the parent class prior to the move, and from the subclass itself after the move is complete.

- Find and delete all the old fields that controlled the choice of functionality by using polymorphism to replace all the operators in which the fields had been used.

6.3 Example

6.3.1 Before

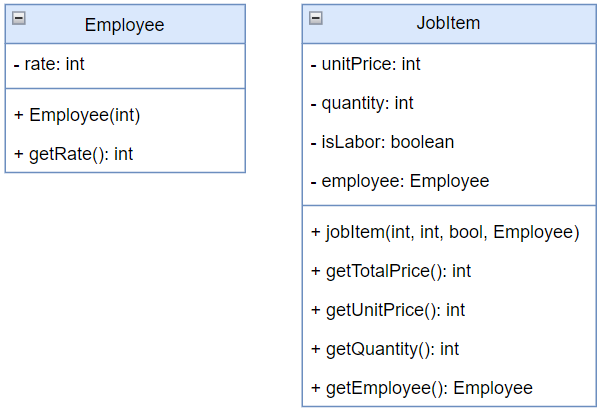


Figure 11: Before use Extract Subclass

6.3.2 After

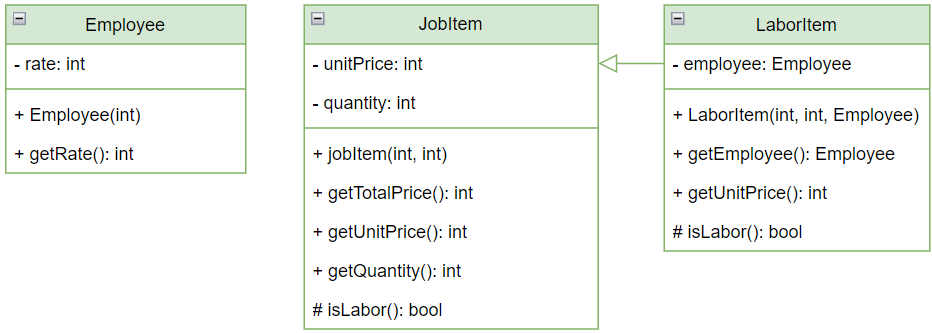


Figure 12: After use Extract Subclass

**6.4 Demo**

**6.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/extractsubclass>

**6.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/extractsubclass>

CHAPTER 7: EXTRACT SUPERCLASS

7.1 Problem

Two classes with common fields and methods.

7.2 Solution

- Create an abstract superclass.

- Use Pull Up Field, Pull Up Method, and Pull Up Constructor Body to move the common functionality to a superclass.

- Start with the fields, since in addition to the common fields you will need to move the fields that are used in the common methods.

- Look for places in the client code where use of subclasses can be replaced with your new class.

7.3 Example

7.3.1 Before

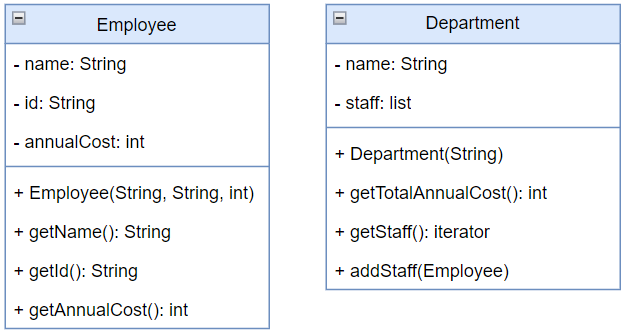


Figure 13: Before use Extract Superclass

7.3.2 After

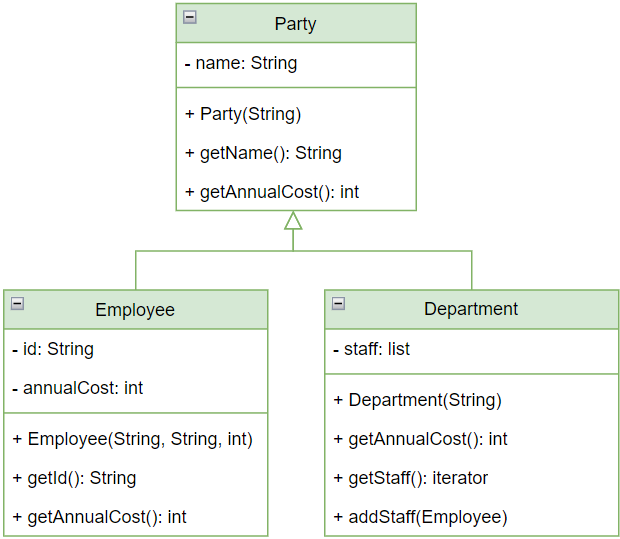


Figure 14: After use Extract Superclass

**7.4 Demo**

**7.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/extractsuperclass>

**7.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/extractsuperclass>

CHAPTER 8: EXTRACT INTERFACE

8.1 Problem

Clients are using the same part of a class interface or part of the interface is the same in two classes.

8.2 Solution

- Create an empty interface.

- Declare operations in the interface.

- Declare the necessary classes as implementing the interface.

- Change type declarations in the client code to use the new interface.

8.3 Example

8.3.1 Before

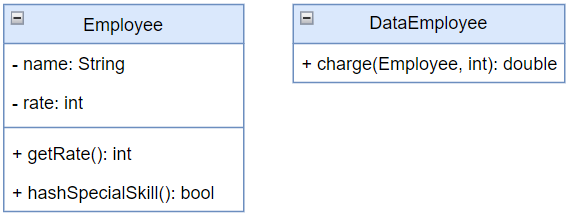


Figure 15: Before use Extract Interface

8.3.2 After

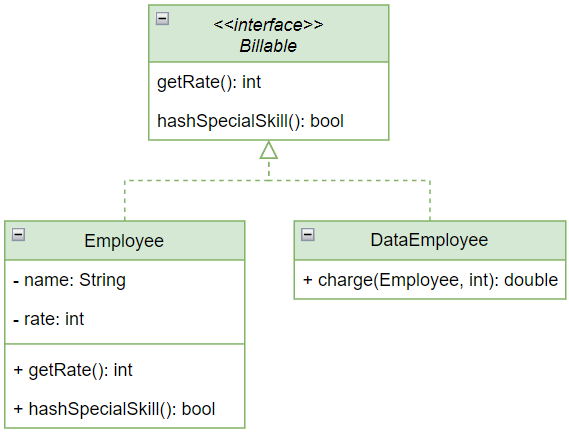


Figure 16: After use Extract Interface

**8.4 Demo**

**8.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/extractinterface>

**8.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/extractinterface>

CHAPTER 9: COLLAPSE HIERARCHY

9.1 Problem

A class hierarchy in which a subclass is practically the same as its superclass.

9.2 Solution

- Select the superclass or subclass.

- Use *Pull Up Field* and *Pull Up Method* if you decide to get rid of the subclass.

- If you choose to eliminate the superclass, use *Push Down Field* and *Push Down Method*.

- Replace all uses of the class that you’re deleting with the class to which the fields and methods are to be migrated.

- Delete the empty class.

9.3 Example

9.3.1 Before

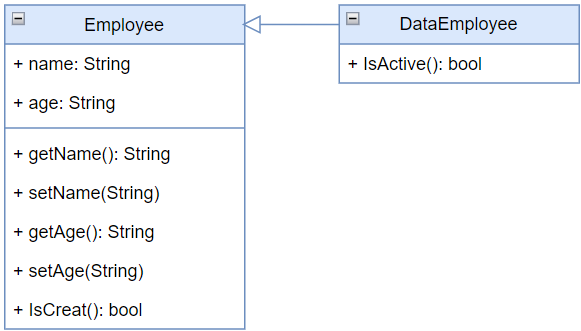


Figure 17: Before use Collapse Hierachy

9.3.2 After

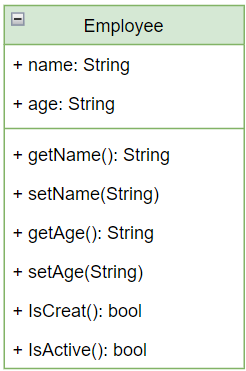


Figure 18: After use Collapse Hierachy

**9.4 Demo**

**9.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/collapsehierachy>

**9.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/collapsehierachy>

CHAPTER 10: FORM TEMPLATE METHOD

10.1 Problem

Subclasses implement algorithms that contain similar steps in the same order.

10.2 Solution

- Split algorithms in the subclasses into their constituent parts described in separate methods by using *Extract Method*.

- Use the *Pull Up Method* to move the resulting methods that are identical for all subclasses to the superclass.

- Rename the non-similar methods.

- Move the signatures of non-similar methods to a superclass as abstract ones by using *Pull Up Method*. Leave their implementations in the subclasses.

- Pull up the main method of the algorithm to the superclass. Now it should work with the method steps described in the superclass, both real and abstract.

10.3 Example

10.3.1 Before

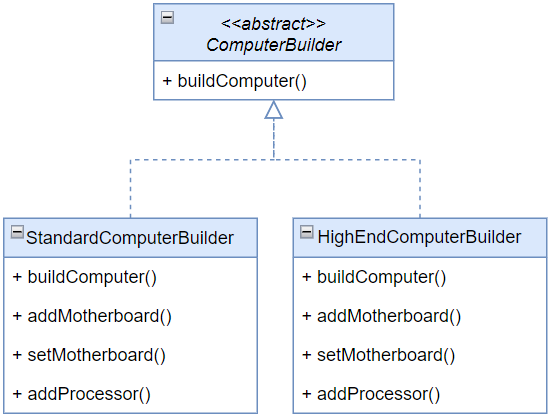


Figure 19: Before use Form Template Method

10.3.2 After

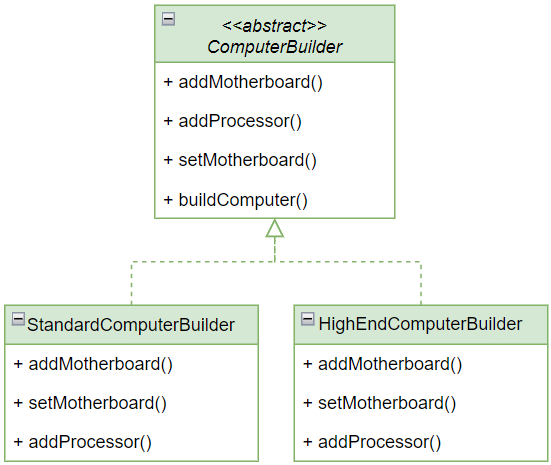


Figure 20: After use Form Template Method

**10.4 Demo**

**10.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/formtemplatemethod>

**10.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/formtemplatemethod>

CHAPTER 11: REPLACE INHERITANCE WITH DELEGATION

11.1 Problem

A subclass that uses only a portion of the methods of its superclass or is not possible to inherit superclass data.

11.2 Solution

- Create a field and place the current object in the subclass for holding the superclass.

- Change the subclass methods so that they use the superclass object.

- For methods inherited from the superclass that is called in the client code, create simple delegating methods in the subclass.

- Remove the inheritance declaration from the subclass.

- Change the initialization code of the field in which the former superclass is stored by creating a new object.

11.3 Example

11.3.1 Before

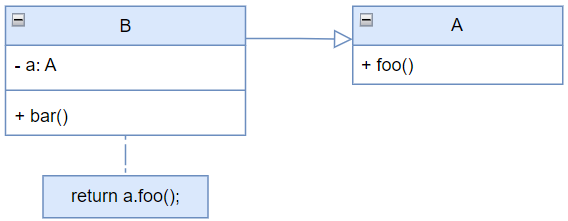


Figure 21: Before use Replace Inheritance With Delegation

11.3.2 After

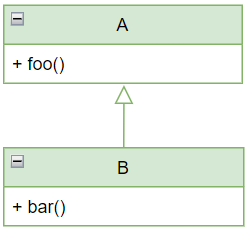


Figure 22: After use Replace Inheritance With Delegation

**11.4 Demo**

**11.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/replaceinheritancewithDelegation>

**11.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/replaceinheritancewithdelegation>

CHAPTER 12: REPLACE DELEGATION WITH INHERITANCE

12.1 Problem

A class contains many simple methods that delegate to all methods of another class.

12.2 Solution

- Make the class a subclass of the delegate class.

- Place the current object in a field containing a reference to the delegate object.

- Delete the methods with simple delegation one by one.

- If their names were different, use Rename Method to give all the methods a single name.

- Replace all references to the delegate field with references to the current object.

- Remove the delegate field.

12.3 Example

12.3.1 Before

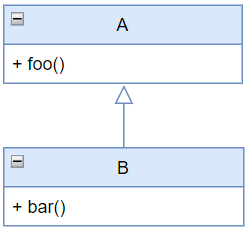


Figure 23: Before use Replace Delegation With Inheritance

12.3.2 After

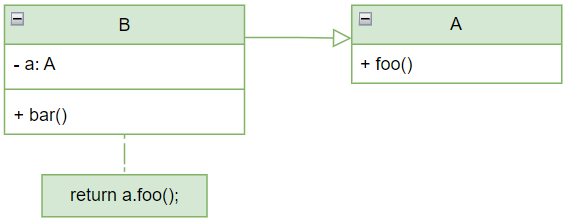


Figure 24: After use Replace Delegation With Inheritance

**12.4 Demo**

**12.4.1 Source Code Java**

<https://github.com/nhatlenlh/dealing-with-generalization/tree/master/src/replacedelegationwithInheritance>

**12.4.2 Source code C#**

<https://github.com/nhatlenlh/dealing-with-generalization-csharp/tree/master/dealing-with-generalization/replacedelegationwithinheritance>

REFERENCE

**Tiếng Việt**

**English**

**[1]** Martin Fowler, Kent Beck, John Brant, William Opdyke, don Roberts

(2002). “*Refactoring: Improving the Design of Existing Code”.* Chapter 11: Dealing with generalization, 259-289

**[2] refactoring.guru:**

https://refactoring.guru/refactoring/techniques/dealing-with-generalization