ASSIGNMENT 2 ADVANDENCE PROGRAM

Qualification	**TEC Level 5 HND Diploma in Computing**		
Unit number and title	**Unit 43: Internet of Things**		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Bui Dinh Kha	**Student ID**	GCH17468
Class	GCH0709	**Assessor name**	Doan Trung Tung
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	

Grading grid

P5	P6	P7	M5	M6	D2	D3

Table of Contents

Figure

I. Introduction.

From Assignment 1: My team has shown the effectiveness of UML diagrams in OOAD and introduced some Design Patterns in applications. Next I will give a demonstration of the use of OOAD and DP in a small matter, as well as an advanced discussion of the scope of the designs.

My group is currently separated and performing similar tasks in parallel. I chose Adapter of Structural Pattern, one of the actual scenarios that my team introduced about DP in the previous stage, then implemented that scenario based on the corresponding class diagram that the team created. I have modified the diagram so that it is possible for the implementation. In addition, I also discussed a series of related DPs and assessed them according to the scenario and proved my choice.

II. Introduction Structural Pattern

Structural Pattern (group of structures - 8 models): Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Private Class Data and Proxy. Design patterns of this type are related to the class and components of the object. It is used to establish and define relationships between objects.

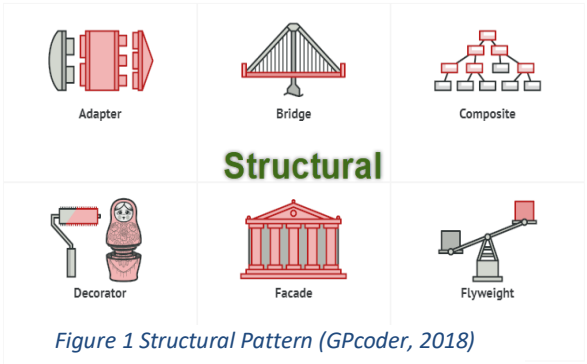

Figure 1 Structural Pattern (GPcoder, 2018)

1. **Adapter:**
   - Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.
   - Wrap an existing class with a new interface.
   - Impedance match an old component to a new system.
2. **Bridge:**
   - Decouple an abstraction from its implementation so that the two can vary independently.
   - Publish interface in an inheritance hierarchy, and bury implementation in its own inheritance hierarchy.
   - Beyond encapsulation, to insulation
3. **Composite:**
   - Compose objects into tree structures to represent whole-part hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
   - Recursive composition
   - "Directories contain entries, each of which could be a directory."
   - 1-to-many "has a" up the "is a" hierarchy
4. **Decorator:**
   - Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.
   - Client-specified embellishment of a core object by recursively wrapping it.

- Wrapping a gift, putting it in a box, and wrapping the box.

5. **Flyweight:**
   - Use sharing to support large numbers of fine-grained objects efficiently.
   - The Motif GUI strategy of replacing heavy-weight widgets with light-weight gadgets.
6. **Facade:**
   - Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.
   - Wrap a complicated subsystem with a simpler interface.
7. **Private Class Data:**
   - Control write access to class attributes
   - Separate data from methods that use it
   - Encapsulate class data initialization
   - Providing new type of final - final after constructor
8. **Proxy:**
   - Provide a surrogate or placeholder for another object to control access to it.
   - Use an extra level of indirection to support distributed, controlled, or intelligent access.
   - Add a wrapper and delegation to protect the real component from undue complexity.

III. Introduce the problem and choose the right model.

1. Problem.

Currently there is a mass of students who have just finished 5th grade of primary school getting ready to move to grade 6 of secondary school. The middle school wants to reuse the information of students from the elementary school such as Name, Student ID, Address, etc. tablets. The two schools have a new system that can connect to the two systems of the two schools and then transfer the data.

2. Solution.

Through the problem we can see that these are two systems with two different interfaces so data cannot be connected to each other. These are specific issues of the Structural Pattern. We will not need to change the structure of the two old systems, instead I will create a new object whose interface matches both interfaces of the two systems, so that we can retrieve data from the magnetic field elementary school and transfer it to high school.

From the above solutions I chose the Structural Pattern Adapter to connect between these two systems for the following reasons:

1. The two systems use two different interfaces but want to transfer data, so use the adapter model that can connect to two interfaces and transmit information.

2. Using the Adapter model will generate a new Class that mediates the two systems, this class is created for general use only and does not change the interface of the two systems of the system.

IV. Design the Diagram and build Scenario .

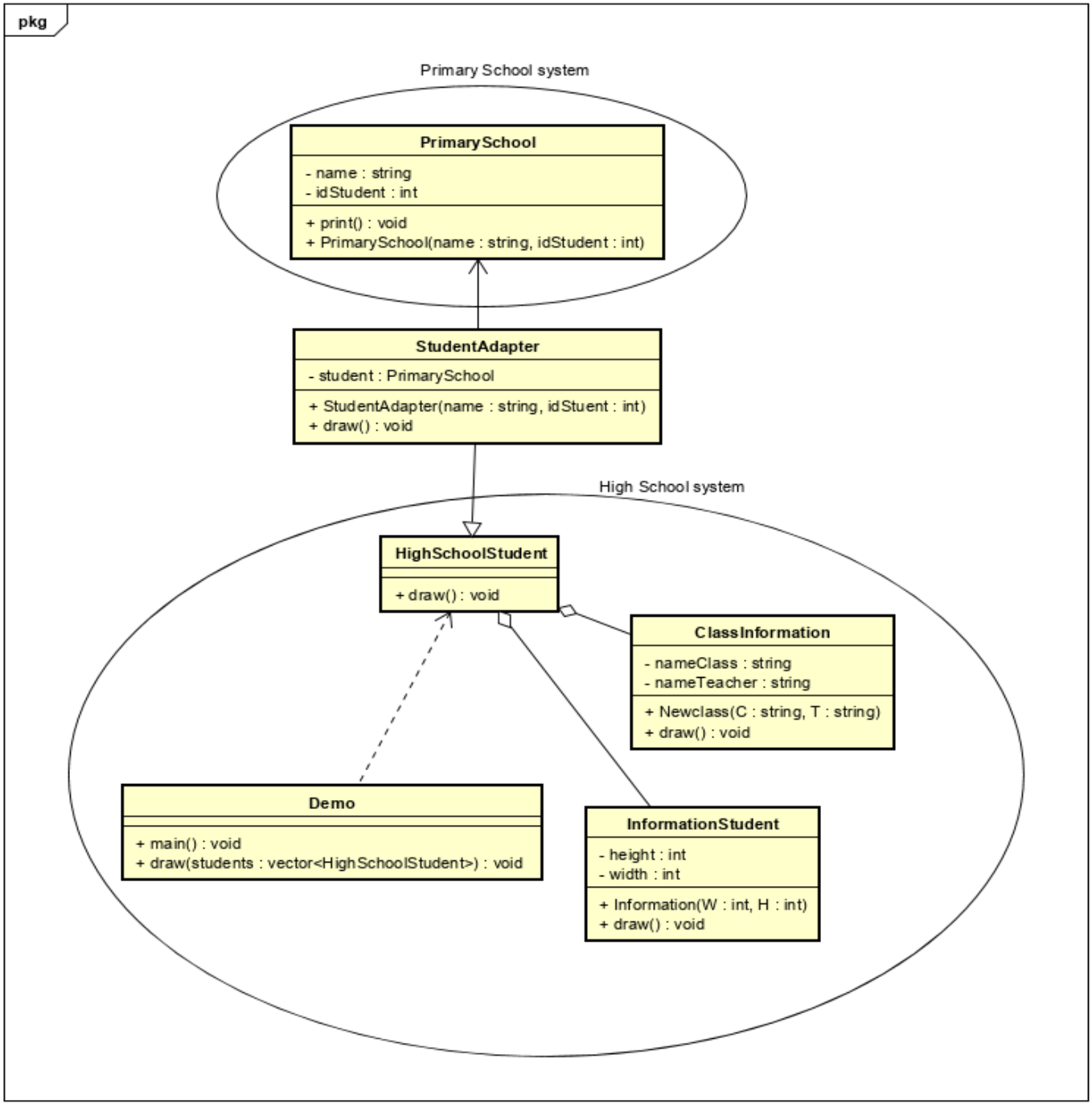

Figure 2.Diagrame of SchoolAdapter systerm

1. Explain the diagram.

My system consists of 6 tightly linked classes. In particular, Class PrimarySchool is the Class representing the interface of the Primary school. This includes student information such as Name, and ID.

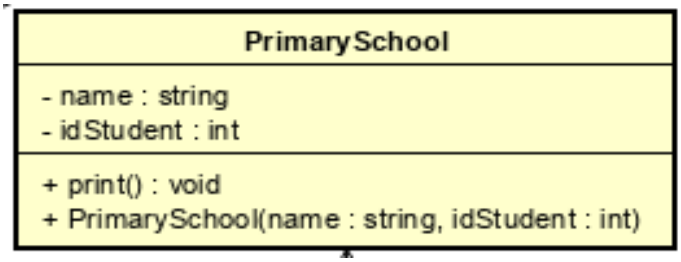

Figure 3. Diagram PrimarySchool

The Secondary school system consists of 4 classes: HighSchoool, Demo, ClassInformation, InformationStudent and Demo.

- Class ClassInformation: contains class information and the name of the class teacher.
- Class InformationStudent : contains new information of students starting secondary school including height and weight.
- Class HighSchoolStudent is a class that synthesizes all student information through the show () function; . The HighSchoolStudent class has an Aggregation relationship with two Class ClassInformation and InformationStudent.
- Class Demo: is used to display field information on the screen by linking with the HighSchoolStudent class.

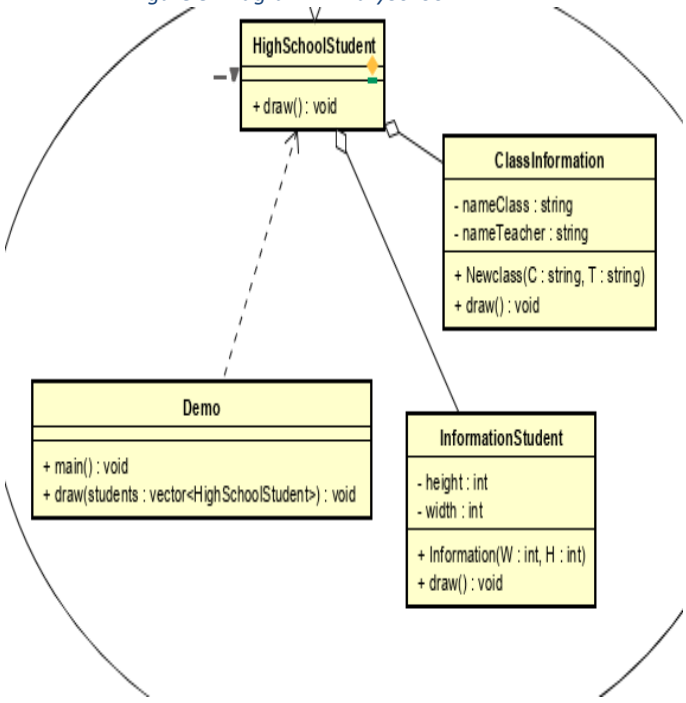

Figure 4. Secondary school system

The StudentAdapter Class is the most important class, it links data between two systems by inheriting the Primary school system (Inheritance - object) and linking to the High School system. From there, StudentAdapter can transfer student information from Primary to High School.

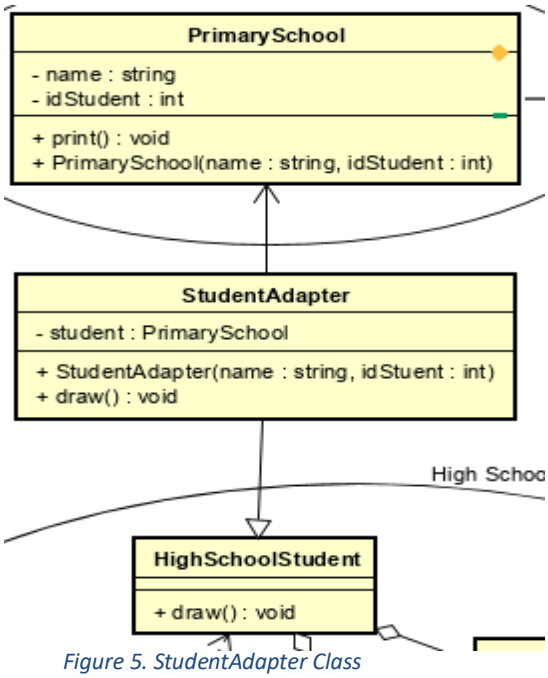

Figure 5. StudentAdapter Class

2. Develop code for SchoolAdapter project.

2.1.Primary school system.

First, I created the interface for the elementary school system including: Name and ID (PrimarySchool.h)

```
#ifndef PRIMARYSCHOOL_H
#define PRIMARYSCHOOL_H

#include <iostream>
#include <string>

using namespace std;

class PrimarySchool
{
private:
    string name;
    int idStudent;
public:
    PrimarySchool(const string &name, const int &idStudent);
    void print() const;
};


#endif // PRIMARYSCHOOL_H
```

Figure 6. Code interface of Primary School systerm

Install the interface and set up the objects in the class. (PrimarySchool.cpp)

```
#include "PrimarySchool.h"

PrimarySchool::PrimarySchool(const string &name, const int &idStudent)
{
    this->name = name;
    this->idStudent = idStudent;
}

void PrimarySchool::print() const
{
    int length = name.length();
    int x = 0;
        if(x < 17)
        {
            cout << "                  ";
        }
        if(17 < x <length + 20)
        {
           for (int i = 0; i < length + 2 ; i++)

            cout << "-";

        }
        cout << endl;
    cout << "Name of Student: " << "|" << name << "|"<< endl;

    if( x < 17 )
       {
            cout << "                  ";
       }
    if(17< x <length + 20)
        {
           for (int i = 0; i < length + 2 ; i++)
           {
            cout << "-";
           }
        }
        cout << endl;
   cout << "ID of Student: " << "|" << idStudent << "|"<< endl;
}
```

Figure 7. Install the interface and set up the objects in the class.

2.2.High school system.

Next I declared the interface of the high school, firstly Class HighSchoolStudent and ClassInformation.

The HighSchoolStudent class declares the draw (); function to aggregate information from 2 classes.

(HighSchoolStudent.h)

```
#ifndef HIGHSCHOOLSTUDENT_H
#define HIGHSCHOOLSTUDENT_H
#include <iostream>
#include <string>
#include "PrimarySchool.h"

using namespace std;
class HighSchoolStudent
{
public:
    virtual void draw() const = 0;
    virtual ~HighSchoolStudent();
};

class ClassInformation : public HighSchoolStudent
{
private:
    string nameClass;
    string nameTeacher;
public:
    ClassInformation(const string &nameClass, const string &nameTeacher);
    void draw() const;
};

class InformationStudent : public HighSchoolStudent
{
private:
    int width;
    int height;
public:
    InformationStudent(const int &w, const int &h);
    void draw() const;
private:
    void draw_base() const;
    void draw_side() const;
};

```

Figure 8. Code interface of High School systerm

In addition, the Class Adapter is also declared in the file so it can be linked to the High School system.

```
class HighSchoolAdapter : public HighSchoolStudent
{
private:
    PrimarySchool text;
public:
    HighSchoolAdapter(const string &name, const int &idStudent);
    void draw() const;
};
#endif // HIGHSCHOOLSTUDENT_H

```

Figure 9. Code interface of High School Adapter systerm

Declare the interface and setup objects for Class ClassInformation and InfromationStudent.

(HighSchoolStudent.cpp)

```cpp
#include "HighSchoolStudent.h"

ClassInformation::ClassInformation(const string &nameClass, const string &nameTeacher)
{
    this->nameClass = nameClass;
    this->nameTeacher = nameTeacher;
}

void ClassInformation::draw() const
{
     int a = nameClass.length();
    for (int i = 0; i < a + 25; i++)
    {
        cout << "+";
    }
    cout << endl;

    cout << "| Student's Classroom : " << nameClass <<"|" <<endl;
     for (int i = 0; i < a + 25; i++)
    {
        cout << "+";
    }
    cout << endl;
    int b = nameTeacher.length();
    for (int i = 0; i < b + 24; i++ )
    {
        cout << "-";
    }
    cout << endl;
    cout << "| Student's Teacher  : " << nameTeacher << "|"<< endl;
    for (int i = 0; i < b + 24; i++)
    {
        cout << "-";
    }
    cout << endl;
}
```

```cpp
InformationStudent::InformationStudent(const int &w, const int &h)
{
    width = w;
    height = h;
}

void InformationStudent::draw() const
{
    draw_base();
    draw_side();
}

void InformationStudent::draw_base() const
{
     cout << "-The height of that student is: "<< height <<" Kg"<< endl;
}

void InformationStudent::draw_side() const
{
    cout << "-The student's weight is:  "<< width <<"cm"<< endl;
}
```

Figure 10. Declare the interface and setup objects for Class ClassInformation and InfromationStudent

Finally, we have configured the Class Adapter link with the two systems.

First, the Adapter takes information of Class PrimarySchool objects and then assigns values to text (); . Then declare the function : draw (); with the command text.print (); .So the data has been passed to the HighSchoolStudent Class.

```
60  HighSchoolAdapter::HighSchoolAdapter(const string &name,const int &idStudent) : text(name, idStudent)
61  {
62  }
63
64  void HighSchoolAdapter::draw() const
65  {
66      text.print();
67  }
68
```

Figure 11. Configured the Class Adapter link with the two systems.

2.3.Display to screen.

```
1   #include<iostream>
2   #include<vector>
3   #include"HighSchoolStudent.h"
4   #include<stdio.h>
5
6   using namespace std;
7
8   void draw(vector<HighSchoolStudent*> student)
9   {
10      for (int i = 0; i < student.size(); i++)
11      {
12          student[i]->draw();
13      }
14  }
15
16  int main(int argc, const char * argv[]) {
17
18      string nameStudent;
19      string className;
20      string teacherName;
21      int height;
22      int width;
23      int idStudent;
24
25      cout << "Enter the information for Primary School" << endl;
26      cout << "Enter Name of Student: ";
27      getline(cin, nameStudent);
28      cout << "Enter ID of "<< nameStudent <<": ";
29      cin >> idStudent;
30      cout <<"\n------------Connected to the High School system------------\n"<< endl;
31      cout << "Enter the information for Hight School" << endl;
32      cout << "Enter Class of "<<nameStudent<<": ";
33      getline(cin, className);
34      getchar();
35      cout << "Enter Teacher of "<<nameStudent<<":";
36      getline(cin, teacherName);
37      getchar();
```

```
35      cout << "Enter Teacher of "<<nameStudent<<":";
36      getline(cin, teacherName);
37      getchar();
38      cout << "Enter Height of "<<nameStudent<<": ";
39      cin >> height;
40      cout << "Enter Width of "<<nameStudent<<": ";
41      cin >> width;
42      cout <<"\n----------------------------------------\n"<< endl;
43      HighSchoolAdapter *studentPrimary = new HighSchoolAdapter(nameStudent, idStudent);
44      InformationStudent *studentHigh = new InformationStudent(height, width);
45      ClassInformation *classInfor = new ClassInformation(className, teacherName);
46
47      vector<HighSchoolStudent*> student;
48      student.push_back(studentPrimary);
49      student.push_back(studentHigh);
50      student.push_back(classInfor);
51
52      draw(student);
53      cout <<"\n-------Fished Enter the information student "<<nameStudent<<"-------\n"<< endl;
54
55      for (int i = 0; i < student.size(); i++)
56          delete student[i];
57
58      return 0;
59  }
60
```

Figure 12. Display the results from main.cpp

3. Showing results.

1. The interface of an primary school includes its name and ID.

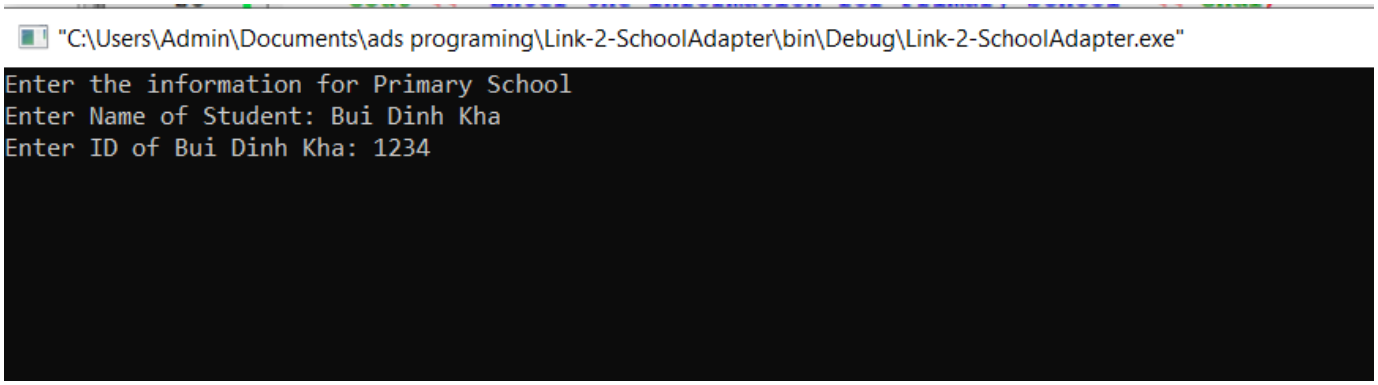

Figure 13. The interface of an elementary school

2. The interface of the high school includes entering the class name, teacher name, height and weight of the student.

```
------------Connected to the High School system-------------

Enter the information for Hight School
Enter Class of Bui Dinh Kha: GCH0709
Enter Teacher of Bui Dinh Kha: Doan Trung Tung
Enter Height of Bui Dinh Kha: 60
Enter Width of Bui Dinh Kha: 172

-----------------------------------------
```

Figure 14. The interface of an elementary school

3. Displaying results after the two systems link and pass data to each other.

```
Enter the information for Primary School
Enter Name of Student: Bui Dinh Kha
Enter ID of Bui Dinh Kha: 1999

------------Connected to the High School system-------------

Enter the information for Hight School
Enter Class of Bui Dinh Kha: GCH0709
Enter Teacher of Bui Dinh Kha: Doan Trung Tung
Enter Height of Bui Dinh Kha: 60
Enter Width of Bui Dinh Kha: 172

-----------------------------------------


                --------------
Name of Student: |Bui Dinh Kha|
                --------------
ID of Student: |1999|
-The height of that student is: 60 Kg
-The student's weight is:  172cm
++++++++++++++++++++++++++++++++
| Student's Classroom : GCH0709|
++++++++++++++++++++++++++++++++
--------------------------------------
| Student's Teacher  : Doan Trung Tung|
--------------------------------------

-------Fished Enter the information student Bui Dinh Kha-------
```

Figure 15. The results after aply Adapter

References

GPcoder, 2018. *GPcoder.* [Online]
Available at: https://gpcoder.com/4164-gioi-thieu-design-patterns/#Nhom_Structural_nhom_cau_truc

Dreamtime. (2019). Retrieved from
https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjg6JyDqc7lAhVZE4gKHSgnB58QjRx6BAgBEAQ&url=https%3A%2F%2Fwww.dreamstime.com%2Froyalty-free-stock-photography-phone-its-functions-three-phones-different-icons-image36183247&psig=AOv

Patterns, D. (2019). Retrieved from https://sourcemaking.com/design_patterns/creational_patterns

Puri, D. (2019). Retrieved from https://www.quora.com/What-are-the-advantages-of-OOP

❒ Summative Feedback:	❒ Resubmission Feedback:

Grade:	Assessor Signature:	Date:

Internal Verifier's Comments:

Signature & Date: