ASSIGNMENT 2 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 20: Advanced Programming		
Submission date	Monday, May 10, 2021	**Date Received 1st submission**	
Re-submission Date	Thursday, May 13, 2021	**Date Received 2nd submission**	
Student Name	Nguyen Quoc Viet	**Student ID**	GCC18157
Class	GCC07F1-1651	**Assessor name**	TRUNG-VIET NGUYEN

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	QuocViet

Grading grid

P3	P4	M3	M4	D3	D4

☐ **Summative Feedback:**	☐ **Resubmission Feedback:**

Grade:	**Assessor Signature:**	**Date:**
Lecturer Signature:		

ASSIGNMENT 2 BRIEF

Unit Number and Title	**20: Advance Programming**
Academic Year	2018
Unit Tutor	Hoàng Đức Quang
Assignment Title	**Assignment 2**
Issue Date	
Submission Date	
IV Name & Date	

Pass	Merit	Distinction
LO3 Implement code applying design patterns		**D3** Evaluate the use of design patterns for the given purpose specified in M3.
P3 Build an application derived from UML class diagrams.	**M3** Develop code that imple ments a design pattern for a given purpose.	
LO4 Investigate scenarios with respect to design patterns		**D4** Critically evaluate a range of design patterns against the range of given scenarios with justification of your choices.
P4 Discuss a range of design patterns with relevant examples of creational, structural and behavioural pattern types.	**M4** Reconcile the most appropriate design pattern from a range with a series of given scenarios.	

Aim of the assignment	This assignment satisfies the following learning outcomes and assessment criteria: **LO3: Implement code applying design patterns** P3. Build an application derived from UML class diagrams. **LO4: Investigate scenarios with respect to design patterns** P4. Discuss a range of design patterns with relevant examples of creational, structural and behavioural pattern types.

Specific requirements *(see Appendix for assessment criteria and grade descriptors)*	**Scenario:** Please see the scenario from Assignment Brief 1. **Task 1** In this task you will need to: • Code the application based on UML diagrams **Task 2** Please prepare a presentation with the following points • Coded UML Class diagram and explanation about Relationships among classes . • How did you implement main functionalities (Add, update, delete) with main flow and code snippet **Task 3** **Discuss a range of design patterns** • Describe the use of design patters with relevant examples of Singleton, Builder, Adapter, Iterator, Observer of Design Pattern
Student guidelines	For the assignment assessments, you are required to: • Produce a presentation to explain the code's structure, IDE's features such as code generation, debugging and show test cases and test result evaluations. • Write the program to fulfill the requirements
Submission requirements	Students are expected to submit hard copy of assignment

Appendix A- Grade Descriptor

In addition to the above PASS criteria, this assignment gives you the opportunity to submit evidence in order to achieve the following MERIT and DISTINCTION grades		
Grade Descriptor	Indicative characteristic/s	Contextualization

M3	Develop code that imple ments a design pattern for a given purpose.	
M4	Reconcile the most appropriate design pattern from a range with a series of given scenarios.	
D3	Evaluate the use of design patterns for the given purpose specified in M3.	
D4	Critically evaluate a range of design patterns against the range of given scenarios with justification of your choices.	

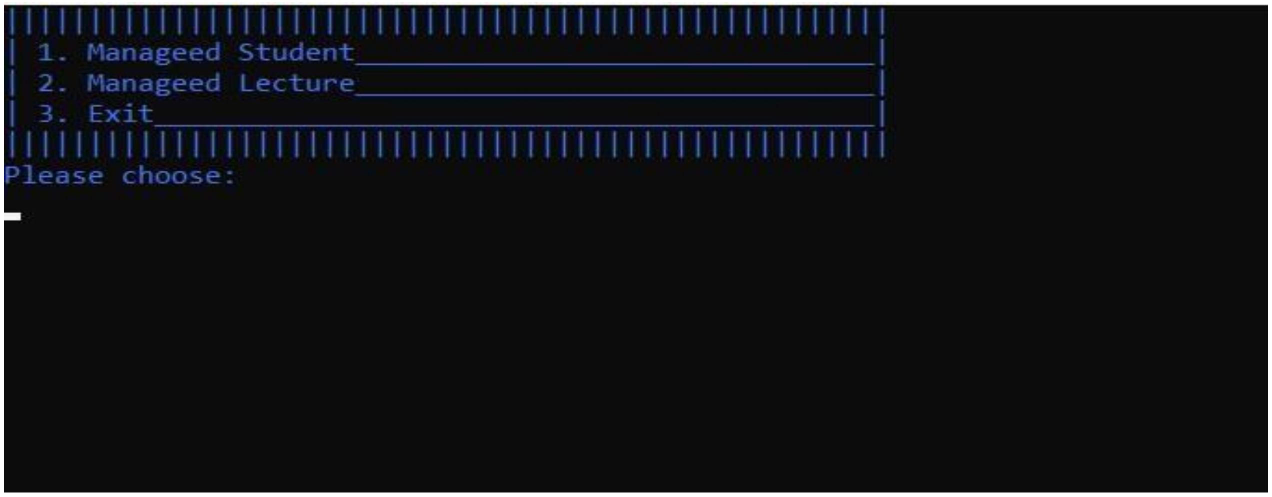
This brief has been verified as being fit for purpose

Internal Verifier 1		Signature		Date	
Internal Verifier 2		Signature		Date	

Contents

A. Build an application derived from UML class diagrams.

I. Main Menu

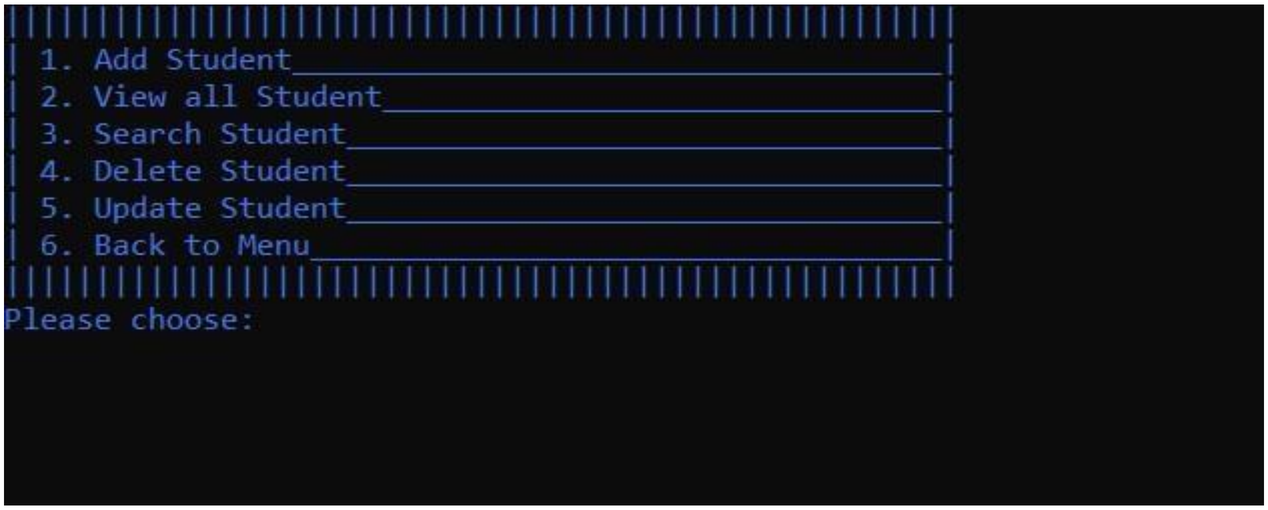


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Manageed Student_________________________________|
| 2. Manageed Lecture_________________________________|
| 3. Exit_____________________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
```

The following is a summary of the functions shown in Figure 1:

- When the user types 1, the software displays a submenu for student management.
- When the user enters 2 into the software, a submenu for managing lecturers will appear.
- The software will exit if the user enters 3.
- If the user enters a number other than 1, 2, or 3, the application will default to the Main Menu.

II. Manage Students



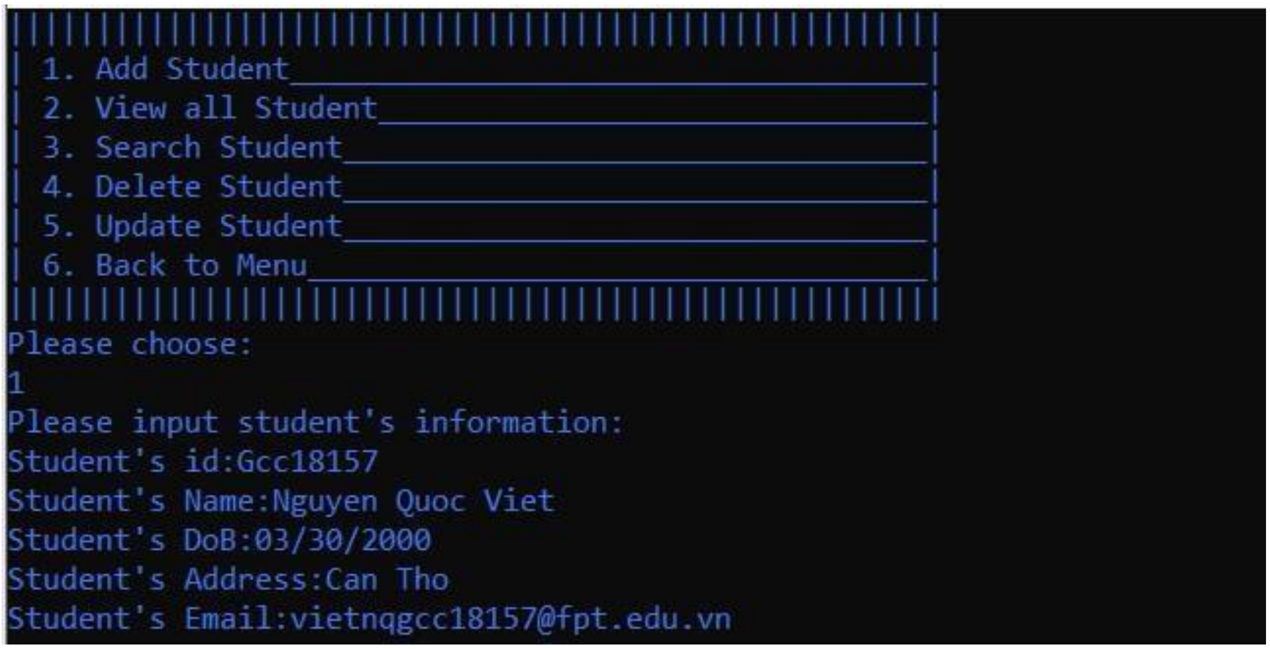
```
||||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student______________________________________|
| 2. View all Student_________________________________|
| 3. Search Student___________________________________|
| 4. Delete Student___________________________________|
| 5. Update Student___________________________________|
| 6. Back to Menu_____________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
```

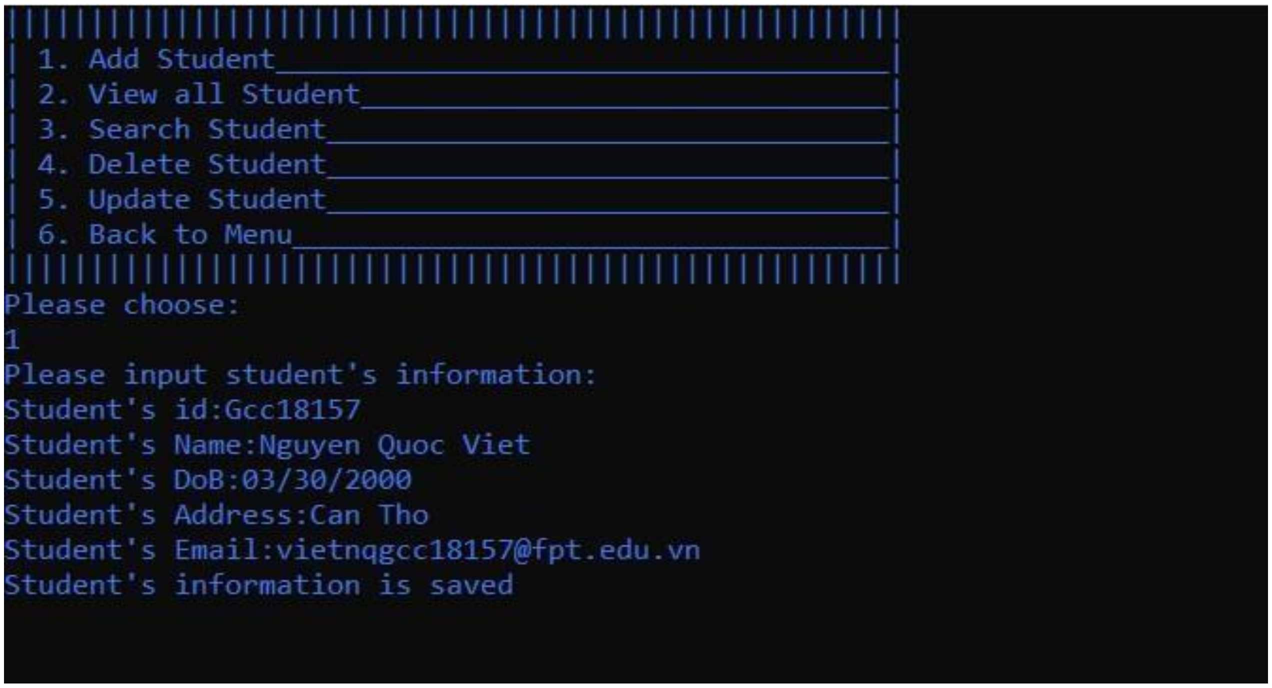
The software has the following functions: add student, display all student, check student, delete student, and return to student, as shown in Figure 2.

1. Add student

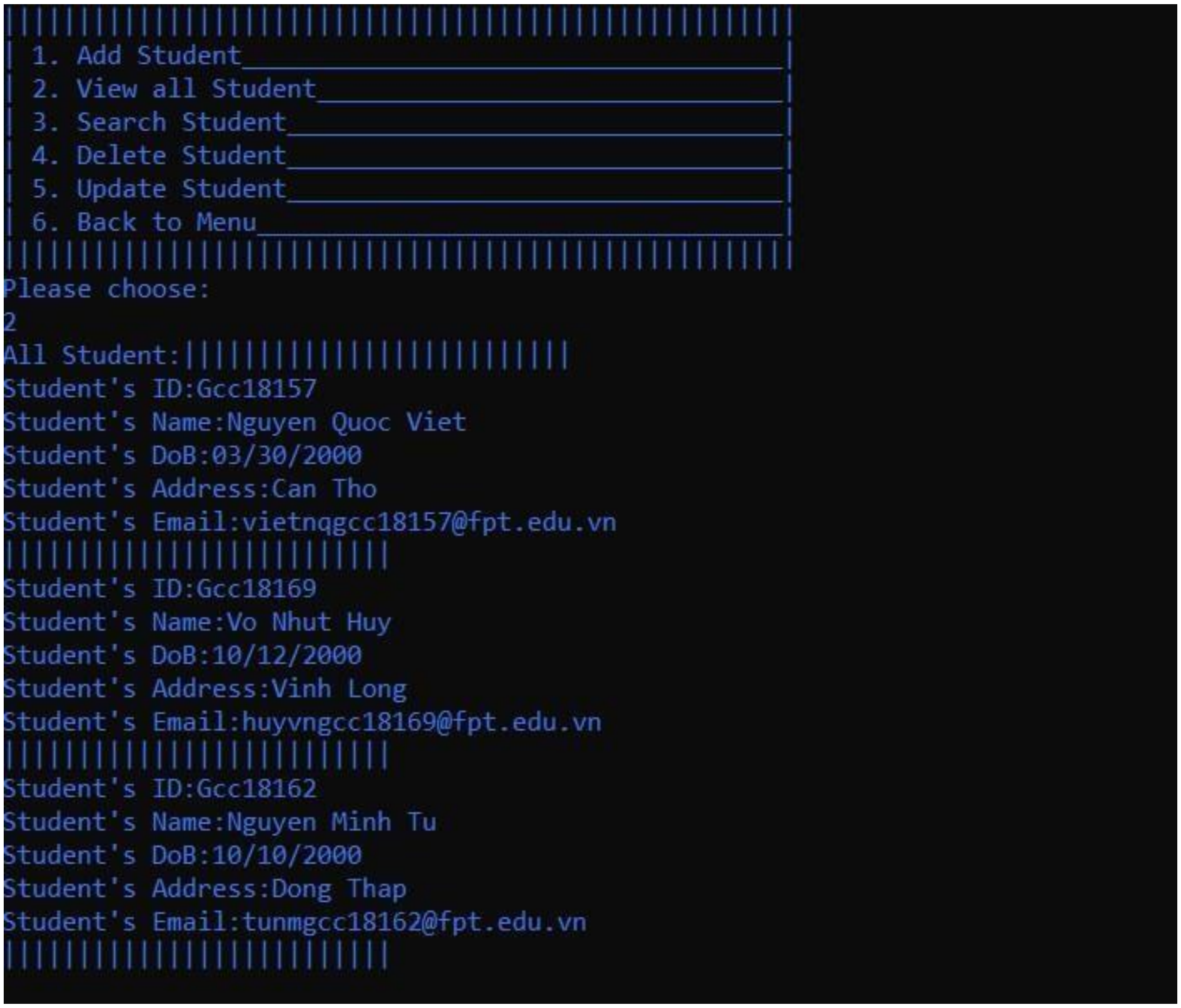
If the user enters 1, the program will ask for student information with the command "Please, input student's information:"

The user will enter all of the student information required by the program, including the student's ID, name, birthday, email, and address.


```
|||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student___________________________________|
| 2. View all Student______________________________|
| 3. Search Student________________________________|
| 4. Delete Student________________________________|
| 5. Update Student________________________________|
| 6. Back to Menu__________________________________|
|||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
1
Please input student's information:
Student's id:Gcc18157
Student's Name:Nguyen Quoc Viet
Student's DoB:03/30/2000
Student's Address:Can Tho
Student's Email:vietnqgcc18157@fpt.edu.vn
```
The user's student ID correctly enters the form's student ID, such as GTxxxxx (x: is a digit). Enter the year, month, and day of the user's birthday for the student.


```
|||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student___________________________________|
| 2. View all Student______________________________|
| 3. Search Student________________________________|
| 4. Delete Student________________________________|
| 5. Update Student________________________________|
| 6. Back to Menu__________________________________|
|||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
1
Please input student's information:
Student's id:Gcc18157
Student's Name:Nguyen Quoc Viet
Student's DoB:03/30/2000
Student's Address:Can Tho
Student's Email:vietnqgcc18157@fpt.edu.vn
Student's information is saved
```
The software will show "Student's information is saved" after the user enters valid Student information, and the student's information will be saved in the program.

2. **View all student**

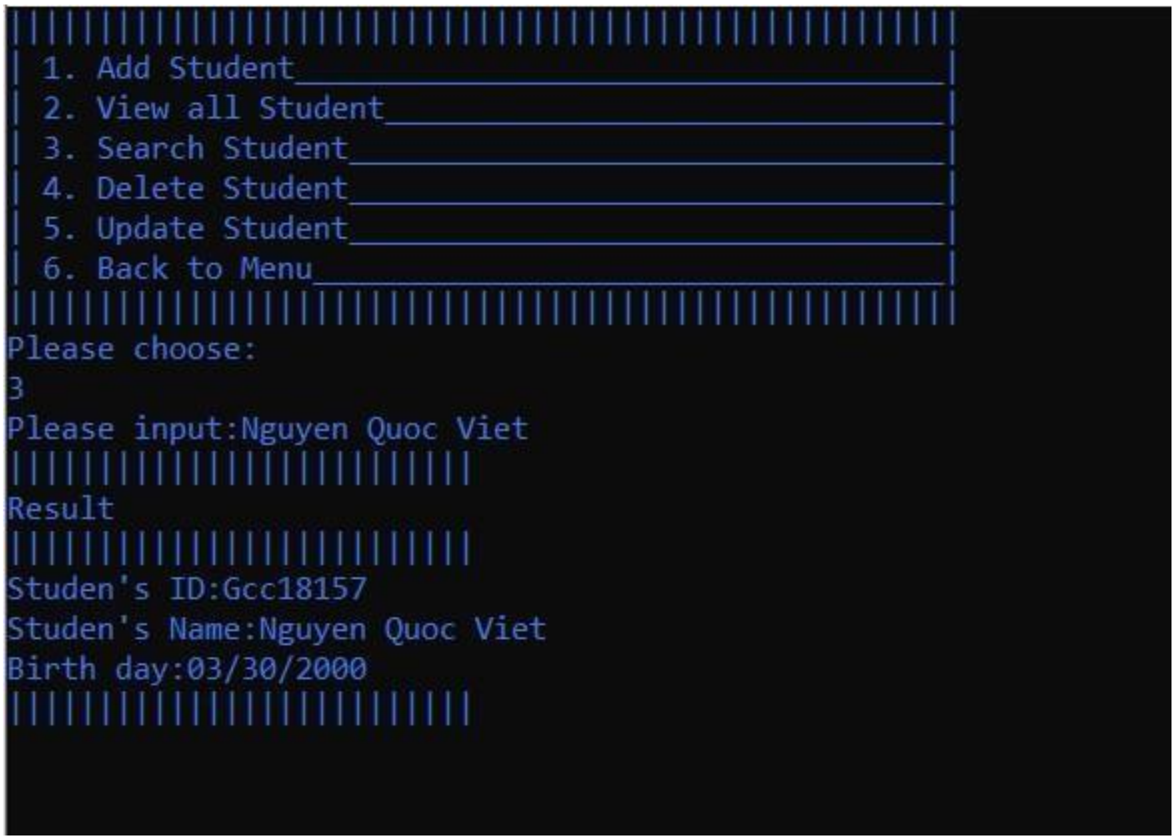


```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student____________________________________|
| 2. View all Student_______________________________|
| 3. Search Student_________________________________|
| 4. Delete Student_________________________________|
| 5. Update Student_________________________________|
| 6. Back to Menu___________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
2
All Student:||||||||||||||||||||||||||
Student's ID:Gcc18157
Student's Name:Nguyen Quoc Viet
Student's DoB:03/30/2000
Student's Address:Can Tho
Student's Email:vietnqgcc18157@fpt.edu.vn
||||||||||||||||||||||||||
Student's ID:Gcc18169
Student's Name:Vo Nhut Huy
Student's DoB:10/12/2000
Student's Address:Vinh Long
Student's Email:huyvngcc18169@fpt.edu.vn
||||||||||||||||||||||||||
Student's ID:Gcc18162
Student's Name:Nguyen Minh Tu
Student's DoB:10/10/2000
Student's Address:Dong Thap
Student's Email:tunmgcc18162@fpt.edu.vn
||||||||||||||||||||||||||
```

When the user types 2 into the application, it displays the entire list of students.

3. Search student

When the user presses 3 to look up a student's information, the program will prompt the user to enter the student's name or ID, along with the command "Please, input." To obtain the student's complete information.

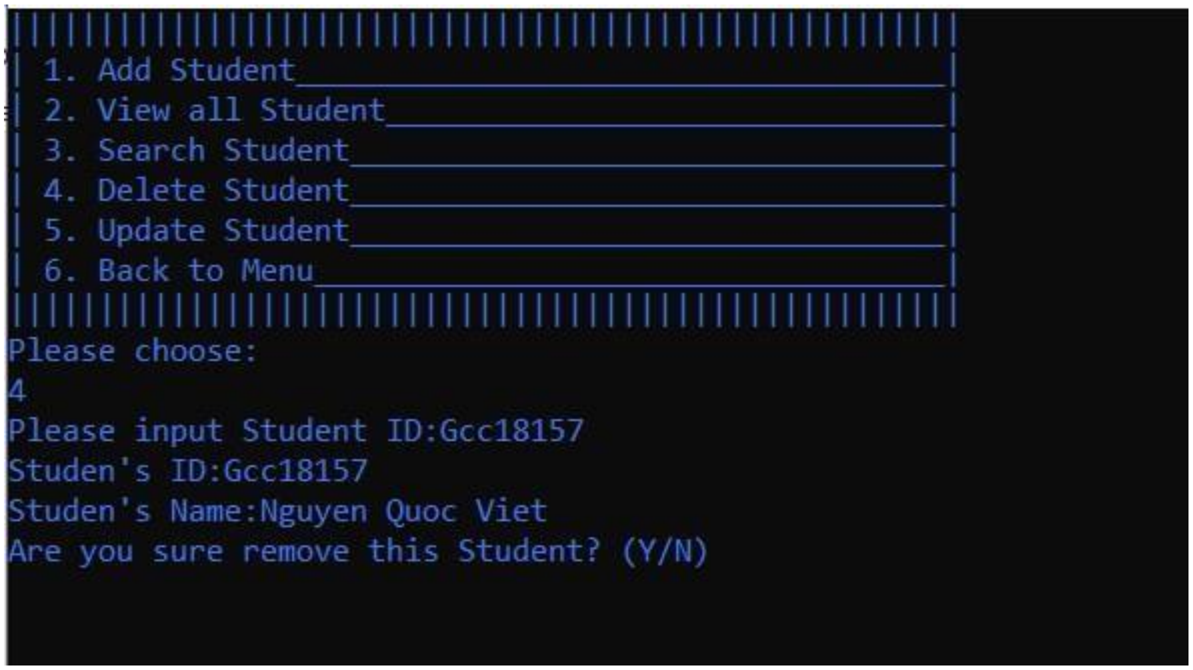


```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student_________________________________|
| 2. View all Student____________________________|
| 3. Search Student______________________________|
| 4. Delete Student______________________________|
| 5. Update Student______________________________|
| 6. Back to Menu________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
3
Please input:Nguyen Quoc Viet
||||||||||||||||||||||||||
Result
||||||||||||||||||||||||||
Studen's ID:Gcc18157
Studen's Name:Nguyen Quoc Viet
Birth day:03/30/2000
||||||||||||||||||||||||||
```

When “Nguyen Quoc Viet” is inserted into the program, all information about Student “Nguyen Quoc Viet” will be shown on the screen.

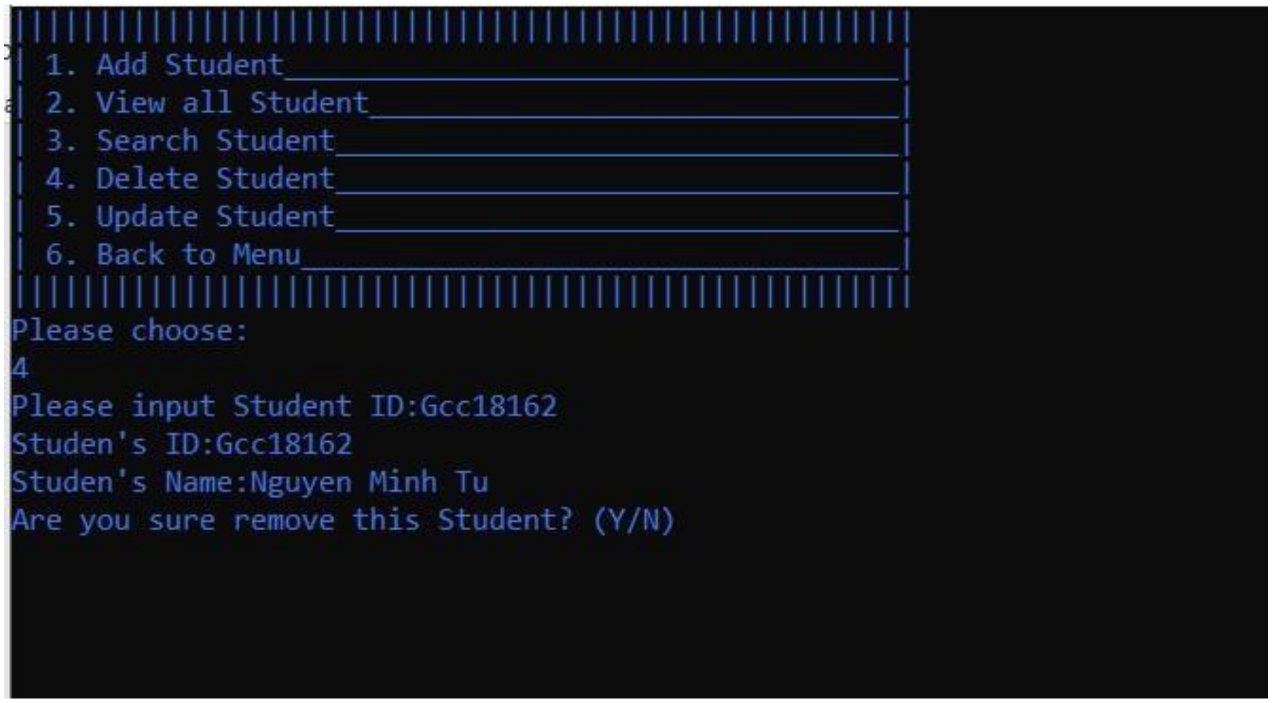
4. **Delete student**

When the user enters 4, the program will prompt the user for the student's ID with the command "Please, input StudentID:" in order to delete the student. If an ID is defined, it will display a notification informing the user that the student with that ID does exist; otherwise, it will display a notification informing the user that the student with that ID does not exist.

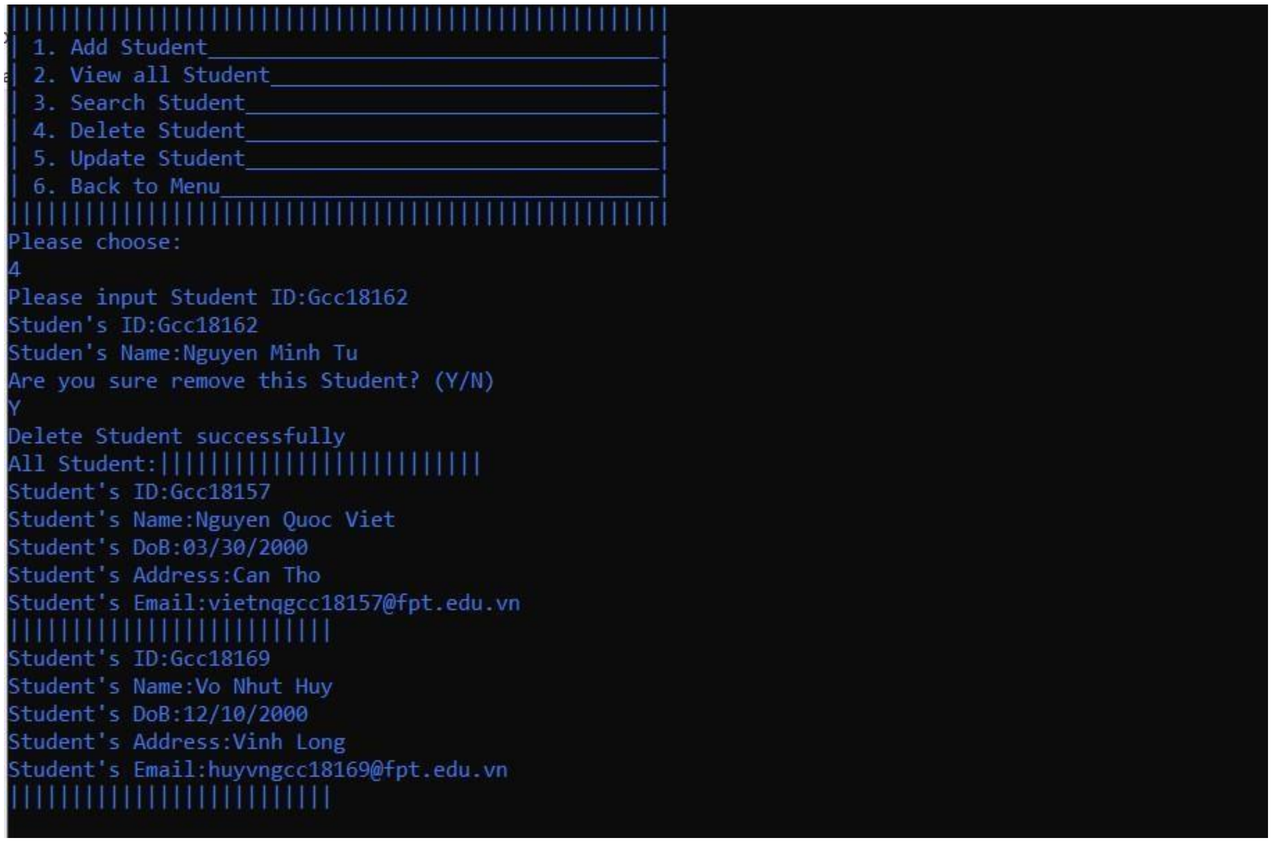


```
|||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student____________________________________|
| 2. View all Student_______________________________|
| 3. Search Student_________________________________|
| 4. Delete Student_________________________________|
| 5. Update Student_________________________________|
| 6. Back to Menu___________________________________|
|||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
4
Please input Student ID:Gcc18157
Studen's ID:Gcc18157
Studen's Name:Nguyen Quoc Viet
Are you sure remove this Student? (Y/N)
```

When the user types in the StudentID to delete, the software asks, "Are you sure remove this student?" If the user wishes to delete, type "Yes" and "No," eliminating the situation where the user accidentally enters the student to delete.



```
|||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student____________________________________|
| 2. View all Student_______________________________|
| 3. Search Student_________________________________|
| 4. Delete Student_________________________________|
| 5. Update Student_________________________________|
| 6. Back to Menu___________________________________|
|||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
4
Please input Student ID:Gcc18162
Studen's ID:Gcc18162
Studen's Name:Nguyen Minh Tu
Are you sure remove this Student? (Y/N)
```

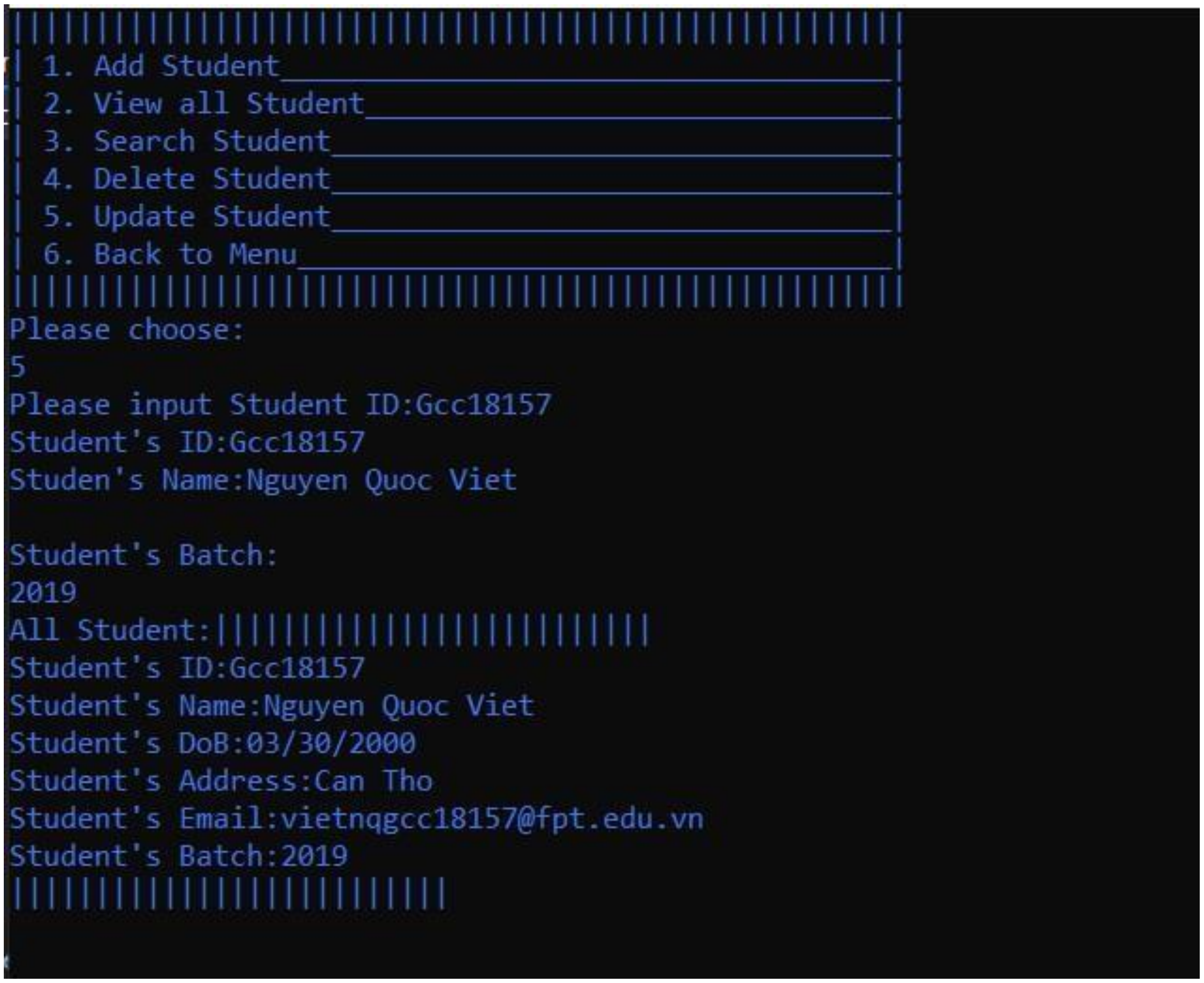


```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student___________________________________|
| 2. View all Student______________________________|
| 3. Search Student________________________________|
| 4. Delete Student________________________________|
| 5. Update Student________________________________|
| 6. Back to Menu__________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
4
Please input Student ID:Gcc18162
Studen's ID:Gcc18162
Studen's Name:Nguyen Minh Tu
Are you sure remove this Student? (Y/N)
Y
Delete Student successfully
All Student:|||||||||||||||||||||||||
Student's ID:Gcc18157
Student's Name:Nguyen Quoc Viet
Student's DoB:03/30/2000
Student's Address:Can Tho
Student's Email:vietnqgcc18157@fpt.edu.vn
||||||||||||||||||||||||||
Student's ID:Gcc18169
Student's Name:Vo Nhut Huy
Student's DoB:12/10/2000
Student's Address:Vinh Long
Student's Email:huyvngcc18169@fpt.edu.vn
||||||||||||||||||||||||||
```

The software then asks, "Are you sure remove this student?" The user then selects "Yes," and the software displays "Delete student successfully," indicating that the information has been saved.

5. **Update student**

When the user types 5 into the app, the program will prompt them to upgrade their StudentID. The software will then verify if the StudentID you just entered is valid. If the ID exists, data for each student's field will be displayed, and the user can either enter new data to update it or simply press enter to retain the school's current data. If the ID does not exist, the user will be notified.


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Student________________________________|
| 2. View all Student___________________________|
| 3. Search Student_____________________________|
| 4. Delete Student_____________________________|
| 5. Update Student_____________________________|
| 6. Back to Menu_______________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
5
Please input Student ID:Gcc18157
Student's ID:Gcc18157
Studen's Name:Nguyen Quoc Viet

Student's Batch:
2019
All Student:||||||||||||||||||||||||||
Student's ID:Gcc18157
Student's Name:Nguyen Quoc Viet
Student's DoB:03/30/2000
Student's Address:Can Tho
Student's Email:vietnqgcc18157@fpt.edu.vn
Student's Batch:2019
||||||||||||||||||||||||||
```

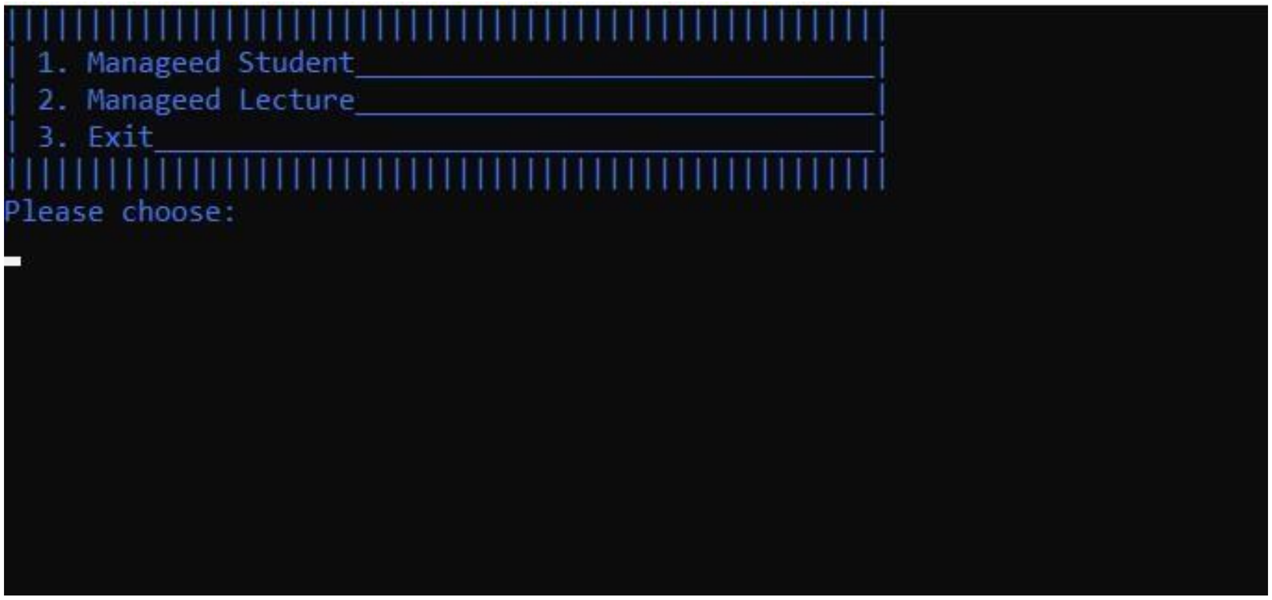
When the user presses 5 to update a student, the user selects StudentID: “GC18157” from the program's saved list. The program then asks the user to enter Student Batch information, which the user did, and the program successfully inserted Student Batch information for GC18157 student.

6. **Back to student**

When the user chooses 6, the program will back to the main menu.

III. Manage lecturers

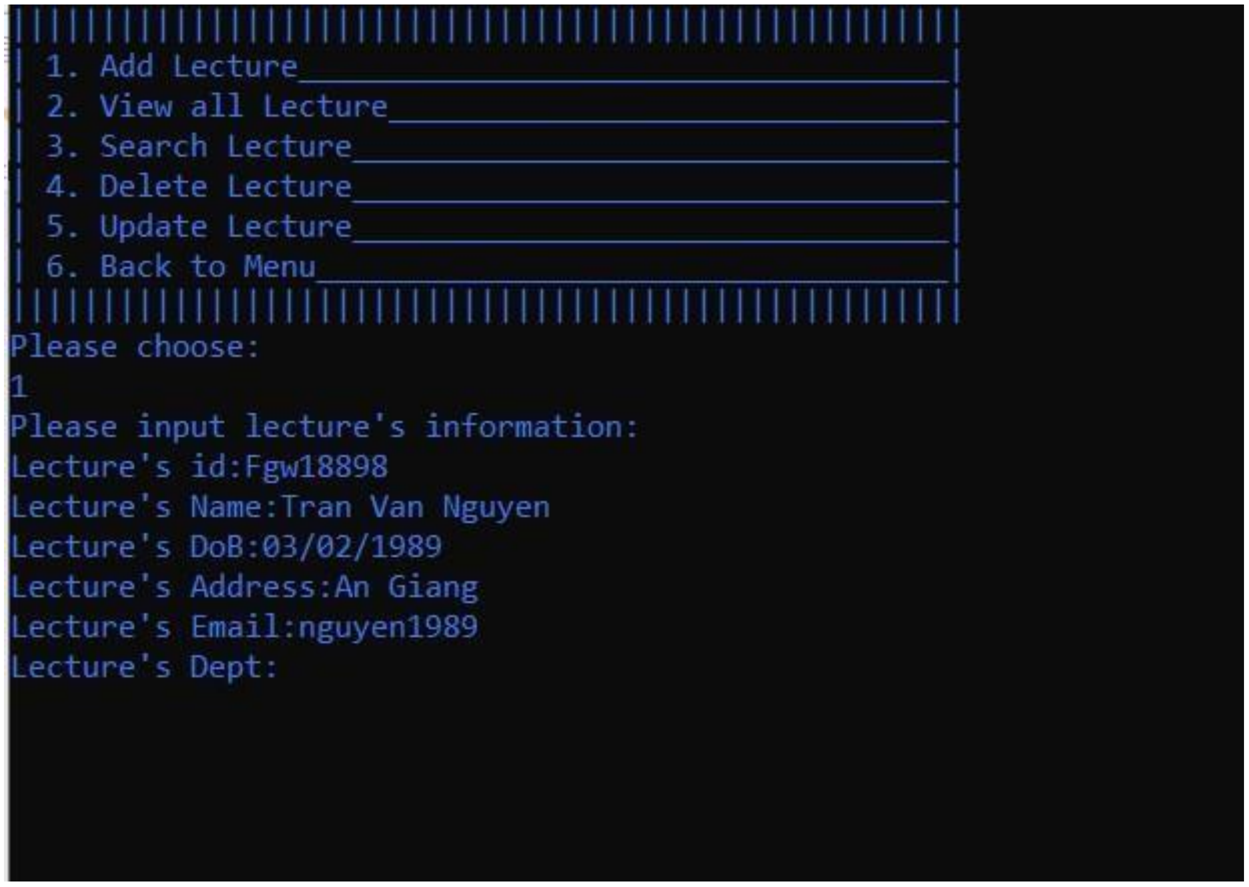
The software will show a submenu for controlling lecturers when the user enters 2:



```
|||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Manageed Student_______________________________|
| 2. Manageed Lecture_______________________________|
| 3. Exit___________________________________________|
|||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
```

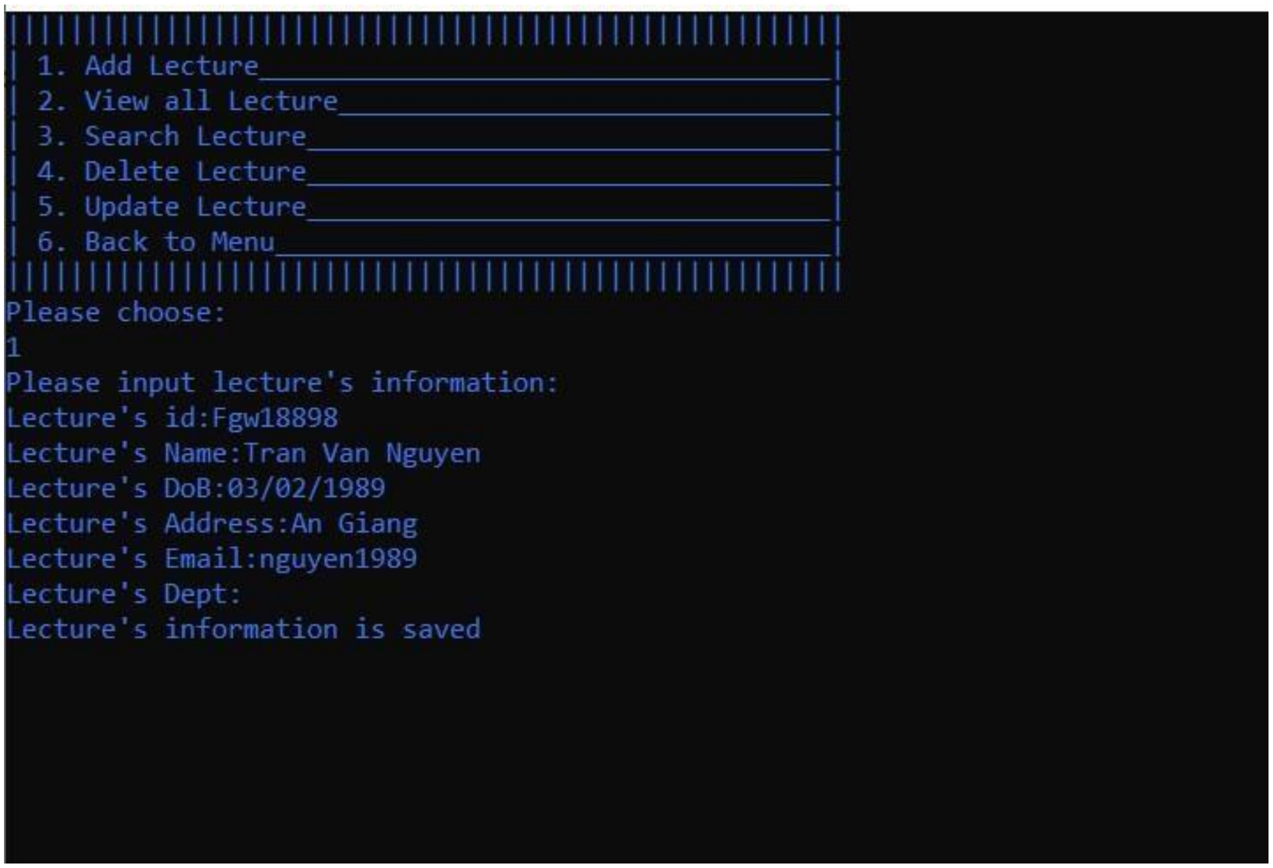
1. **Add lecturer**

If the user enters 1, the program will ask for lecturer information with the command "Please, input lecture’s information:"
The user will enter all lecturer information required by the program, including lecture ID, name, birthday, email, and address.



```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Lecture________________________________|
| 2. View all Lecture___________________________|
| 3. Search Lecture_____________________________|
| 4. Delete Lecture_____________________________|
| 5. Update Lecture_____________________________|
| 6. Back to Menu_______________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
1
Please input lecture's information:
Lecture's id:Fgw18898
Lecture's Name:Tran Van Nguyen
Lecture's DoB:03/02/1989
Lecture's Address:An Giang
Lecture's Email:nguyen1989
Lecture's Dept:
```

The user's Lecture ID correctly enters the form's Lecture ID, such as Fgwxxxxx (x: is a digit). Enter the year, month, and day of the user's birthday for the Lecture.

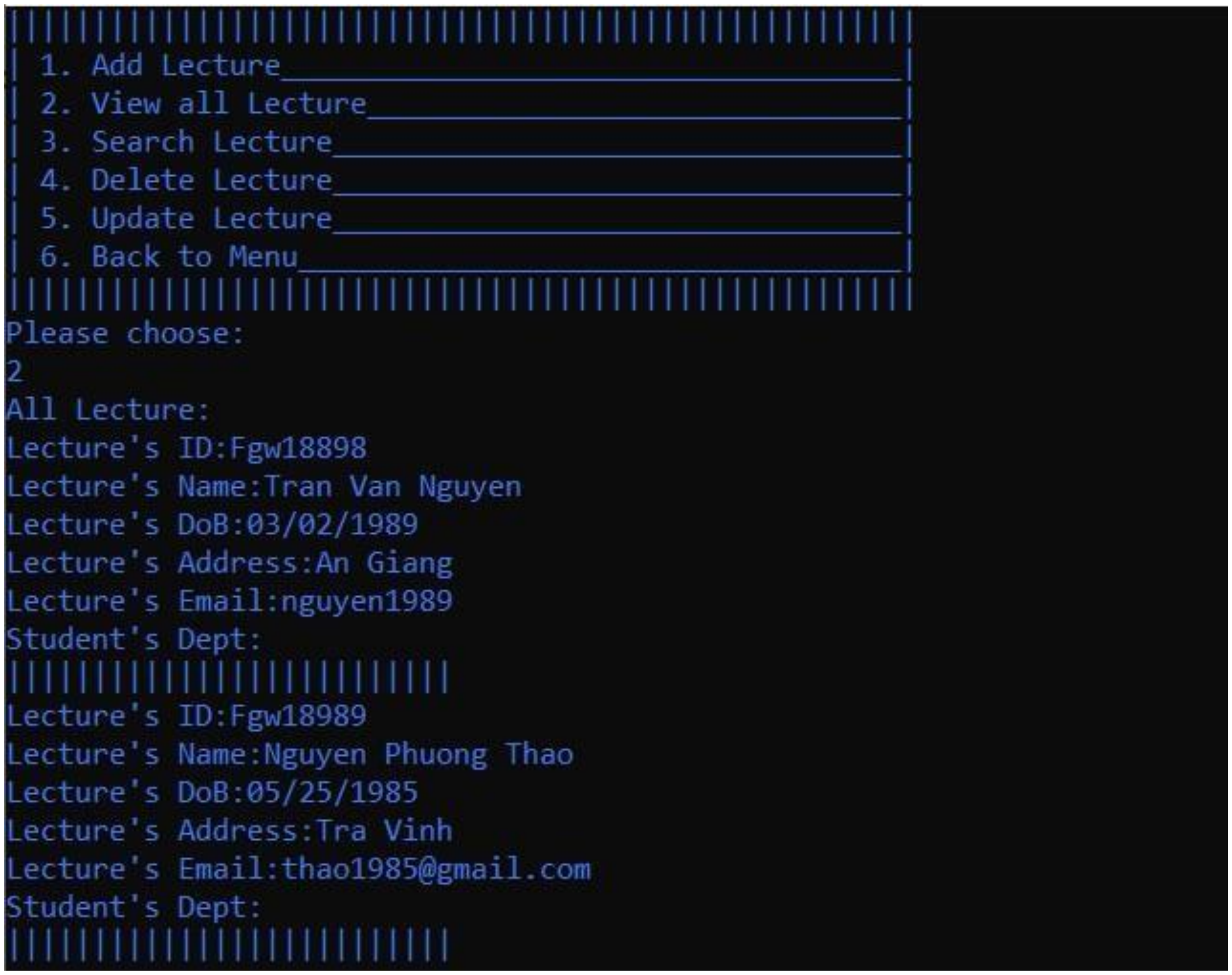


```
|||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Lecture__________________________________|
| 2. View all Lecture_____________________________|
| 3. Search Lecture_______________________________|
| 4. Delete Lecture_______________________________|
| 5. Update Lecture_______________________________|
| 6. Back to Menu_________________________________|
|||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
1
Please input lecture's information:
Lecture's id:Fgw18898
Lecture's Name:Tran Van Nguyen
Lecture's DoB:03/02/1989
Lecture's Address:An Giang
Lecture's Email:nguyen1989
Lecture's Dept:
Lecture's information is saved
```

The program will show "Lectures information is successful" after the user enters valid Lecturer information, and the lecturer information will be saved in the program.

2. **View all lecturer**

The program will show the command “Please, input lecturers information” after the user enters 2. The consumer then enters Lecturer ID information, and the program displays all of the program's Lecture information.

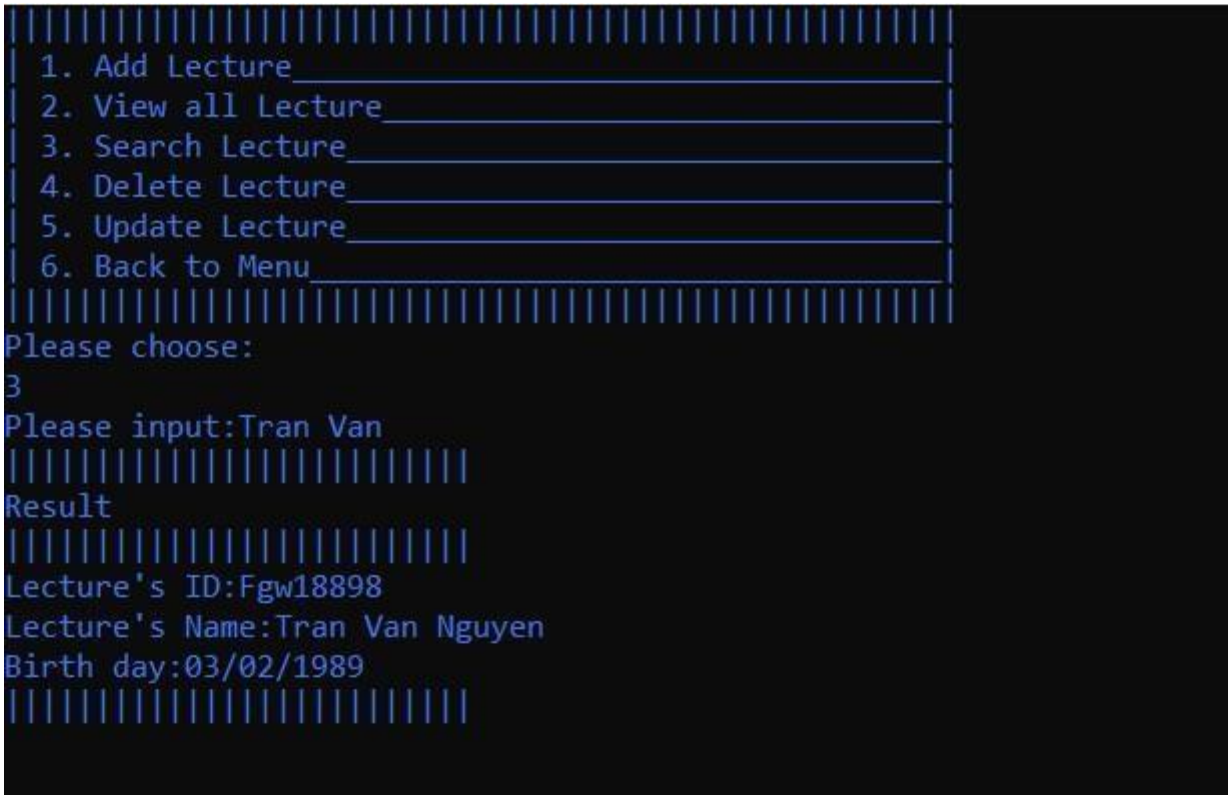


```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Lecture_________________________________|
| 2. View all Lecture____________________________|
| 3. Search Lecture______________________________|
| 4. Delete Lecture______________________________|
| 5. Update Lecture______________________________|
| 6. Back to Menu________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
2
All Lecture:
Lecture's ID:Fgw18898
Lecture's Name:Tran Van Nguyen
Lecture's DoB:03/02/1989
Lecture's Address:An Giang
Lecture's Email:nguyen1989
Student's Dept:
||||||||||||||||||||||||||
Lecture's ID:Fgw18989
Lecture's Name:Nguyen Phuong Thao
Lecture's DoB:05/25/1985
Lecture's Address:Tra Vinh
Lecture's Email:thao1985@gmail.com
Student's Dept:
||||||||||||||||||||||||||
```

The program will show the command "Please, input lecturers information" after the user enters 2. The consumer then enters Lecturer ID information, and the program displays all of the program's Lecture information.

3. Search lecturer

When you want to find details about a teacher, the user who enters three programs will ask you to enter the instructor's name using the command "Please, input." Users only need to enter a portion of the program instructor's name to find full details. The instructor to search for will be shown.

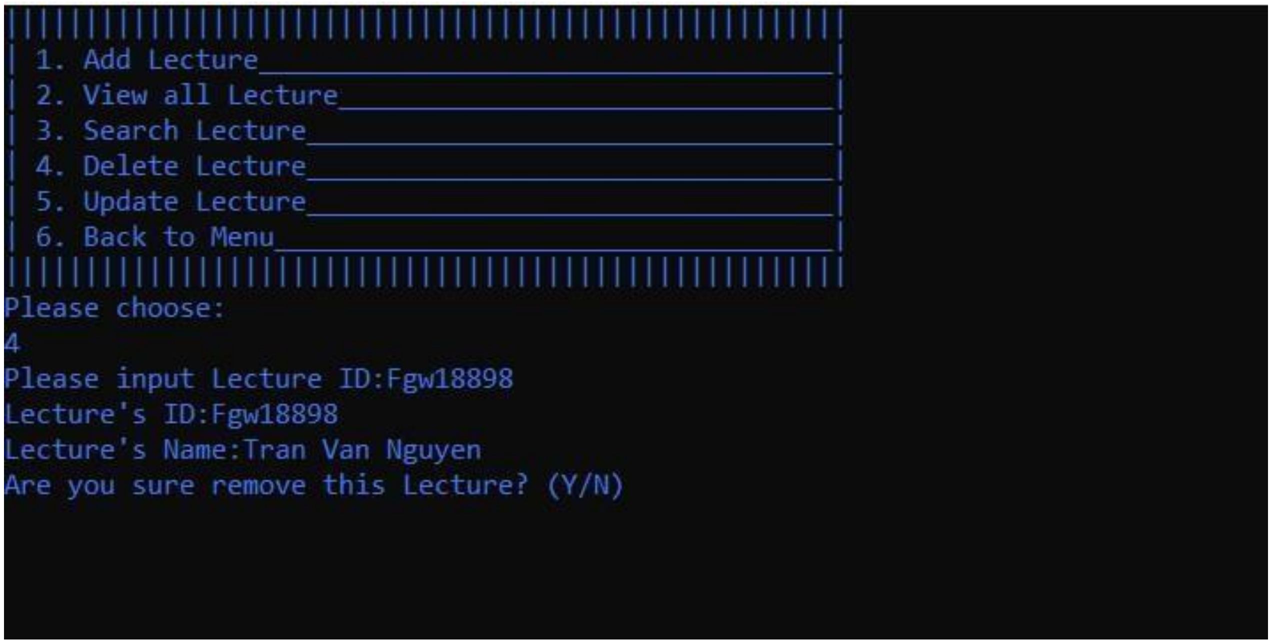


```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Lecture___________________________________|
| 2. View all Lecture______________________________|
| 3. Search Lecture________________________________|
| 4. Delete Lecture________________________________|
| 5. Update Lecture________________________________|
| 6. Back to Menu__________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
3
Please input:Tran Van
||||||||||||||||||||||||||
Result
||||||||||||||||||||||||||
Lecture's ID:Fgw18898
Lecture's Name:Tran Van Nguyen
Birth day:03/02/1989
||||||||||||||||||||||||||
```

When “Tran Van” is entered into the program, the program will display all information about lecturer “Tran Van Nguyen” on the screen.

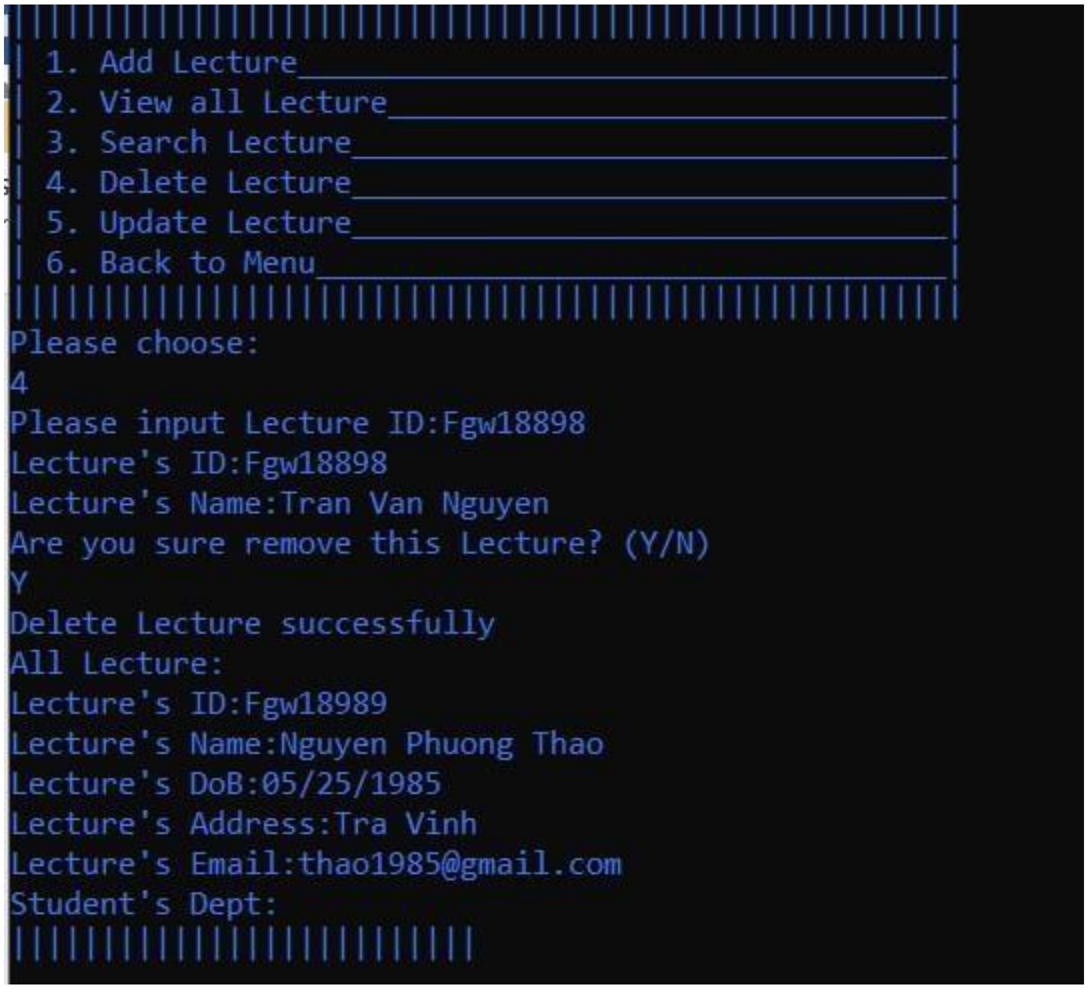
4. Delete lecture

When the user enters 4, the program will ask for the lecturer's ID and prompt the user to delete the lecturer with the command "Please, input lecturer ID:" If an ID is defined and that ID exists, a message will be shown informing the user that the lecturer that ID does not exist.



```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Lecture____________________________________|
| 2. View all Lecture_______________________________|
| 3. Search Lecture_________________________________|
| 4. Delete Lecture_________________________________|
| 5. Update Lecture_________________________________|
| 6. Back to Menu___________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
4
Please input Lecture ID:Fgw18898
Lecture's ID:Fgw18898
Lecture's Name:Tran Van Nguyen
Are you sure remove this Lecture? (Y/N)
```

When the user types in the Lecturer ID to delete, the software asks, "Are you sure you want to delete this lecturer?" If the user wishes to delete, type "Yes" and "No," eliminating the situation where the user accidentally enters the lecturer to delete.

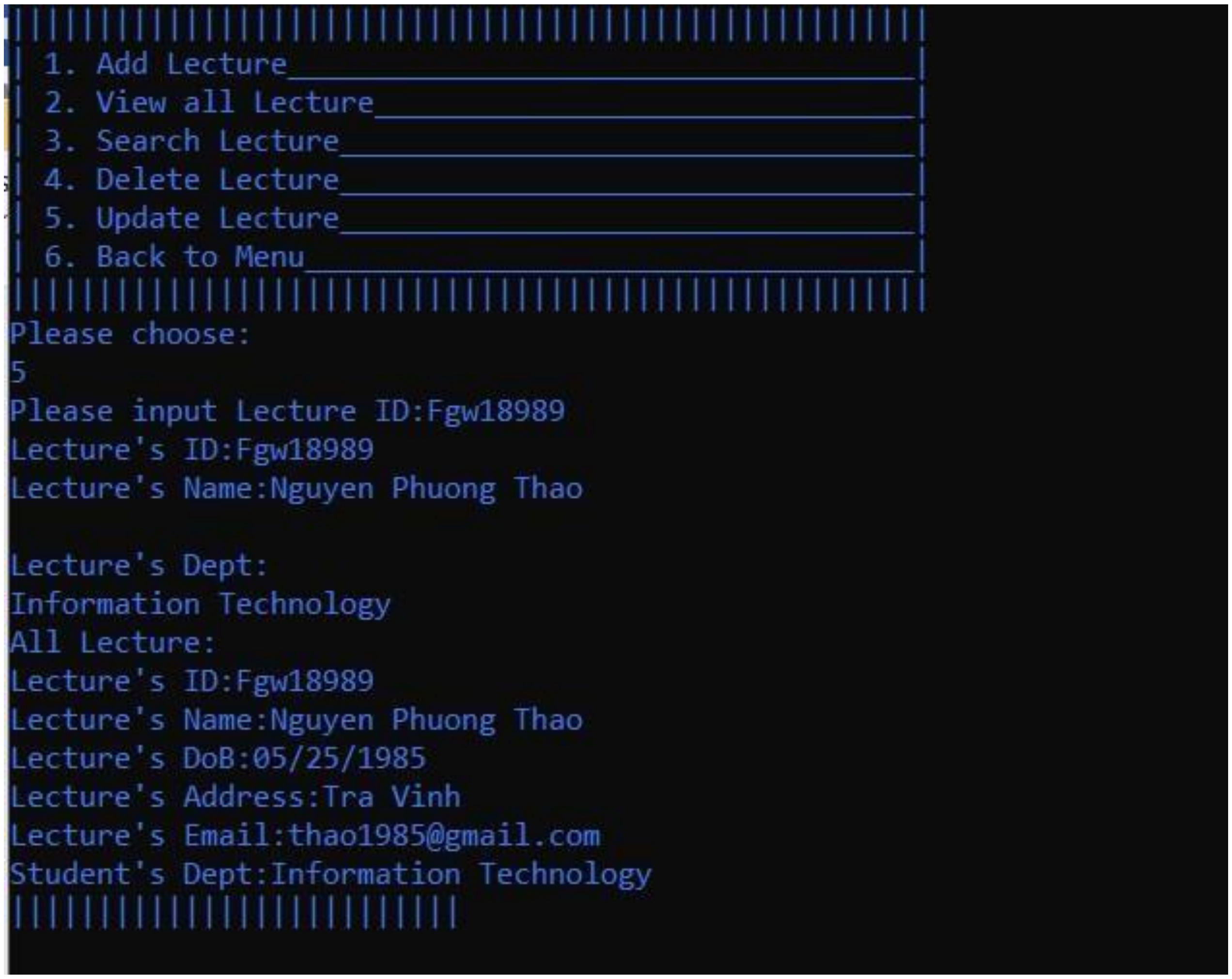


```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Lecture____________________________________|
| 2. View all Lecture_______________________________|
| 3. Search Lecture_________________________________|
| 4. Delete Lecture_________________________________|
| 5. Update Lecture_________________________________|
| 6. Back to Menu___________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
4
Please input Lecture ID:Fgw18898
Lecture's ID:Fgw18898
Lecture's Name:Tran Van Nguyen
Are you sure remove this Lecture? (Y/N)
Y
Delete Lecture successfully
All Lecture:
Lecture's ID:Fgw18989
Lecture's Name:Nguyen Phuong Thao
Lecture's DoB:05/25/1985
Lecture's Address:Tra Vinh
Lecture's Email:thao1985@gmail.com
Student's Dept:
||||||||||||||||||||||||||
```

The software then asks, "Are you sure you want to drop this lecturer?" The user then selects "Yes," and the software displays "Delete lecturer successfully," indicating that the information has been saved.

5. **Update lecturer**

When the user types 5 into the program, it will first prompt the user to change the lecturer ID. The software will then verify if the lecturer ID you just entered is valid. If the ID exists, data for each lecturer area will be shown, and the user can update the data or simply press enter to retain the current data for the school. It will warn the user that the lecturer does not exist if the ID does not exist.



```
||||||||||||||||||||||||||||||||||||||||||||||||||||
| 1. Add Lecture___________________________________|
| 2. View all Lecture______________________________|
| 3. Search Lecture________________________________|
| 4. Delete Lecture________________________________|
| 5. Update Lecture________________________________|
| 6. Back to Menu__________________________________|
||||||||||||||||||||||||||||||||||||||||||||||||||||
Please choose:
5
Please input Lecture ID:Fgw18989
Lecture's ID:Fgw18989
Lecture's Name:Nguyen Phuong Thao

Lecture's Dept:
Information Technology
All Lecture:
Lecture's ID:Fgw18989
Lecture's Name:Nguyen Phuong Thao
Lecture's DoB:05/25/1985
Lecture's Address:Tra Vinh
Lecture's Email:thao1985@gmail.com
Student's Dept:Information Technology
||||||||||||||||||||||||||
```

When pressing 5 to update lecturers, user selects trainer ID: “Fgw18989” from the saved list of the program. Then, the program asks the user to enter information about the faculty part that the user input is "information technology" and the program has successfully inserted the trainer department information for the instructor.

6. Back to student

When a user enters 6, the program will back to the main menu.

IV. Test case:

No.	Test case	Functio n	Input data	Expected output	Actual output	Evalu ation
1.	Verify that go to the main menu	Manage student	To program, type 1 into the box.	the program will	the program will display submenu for	Pass

	to select successful student management			display submenu for managing students	managing students	
2.	Verify that go to main menu to select successful lecturer management	Manage lecturer	To program, type 2 into the box.	the program will display submenu for managing lecturers	the program will display submenu for managing lecturers	Pass
3.	Verify that exit was successful	Exit	To program, type 3 into the box.	the program will exit.	the program will exit.	Pass
4.	Verify that add student information was created in the successful program	Add student	Enter the correct student details (student ID, name, date of birth, email, and address) (“Student ID: Gcc18157”, “Name: Nguyen Quoc Viet”, “date of birth: 03/30/2000”, “email: vietnqgcc18157@fpt.edu.vn” “Address: Can Tho”)	Display "Add student information is successfully" and is saved to the program	Display "Add student information is successfully" and is saved to the program	Pass
5.	Verification that add student information was created in the	Add student	Gcc123456789 is the incorrect student ID.	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass

	program failed					
6.	Verification that add student information was created in the program failed	Add student	Fill in the blanks in the ID ("Gcc1815")	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass
7.	Verification that add student information was created in the program failed	Add student	Fill in the blanks with text(“DFGHWQR”).	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass
8.	Verification that add student information was created in the program failed	Add student	Fill in the blanks with a number (“123456789”).	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass
9.	Verification that add student information was created in the program failed	Add student	Fill in the blanks with residual characters (“GC12345678”).	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass

10.	Verify that all student information is displayed in successful programs	View all student	In the program, type the number "2."	Display all student information	Display all student information	Pass
11.	Verification that the student information is all displayed in the failed program	View all student	Enter a different number, such as "2."	Does not display student information	Does not display student information	Pass
12.	Verify that the search student in the program is successful	Search student	Enter the student ID that will be looked up in the software ("Gcc18157")	Display all student search information	Display all student search information	Pass
13.	Verify that the search student in the program failed	Search student	Input the incorrect student ID ("Gcc1234")	Does not display information students need to search for and allows re-entering student ID	Does not display information students need to search for and allows re-entering student ID	Pass
14.	Verify that the search student in the program failed	Search student	With no data in the program, enter the incorrect student ID.	Does not display information students need to search for and allows	Does not display information students need to search for and allows re-entering student ID	Pass

				re-entering student ID		
15.	Verify that the search student in the program failed	Search student	With no data in the software, enter the wrong student name.	Does not display information students need to search for and allows re-entering name student	Does not display information students need to search for and allows re-entering name student	Pass
16.	Verify that student information in the program was successfully deleted.	Delete student	To be removed from the program's records, enter the correct student ID. ("Gcc18157") is an abbreviated version of "Gcc18157	Display "Delete student information successfully" and saved in the program	Display "Delete student information successfully" and saved in the program	Pass
17.	Verify that student information in the program was unsuccessfully deleted.	Delete student	The student ID you're trying to uninstall doesn't exist in the software.	Deletion failed, the results will show the remaining students	Deletion failed, the results will show the remaining students	Pass
18.	Verify that student information in the program was unsuccessfully deleted.	Delete student	To delete the balance of a student ID, enter it here ("Gcc18157")	Deletion failed, the results will show the remaining students	Deletion failed, the results will show the remaining students	Pass

19.	Verify that student information in the program was unsuccessfully deleted.	Delete student	Enter the missing characters in the student ID ("Gcc181")	Deletion failed, the results will show the remaining students	Deletion failed, the results will show the remaining students	Pass
20.	Verify that student information in the program has been successfully updated.	Update student	To change, enter the correct student ID.	Display "Update student information successfully" and save data into the program	Display "Update student information successfully" and save data into the program	Pass
21.	Verify that student information in the program was updated unsuccessfully.	Update student	The student ID you entered does not exist in the program's database.	Update student information failed and show all remaining student information	Update student information failed and show all remaining student information	Pass
22.	Verify that student information in the program was updated unsuccessfully.	Update student	To change, enter the incorrect student ID.	Update student information failed and show all remaining student information	Update student information failed and show all remaining student information	Pass
23.	Verify that the back to menu is successful	Back to menu	Digit “6” is entered.	The program exits the main menu	The program exits the main menu	Pass

24.	Verify that add lecturer information was created in the successful program	Add lecture	Digit “6” is entered. Lecturer (lecture ID, name, date of birth, email, address): (“LecturerID: Fgw18898”, “Name: TRAN VAN Nguyen”, “Date of birth: 03/02/1989”, “Email: nguyen1989@gamil.com”, “Address: An Giang”).	Display "Add lecture information is successfully" and is saved to the program	Display "Add lecture information is successfully" and is saved to the program	Pass
25.	Verification that add lecturer information was created in the program failed	Add lecturer	Fill in the blanks in the ID ("Fgw188")	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass
26.	Verification that add lecturer information was created in the program failed	Add lecturer	Enter the ID number of a residual trainer ("Fgw18989")	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass
27.	Verification that add lecturer information was	Add lecture	Use alphanumeric characters to enter the	Are not saved in the program and allowed to	Are not saved in the program and allowed to import again	Pass

	created in the program failed		instructor's ID ("Fgw18898")	import again		
28.	Verification that add lecturer information was created in the program failed	Add lecturer	Fill in a teacher ID that isn't filled in yet.	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass
29.	Verification that add lecturer information was created in the program failed	Add lecturer	Enter all lecturer IDs that are numeric ("ABCDEF")	Are not saved in the program and allowed to import again	Are not saved in the program and allowed to import again	Pass
30.	Verify that all lecturer information is displayed in successful programs	View all lecturer	Enter the number “2” here.	Display all lecturer information	Display all lecturer information	Pass
31.	Verification that the lecturer information is all displayed in the failed program	View all lecture	Enter a different number, such as "2."	Does not display lecturer information	Does not display lecturer information	Pass

32.	Verify that the search lecturer in the program is successful	Search lecturer	Enter the lecture ID that you want to find in the software ("Fgw18989")	Display all student search information	Display all student search information	Pass
33.	Verify that the search lecturer in the program failed	Search lecturer	Input the incorrect lecture ID ("Fgw1234567")	Does not display information students need to search for and allows re-entering student ID	Does not display information students need to search for and allows re-entering student ID	Pass
34.	Verify that the search lecturer in the program failed	Search lecturer	There isn't a lecturer ID to enter.	Does not display information students need to search for and allows re-entering student ID	Does not display information students need to search for and allows re-entering student ID	Pass
35.	Verify that the search lecturer in the program failed	Search lecturer	With no data in the software, enter the wrong lecturer's name.	Does not display information students need to search for and allows re-entering student ID	Does not display information students need to search for and allows re-entering student ID	Pass
36.	Verify that lecturer information in the program was	Delete lecture	To delete a lecture from the program details, enter the correct lecture code.	Display "Delete lecturer information successfully" and	Display "Delete lecturer information successfully" and saved in the program	Pass

	successfully deleted.			saved in the program		
37.	Verify that lecturer information in the program was unsuccessfully deleted.	Delete lecturer	The lecturer ID that needs to be removed does not exist in the software.	Deletion failed, the results will show the remaining lecturers	Deletion failed, the results will show the remaining lecturers	Pass
38.	Verify that lecturer information in the program was unsuccessfully deleted.	Delete lecturer	To delete the lecturer ID balance, enter it here ("Fgw18898")	Deletion failed, the results will show the remaining lecturers	Deletion failed, the results will show the remaining lecturers	Pass
39.	Verify that lecturer information in the program was unsuccessfully deleted.	Delete lecturer	Enter the missing characters in the lecturer ID ("Fgw188")	Deletion failed, the results will show the remaining lecturers	Deletion failed, the results will show the remaining lecturers	Pass
40.	Verify that lecturer information in the program has been successfully updated.	Update lecturer	To update, enter the correct lecturer ID.	Display "Update lecturer information successfully" and save data into the program	Display "Update lecturer information successfully" and save data into the program	Pass
41.	Verify that lecturer	Update lecturer	The lecturer ID you entered is	Update lecturer	Update lecturer	Pass

	information in the program was updated unsuccessfully		not found in the program results.	information failed and show all remaining lecturer information	information failed and show all remaining lecturer information	
42.	Verify that lecturer information in the program was updated unsuccessfully	Update lecturer	To change, enter the incorrect lecturer ID.	Update lecturer information failed and show all remaining lecturer information	Update lecturer information failed and show all remaining lecturer information	Pass
43.	Verify that the back to menu is successful.	Back to menu	Numbers “6” should be entered.	The program exits the main menu	The program exits the main menu	Pass

B. Discuss a range of design patterns with relevant examples of creational, structural and behavioral pattern types.

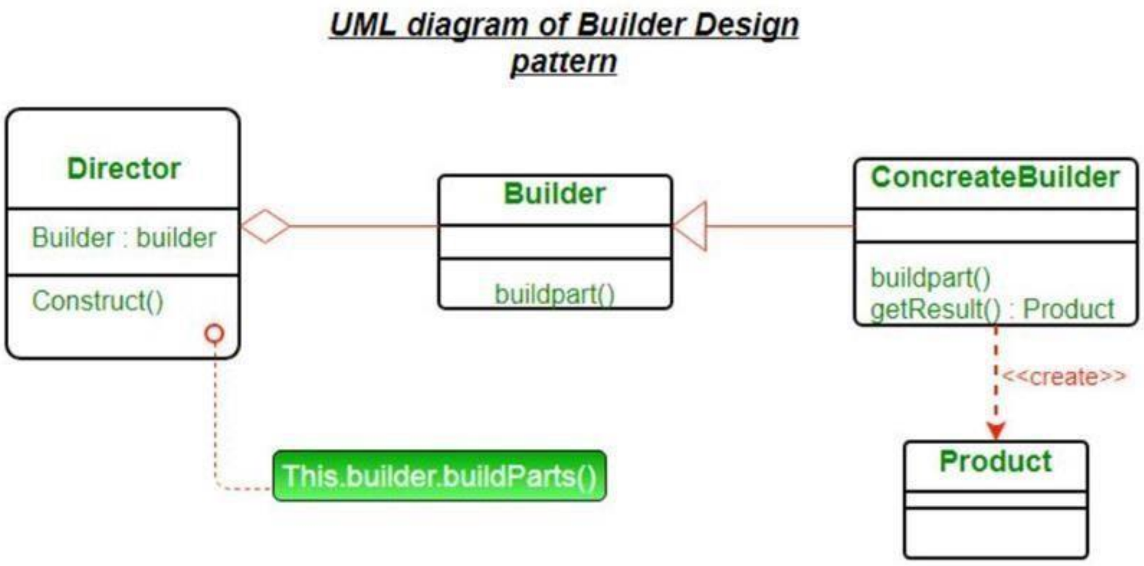
I. Builder pattern

1. Definition

Distinguish the construction of a complex object from its representation such that the same construction methods can be used to produce several representations.

2. Concept

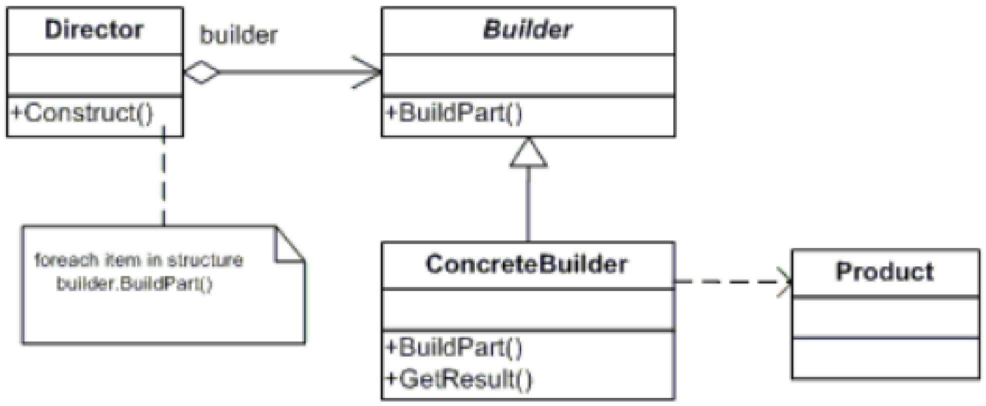
The Builder pattern can be used to make complex objects with multiple pieces.
The construction process should be unconcerned about how these pieces are put together; in other words, the construction process should be unconcerned about how they are put together.
Furthermore, you should be able to build different representations of the objects using the same construction method.



3. **Real-Life Example**

Different hardware parts are assembled based on consumer requirements to complete a machine order. A consumer can choose between a 500GB hard disk with an Intel processor and a 250GB hard disk with an AMD processor, for example.

4. **Class diagram**



5. **Implementation**

```
using System;
using System.Collections.Generic;

namespace DoFactory.GangOfFour.Builder.Structural
{
  /// <summary>

  /// MainApp startup class for Structural

  /// Builder Design Pattern.

  /// </summary>

  public class MainApp

  {
    /// <summary>

    /// Entry point into console application.

    /// </summary>
    public static void Main()
    {
      // Create director and builders

      Director director = new Director();

      Builder b1 = new ConcreteBuilder1();
      Builder b2 = new ConcreteBuilder2();

      // Construct two products

      director.Construct(b1);
      Product p1 = b1.GetResult();
      p1.Show();

      director.Construct(b2);
      Product p2 = b2.GetResult();
      p2.Show();

      // Wait for user

      Console.ReadKey();
    }
  }

  /// <summary>

  /// The 'Director' class

  /// </summary>
```

```
class Director

{
  // Builder uses a complex series of steps

  public void Construct(Builder builder)
  {
    builder.BuildPartA();
    builder.BuildPartB();
  }
}

/// <summary>

/// The 'Builder' abstract class

/// </summary>

abstract class Builder

{
  public abstract void BuildPartA();
  public abstract void BuildPartB();
  public abstract Product GetResult();
}

/// <summary>

/// The 'ConcreteBuilder1' class
```

```
class ConcreteBuilder1 : Builder

{
  private Product _product = new Product();

  public override void BuildPartA()
  {
    _product.Add("PartA");
  }

  public override void BuildPartB()
  {
    _product.Add("PartB");
  }

  public override Product GetResult()
  {
    return _product;
  }
}

/// <summary>

/// The 'ConcreteBuilder2' class

/// </summary>
```

```
class ConcreteBuilder2 : Builder

{
  private Product _product = new Product();

  public override void BuildPartA()
  {
    _product.Add("PartX");
  }

  public override void BuildPartB()
  {
    _product.Add("PartY");
  }

  public override Product GetResult()
  {
    return _product;
  }
}

/// <summary>

/// The 'Product' class

/// </summary>
```

```
  class Product

  {
    private List<string> _parts = new List<string>();

    public void Add(string part)
    {
      _parts.Add(part);
    }

    public void Show()
    {
      Console.WriteLine("\nProduct Parts -------");
      foreach (string part in _parts)
        Console.WriteLine(part);
    }
  }
}
```

6. **Out Put**

```
Product Parts -------
PartA
PartB

Product Parts -------
PartX
PartY
```

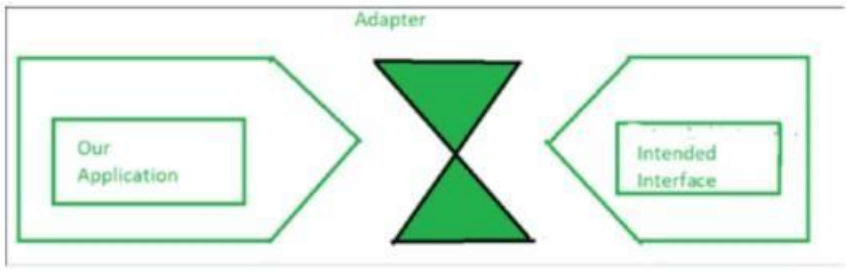
II. Adapter pattern

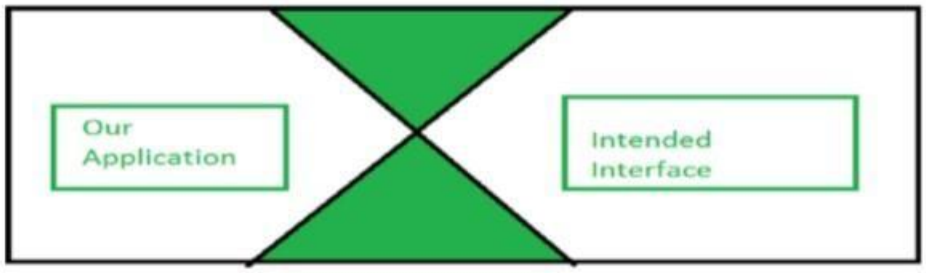
1. Definition

Convert a class's interface to a different interface that clients predict. The Adapter pattern allows classes that would not otherwise be able to work together due to incompatible interfaces to do so.

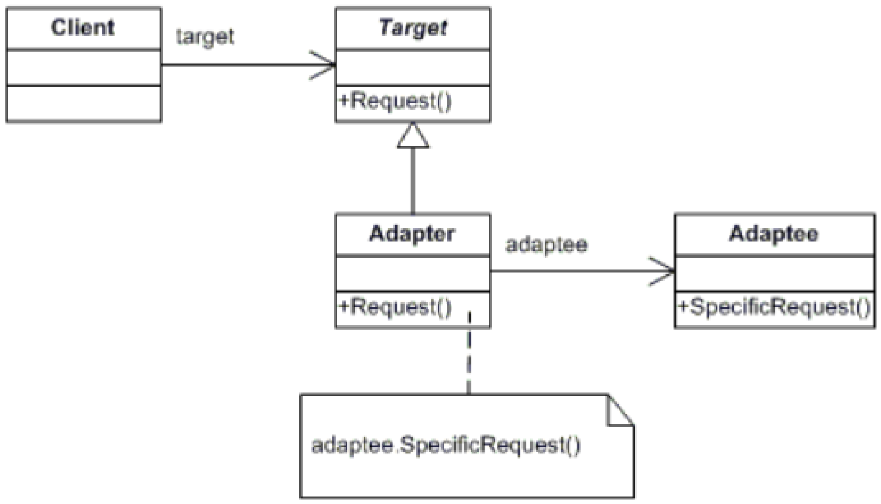
2. Real-Life Example

When traveling internationally, this pattern is commonly used when using an electrical outlet adapter/AC power adapter. These adapters may serve as intermediaries, allowing an electronic device, such as a laptop, to be plugged into a European power outlet. Consider the following scenario. Let's say you need to charge your cell. However, you note that the outlet is incompatible with your battery. You will need to use an adapter in this situation. In real life, even a translator who is translating one language to another can be said to be following this trend.
Consider a situation in which you must insert an application into an adapter (in this case, an Xshaped adapter) in order to use the intended device. You can't connect the application and the gui without this adapter.

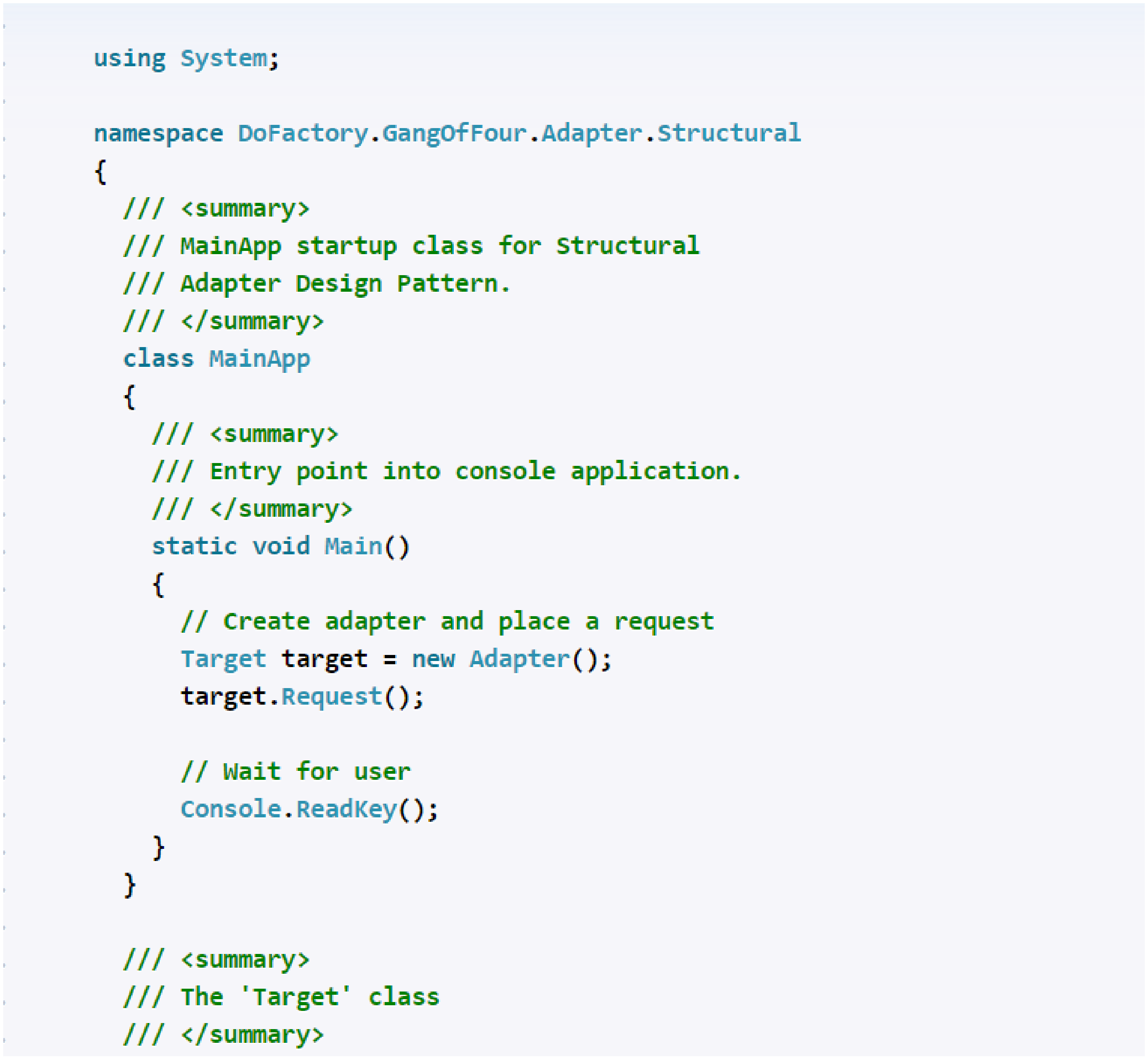




3. Class Adapter



4. Implementation



```
using System;

namespace DoFactory.GangOfFour.Adapter.Structural
{
  /// <summary>
  /// MainApp startup class for Structural
  /// Adapter Design Pattern.
  /// </summary>
  class MainApp
  {
    /// <summary>
    /// Entry point into console application.
    /// </summary>
    static void Main()
    {
      // Create adapter and place a request
      Target target = new Adapter();
      target.Request();

      // Wait for user
      Console.ReadKey();
    }
  }

  /// <summary>
  /// The 'Target' class
  /// </summary>
```

```
class Target
{
  public virtual void Request()
  {
    Console.WriteLine("Called Target Request()");
  }
}

/// <summary>
/// The 'Adapter' class
/// </summary>
class Adapter : Target
{
  private Adaptee _adaptee = new Adaptee();

  public override void Request()
  {
    // Possibly do some other work
    //  and then call SpecificRequest
    _adaptee.SpecificRequest();
  }
}
```

```
    /// <summary>
    /// The 'Adaptee' class
    /// </summary>
    class Adaptee
    {
      public void SpecificRequest()
      {
        Console.WriteLine("Called SpecificRequest()");
      }
    }
  }
```

5. **Out put**

```
Called SpecificRequest()
```

III. Observers pattern

1. Definition

Create a one-to-many relationship between objects such that when one object changes state, all of its dependents are immediately notified and modified.

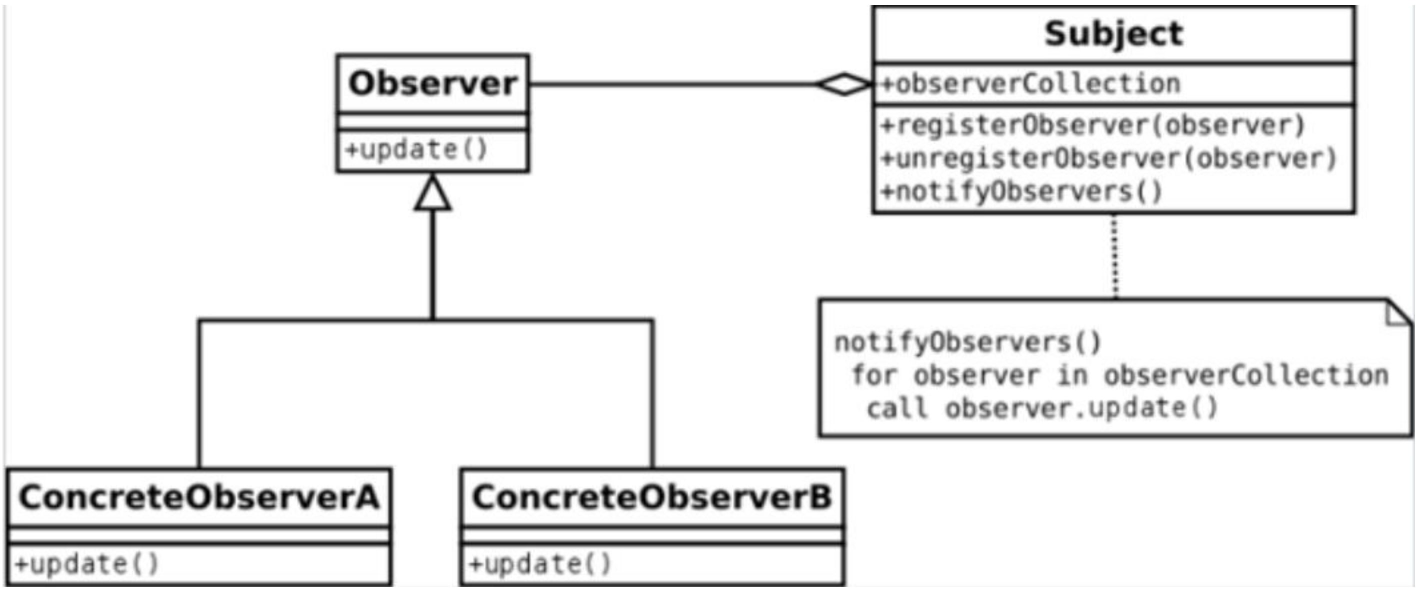
2. Concept

A large number of observers (objects) are observing a single subject in this pattern (also an object). When there is a shift in the topic, observers want to be informed. As a result, they enroll in that subject. They actually unregister from the topic when they lose interest in it. The PublisherSubscriber model is another name for this model. The following is a summary of the entire concept: An object (subject) can send notifications to multiple observers (a group of objects) at the same time using this pattern.

3. Real-Life Example

Consider a celebrity with a large social media following. Each of these fans needs to be kept up to date on their favorite celebrity's activities. As a result, they continue to pursue the celebrity until their interest dwindles. When they lose interest in a celebrity, they actually stop following them. Consider each of these admirers or followers to be an observer, and the celebrity to be the focus.

4. Class diagram



5. Implementation

```
public interface ISubject {
    void registerObserver(Observer observer);
    void unregisterObserver(Observer observer);
    void notifyObservers();
}
```

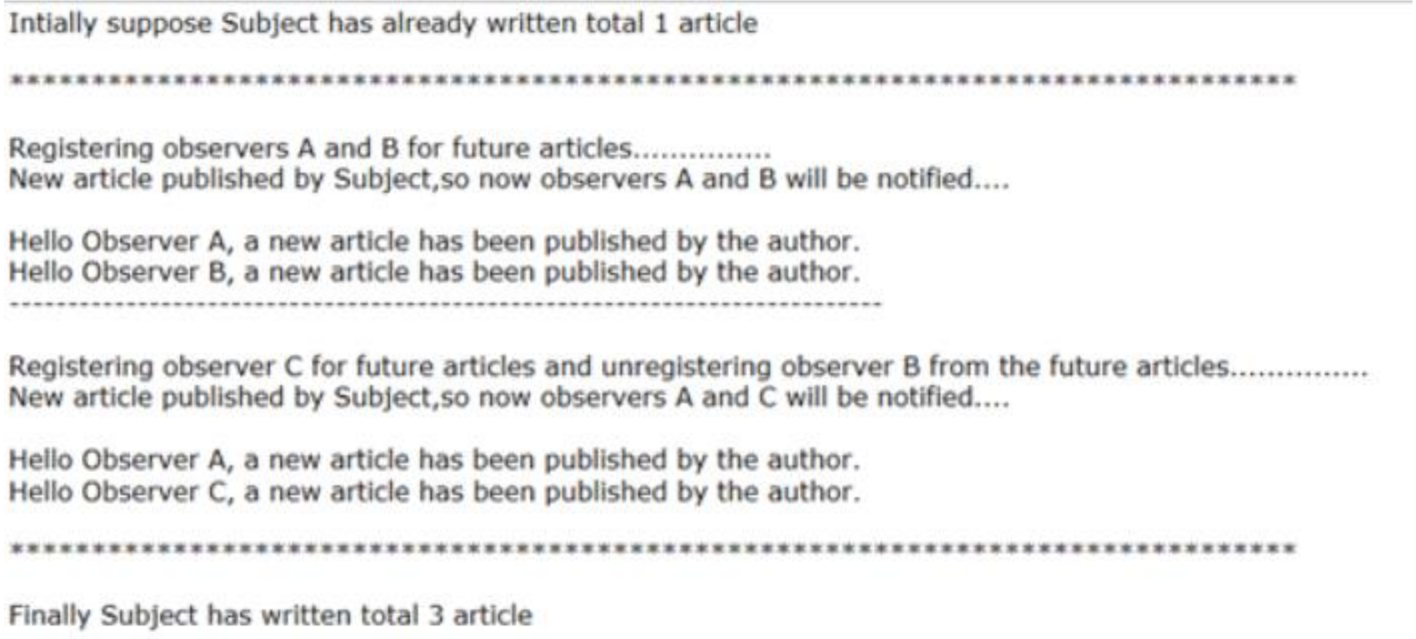
```
public class Subject: ISubject {
    private List < Observer > Observers = new List < Observer > ();
    private int articlesCount = 1;
    public int Articles {
        get {
            return articlesCount;
        }
        set {
            if (value > articlesCount) {
                articlesCount++;
                notifyObservers();
            }
        }
    }
    public void registerObserver(Observer observer) {
        Observers.Add(observer);
    }
    public void unregisterObserver(Observer observer) {
        Observers.Remove(observer);
    }
    public void notifyObservers() {
        foreach(var observer in Observers) {
            observer.Update();
        }
    }
}
```

```
public interface IObserver {
    void Update();
}
```

```
public class Observer: IObserver {
    public string ObserverName;
    public Observer(string name) {
        ObserverName = name;
    }
    public void Update() {
        //Observer can update his system accordingly
        Console.WriteLine("Hello " + ObserverName + ", a new article has been publishe
    }
}
```

```
void Main() {
    var subject = new Subject();
    var observerA = new Observer("Observer A");
    var observerB = new Observer("Observer B");
    var observerC = new Observer("Observer C");
    Console.WriteLine("Intially suppose Subject has already written total " + subject.
    Console.WriteLine("\n**********************************************************
    Console.WriteLine("Registering observers A and B for future articles.............
    subject.registerObserver(observerA);
    subject.registerObserver(observerB);
    Console.WriteLine("New article published by Subject,so now observers A and B will
    subject.Articles++;
    Console.WriteLine("--------------------------------------------------------------
-------------\n");
    Console.WriteLine("Registering observer C for future articles and unregistering ob
    subject.registerObserver(observerC);
    subject.unregisterObserver(observerB);
    Console.WriteLine("New article published by Subject,so now observers A and C will
    subject.Articles++;
    Console.WriteLine("\n**********************************************************
    Console.WriteLine("Finally Subject has written total " + subject.Articles + " arti
}
```

6. Out put


```
Intially suppose Subject has already written total 1 article

******************************************************************************

Registering observers A and B for future articles...............
New article published by Subject,so now observers A and B will be notified....

Hello Observer A, a new article has been published by the author.
Hello Observer B, a new article has been published by the author.
------------------------------------------------------------------------

Registering observer C for future articles and unregistering observer B from the future articles...............
New article published by Subject,so now observers A and C will be notified....

Hello Observer A, a new article has been published by the author.
Hello Observer C, a new article has been published by the author.

******************************************************************************

Finally Subject has written total 3 article
```

IV. Iterator Pattern

1. Definition

Provide a method for sequentially accessing the elements of an aggregate entity without revealing its underlying representation.

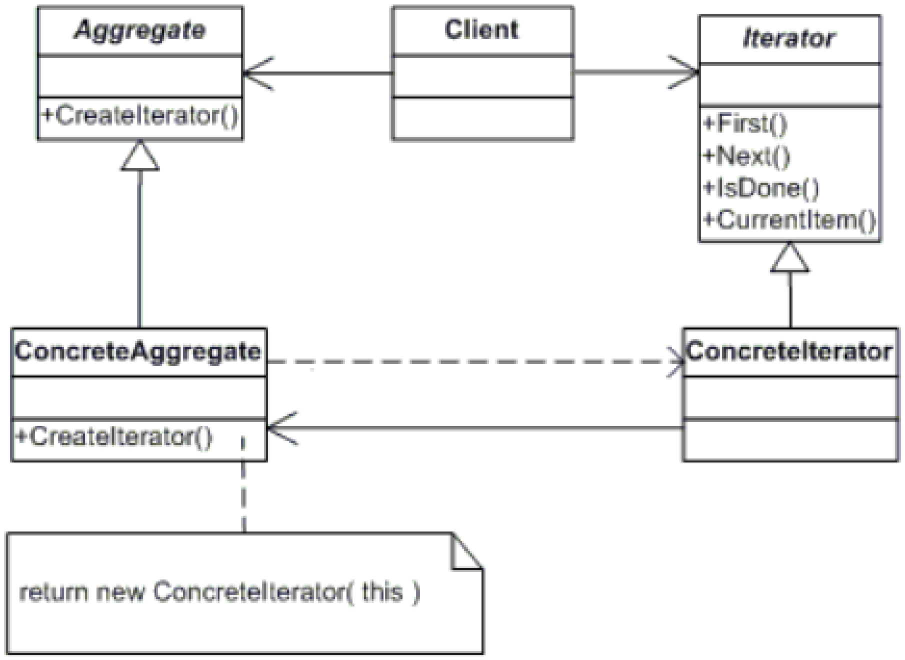
2. Concept

Iterators are commonly used to traverse a container (which is essentially an object) to reach its components, but you don't have to deal with the internal information of the element. Iterators are commonly used to traverse various types of collection items in a standard and uniform manner. This term is often frequently used to traverse the nodes of a tree-like structure. As a result, you can see the Iterator pattern combined with the Composite pattern in several scenarios.

3. Real-Life Example

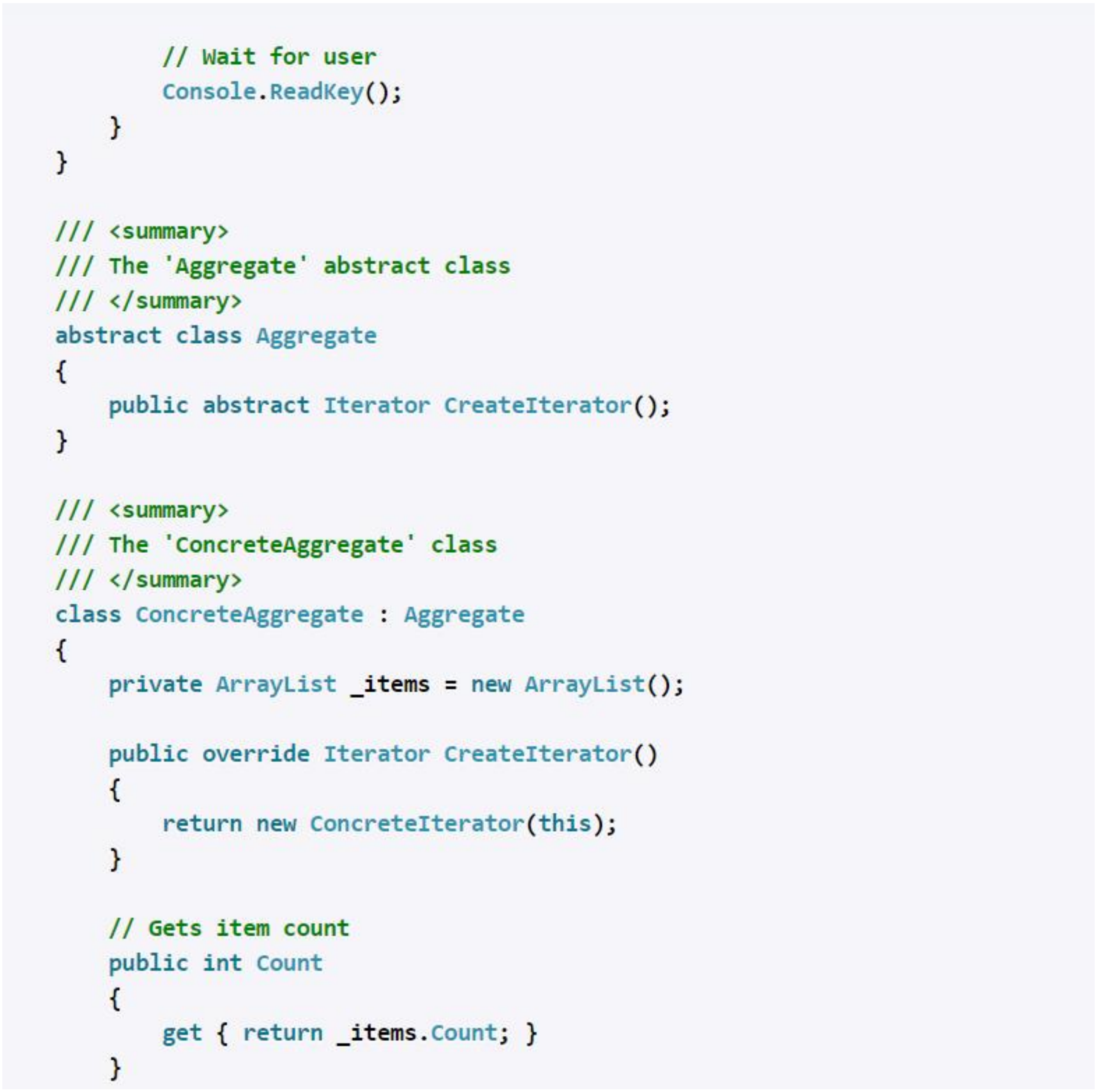
Assume Company A and Company B are two separate businesses.
Company A uses a linked list data structure to store its employee records (such as an employee's name, address, salary information, and so on),
while Company B uses an array to store its employee data. The two firms plan to combine one day. In this case, the Iterator pattern comes in handy because you don't have to write any code from scratch. In this case, you can create a similar interface that allows you to access data from both companies. As a result, instead of rewriting the code, you can simply call those methods.

4. Class diagram



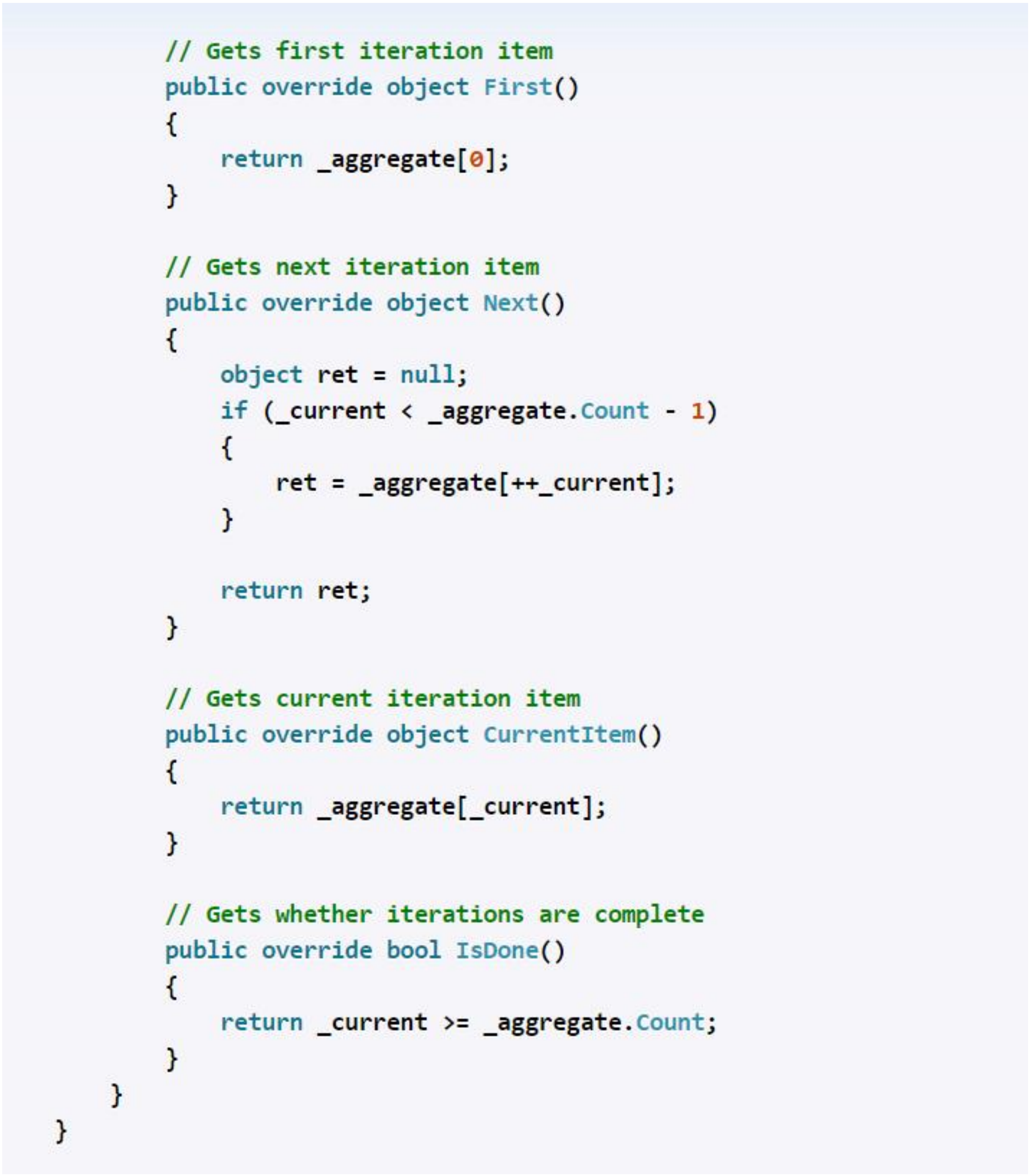
5. Implementation

```
using System;
using System.Collections;

namespace DoFactory.GangOfFour.Iterator.Structural
{
    /// <summary>
    /// MainApp startup class for Structural
    /// Iterator Design Pattern.
    /// </summary>
    class MainApp
    {
        /// <summary>
        /// Entry point into console application.
        /// </summary>
        static void Main()
        {
            ConcreteAggregate a = new ConcreteAggregate();
            a[0] = "Item A";
            a[1] = "Item B";
            a[2] = "Item C";
            a[3] = "Item D";

            // Create Iterator and provide aggregate
            Iterator i = a.CreateIterator();

            Console.WriteLine("Iterating over collection:");

            object item = i.First();
            while (item != null)
            {
                Console.WriteLine(item);
                item = i.Next();
            }
```



```
        // Wait for user
        Console.ReadKey();
    }
}

/// <summary>
/// The 'Aggregate' abstract class
/// </summary>
abstract class Aggregate
{
    public abstract Iterator CreateIterator();
}

/// <summary>
/// The 'ConcreteAggregate' class
/// </summary>
class ConcreteAggregate : Aggregate
{
    private ArrayList _items = new ArrayList();

    public override Iterator CreateIterator()
    {
        return new ConcreteIterator(this);
    }

    // Gets item count
    public int Count
    {
        get { return _items.Count; }
    }
```

```
        // Indexer
        public object this[int index]
        {
            get { return _items[index]; }
            set { _items.Insert(index, value); }
        }
    }

    /// <summary>
    /// The 'Iterator' abstract class
    /// </summary>
    abstract class Iterator
    {
        public abstract object First();
        public abstract object Next();
        public abstract bool IsDone();
        public abstract object CurrentItem();
    }

    /// <summary>
    /// The 'ConcreteIterator' class
    /// </summary>
    class ConcreteIterator : Iterator
    {
        private ConcreteAggregate _aggregate;
        private int _current = 0;

        // Constructor
        public ConcreteIterator(ConcreteAggregate aggregate)
        {
            this._aggregate = aggregate;
        }
```



```
        // Gets first iteration item
        public override object First()
        {
            return _aggregate[0];
        }

        // Gets next iteration item
        public override object Next()
        {
            object ret = null;
            if (_current < _aggregate.Count - 1)
            {
                ret = _aggregate[++_current];
            }

            return ret;
        }

        // Gets current iteration item
        public override object CurrentItem()
        {
            return _aggregate[_current];
        }

        // Gets whether iterations are complete
        public override bool IsDone()
        {
            return _current >= _aggregate.Count;
        }
    }
}
```

6. Out Put

```
Iterating over collection:
Item A
Item B
Item C
Item D
```

V. Singleton pattern

1. Definition

Make sure a class only has one instance and give it a global access point.

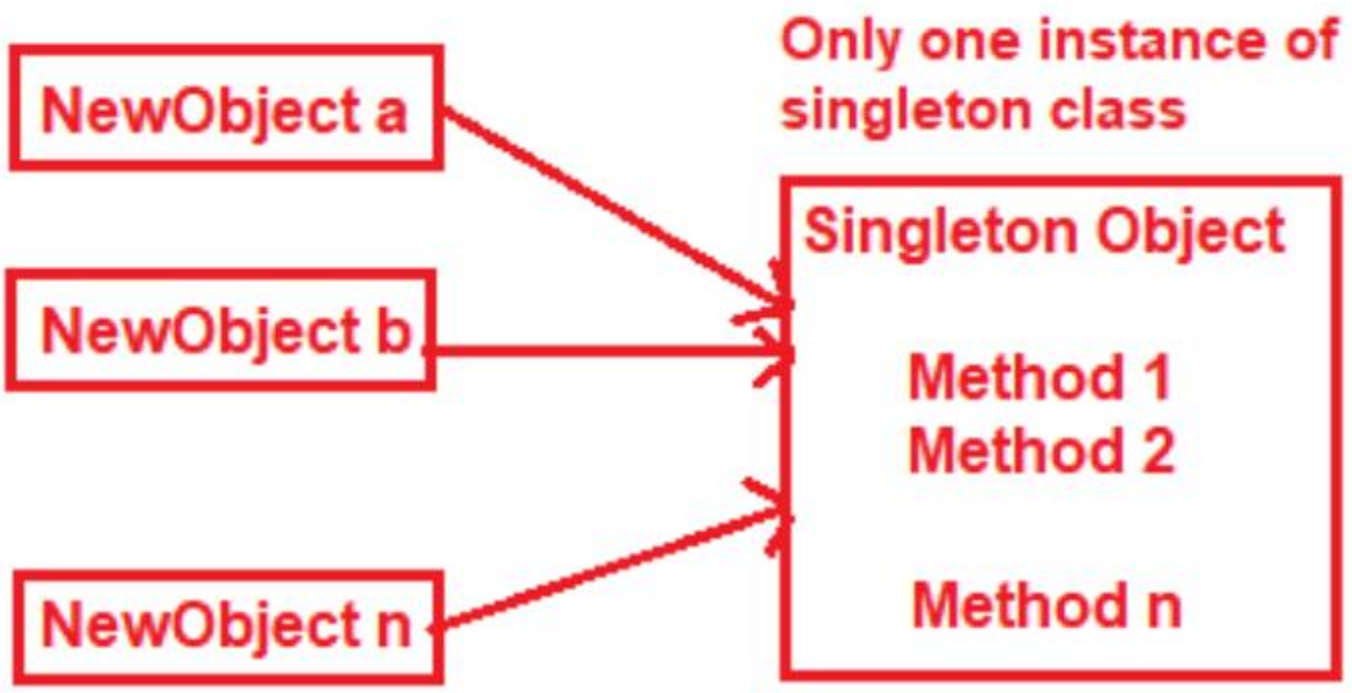
2. Concept

Only one instance of a class should exist. You can use this instance whenever you need it, avoiding the creation of redundant artifacts

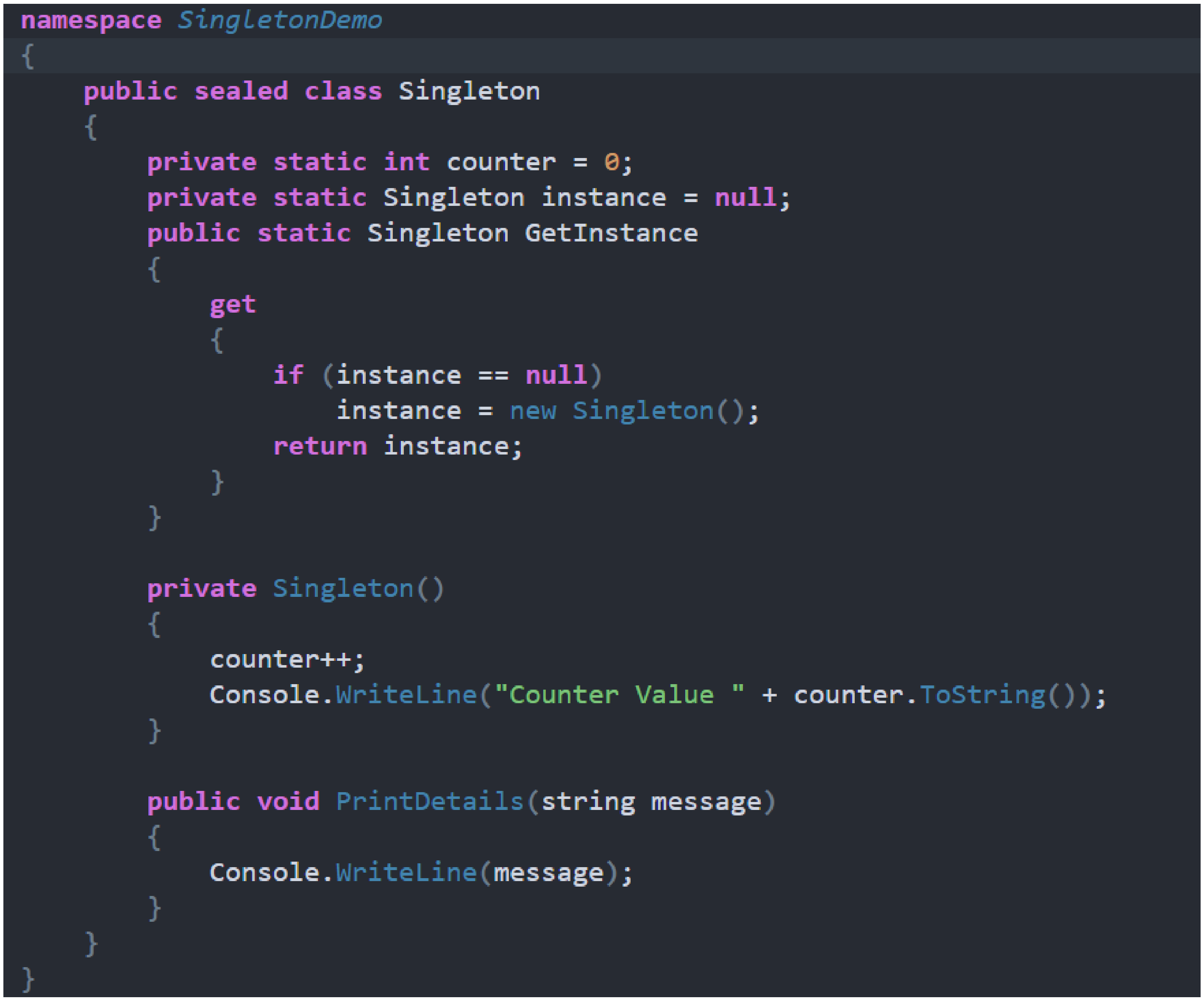
3. Real-Life Example

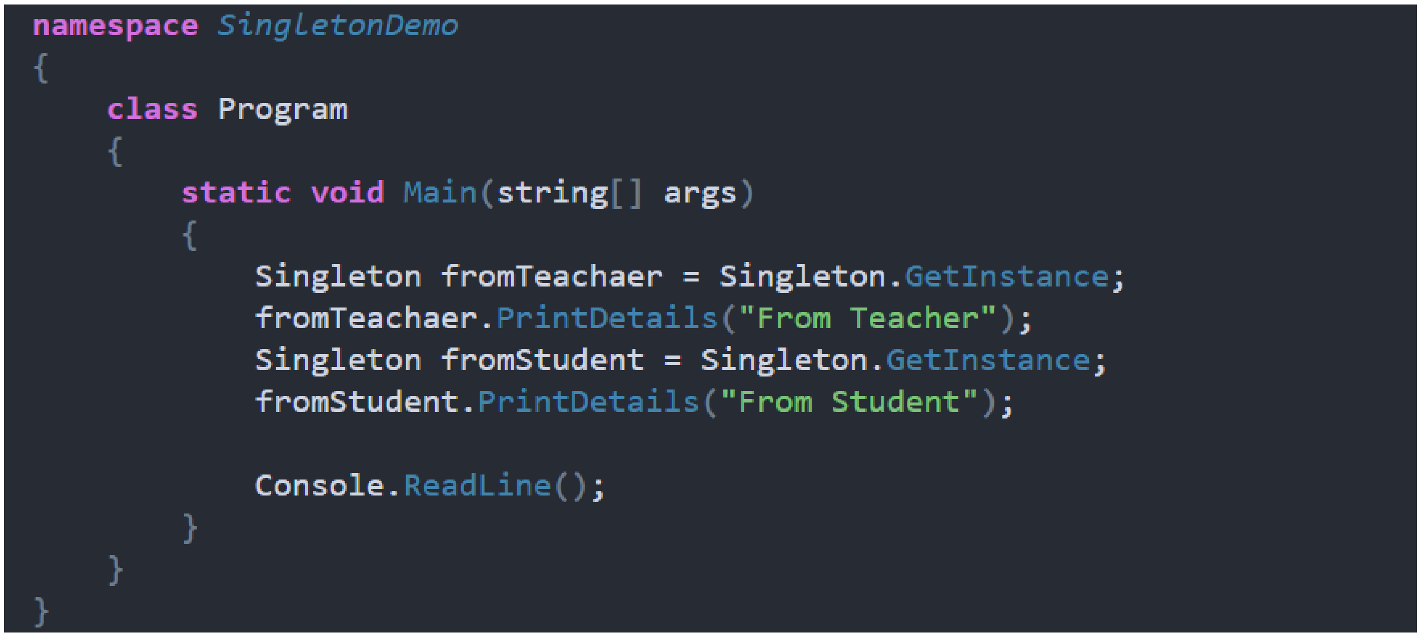
Assume that you are a member of a sports team that is competing in a competition. When your team plays against another team, the captains of both teams must have a coin toss, according to the rules of the game. If your team lacks a captain, you must first elect someone to serve as captain. There must be just one captain on your squad.

4. Class diagram



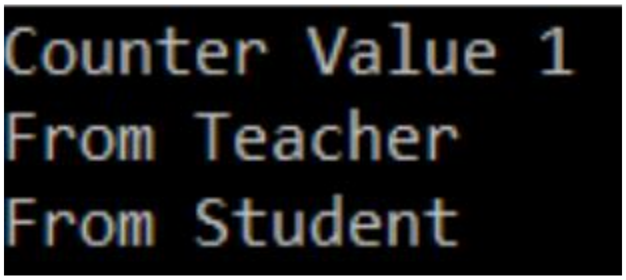
5. Implementation


```
namespace SingletonDemo
{
    public sealed class Singleton
    {
        private static int counter = 0;
        private static Singleton instance = null;
        public static Singleton GetInstance
        {
            get
            {
                if (instance == null)
                    instance = new Singleton();
                return instance;
            }
        }

        private Singleton()
        {
            counter++;
            Console.WriteLine("Counter Value " + counter.ToString());
        }

        public void PrintDetails(string message)
        {
            Console.WriteLine(message);
        }
    }
}
```



```
namespace SingletonDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            Singleton fromTeachaer = Singleton.GetInstance;
            fromTeachaer.PrintDetails("From Teacher");
            Singleton fromStudent = Singleton.GetInstance;
            fromStudent.PrintDetails("From Student");

            Console.ReadLine();
        }
    }
}
```

6. **Out Put**



```
Counter Value 1
From Teacher
From Student
```

REFERENCE:

Quatrani, T., 2000. *Visual modeling with Rational Rose 2000 and UML*. Addison-Wesley Professional.

Stevens, W.R. and Rago, S.A., 2008. *Advanced programming in the UNIX environment*. AddisonWesley.

Finkel, R.A. and Finkel, R.A., 1996. *Advanced programming language design* (p. 372). Reading: Addison-Wesley.

Sterling, L. and Shapiro, E.Y., 1994. *The art of Prolog: advanced programming techniques*. MIT press.