## **Class and Object**

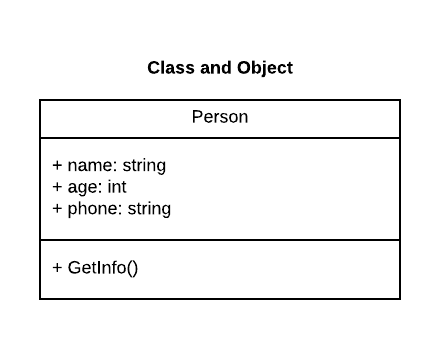


Figure . Class and Object class diagram

* **Source code of Class and Object:**

class Persons

{

public string name;

public int age;

public string phone;

public Persons()

{

name = "Empty";

age = 0;

}

public Persons(string Name)

{

name = Name;

age = 0;

phone = "0";

}

public Persons(string Name, int Age)

{

name = Name;

age = Age;

phone = "0";

}

public Persons(string Name, int Age, string Phone)

{

name = Name;

age = Age;

phone = Phone;

}

public void setName(string Name)

{

name = Name;

}

public void setAge(int Age)

{

age = Age;

}

public void setPhone(string Phone)

{

phone = Phone;

}

public void getInfo()

{

Console.WriteLine($"{name} is {age} years old and has phone number: {phone}");

}

* **Code Main function of Class and Object:**

class Program

{

static void Main(string[] args)

{

Persons myPersons = new Persons();

myPersons.setName("Quang Huy");

myPersons.setAge(24);

myPersons.setPhone("0795541090");

myPersons.getInfo();

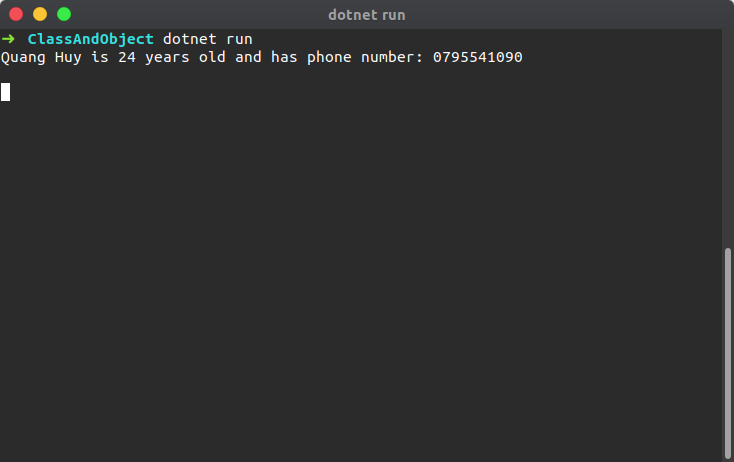
Console.WriteLine();

Console.ReadKey();

}

}

* **Result:**

****

Picture . Result of Class and Object code

## **Encapsulation**

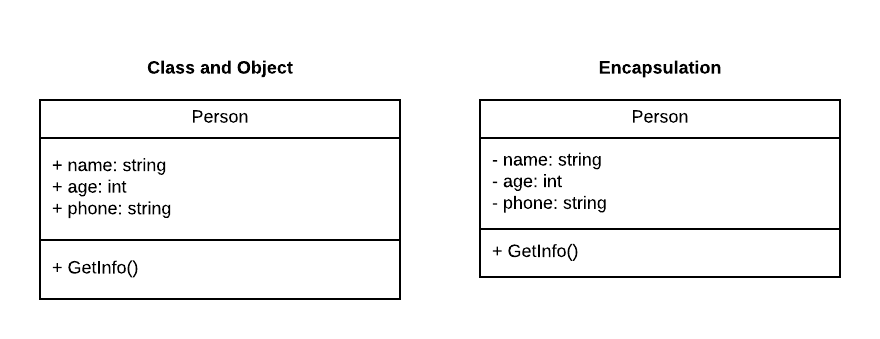
* **Class Diagram of Encapsulation:**

Figure . Encapsulation class diagram

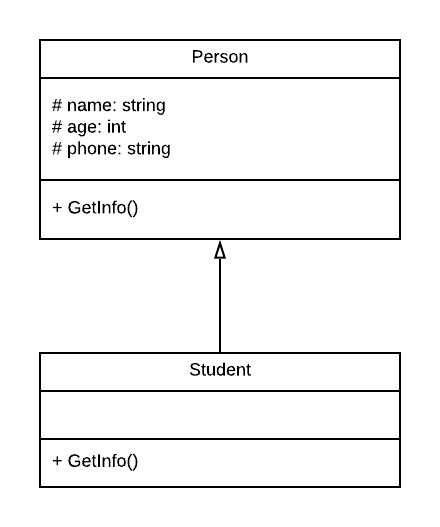


Figure . Protected in Encapsulation class diagram

* **Source code of Encapsulation:**

class Persons

{

private string name;

private int age;

private string phone;

public string Name

{

get { return name; }

set { name = value; }

}

public int Age

{

get { return age; }

set { age = value; }

}

public string Phone

{

get { return phone; }

set { phone = value; }

}

public void getInfo()

{

Console.WriteLine($"{Name} is {Age} years old and has phone number: {Phone}");

}

}

* **Code Main function of Encapsulation:**

class Program

{

static void Main(string[] args)

{

Persons myPersons = new Persons();

myPersons.Name = "Quang Huy";

myPersons.Age = 24;

myPersons.Phone = "0795541090";

Console.Write($"{myPersons.Name} is {myPersons.Age} years old and has phone number: {myPersons.Phone}");

Console.WriteLine();

myPersons.getInfo();

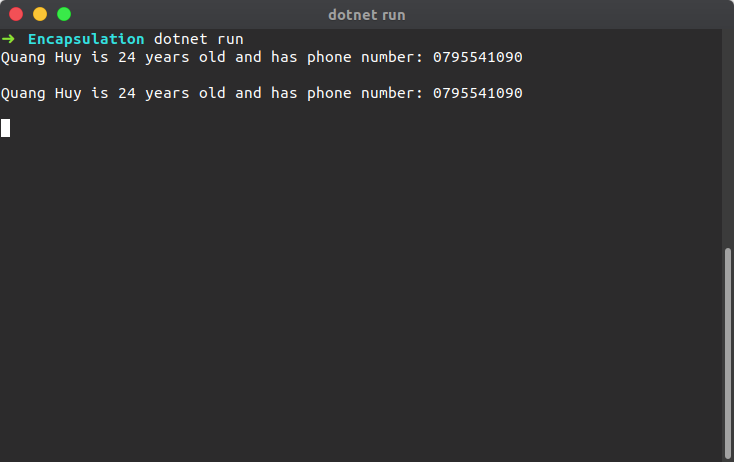
Console.WriteLine();

Console.ReadKey();

}

}

* **Result:**



Picture 2. Result of Encapsulation code

## 

## **Inheritance**

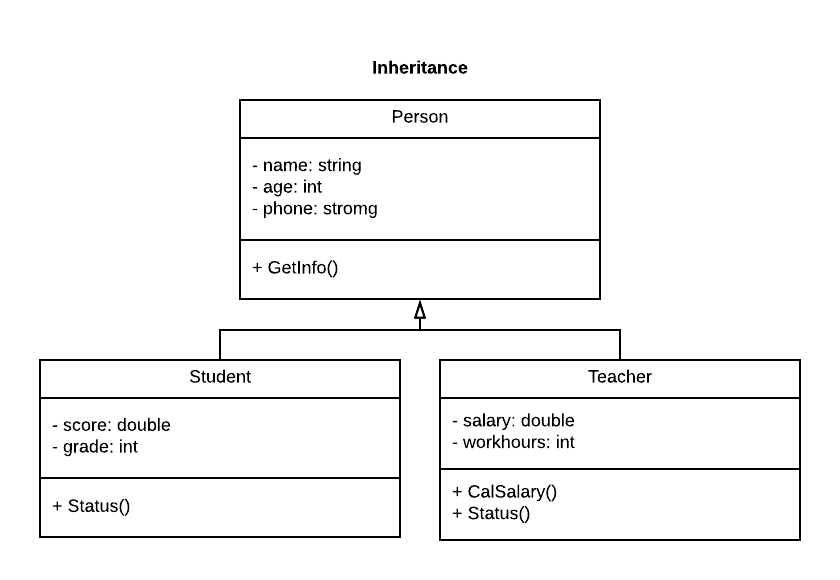
* **Class Diagram of Inheritance:**

Figure . Inheritance class diagram

**Source code of Inheritance:**

class Persons

{

private string name;

private int age;

private string phone;

public virtual string Name

{

get { return name; }

set { name = value; }

}

public virtual int Age

{

get { return age; }

set { age = value; }

}

public virtual string Phone

{

get { return phone; }

set { phone = value; }

}

public Persons(string name, int age, string phone)

{

this.Name = name;

this.Age = age;

this.Phone = phone;

}

}

class Student : Persons

{

private double score;

private int grade;

public Student(string name, int age, string phone) : base(name, age, phone)

{

score = 0;

grade = 0;

}

public Student(string name, int age, string phone, double score) : base(name, age, phone)

{

this.score = score;

}

public Student(string name, int age, string phone, double score, int grade) : base(name, age, phone)

{

this.score = score;

this.grade = grade;

}

public void status()

{

if (score >= 5)

Console.Write($"Infomation of student {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tScore: {score}\n\tMove to next grade({grade} => {grade + 1})\n\n");

else Console.Write($"Infomation of student {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tScore: {score}\n\tCant move to next grade({grade} => {grade + 1})\n\n");

}

}

class Teacher : Persons

{

private double salary;

private int workhours;

public Teacher(string name, int age, string phone) : base(name, age, phone)

{

salary = 0;

workhours = 0;

}

public Teacher(string name, int age, string phone, double salary) : base(name, age, phone)

{

this.salary = salary;

workhours = 0;

}

public Teacher(string name, int age, string phone, double salary, int workhours) : base(name, age, phone)

{

this.salary = salary;

this.workhours = workhours;

}

public int CalSalary(int workhours)

{

return 10 \* workhours;

}

public void status()

{

Console.Write($"Infomation of teacher {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tSalary: {salary}\n\tWork hours: {workhours}\n\tSalary get: {CalSalary(workhours)}\n");

}

}

**Code Main function of Inheritance:**

class Program

{

static void Main(string[] args)

{

Student myStudent1 = new Student("Quang Huy", 24, "0795541090", 9, 10);

Student myStudent2 = new Student("Hieu Nguyen", 20, "012333232", 4, 10);

Teacher myTeacher1 = new Teacher("Vinh Hoang", 28, "0987654321", 400, 48);

myStudent1.status();

myStudent2.status();

myTeacher1.status();

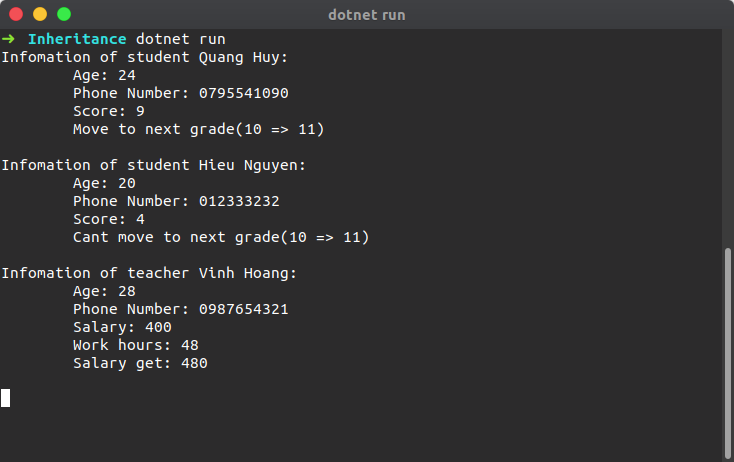
Console.WriteLine();

Console.ReadKey();

}

}

**Result:**

****

Picture 3. Result of Inheritance code

## **Polymorphism**

**Class Diagram of Polymorphism:**

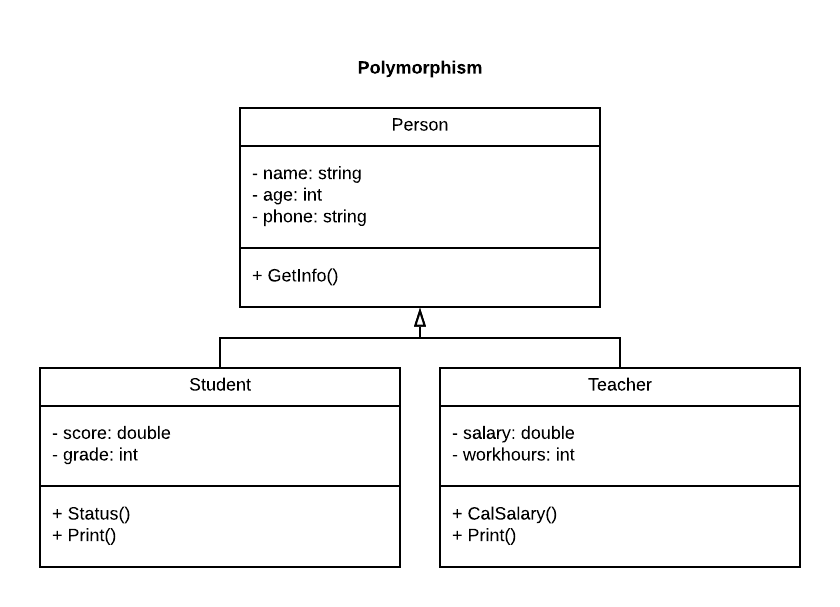
****

Figure 5. Polymorphism class diagram

**Source code of Polymorphism:**

class Persons

{

private string name;

private int age;

private string phone;

public virtual string Name

{

get { return name; }

set { name = value; }

}

public virtual int Age

{

get { return age; }

set { age = value; }

}

public virtual string Phone

{

get { return phone; }

set { phone = value; }

}

public Persons(string name, int age, string phone)

{

this.Name = name;

this.Age = age;

this.Phone = phone;

}

}

class Student : Persons

{

private double score;

private int grade;

public Student(string name, int age, string phone) : base(name, age, phone)

{

score = 0;

grade = 0;

}

public Student(string name, int age, string phone, double score) : base(name, age, phone)

{

this.score = score;

}

public Student(string name, int age, string phone, double score, int grade) : base(name, age, phone)

{

this.score = score;

this.grade = grade;

}

public void status()

{

if (score >= 5)

Console.Write($"Infomation of student {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tScore: {score}\n\tMove to next grade({grade} => {grade + 1})\n\n");

else Console.Write($"Infomation of student {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tScore: {score}\n\tCant move to next grade({grade} => {grade + 1})\n\n");

}

}

class Teacher : Persons

{

private double salary;

private int workhours;

public Teacher(string name, int age, string phone) : base(name, age, phone)

{

salary = 0;

workhours = 0;

}

public Teacher(string name, int age, string phone, double salary) : base(name, age, phone)

{

this.salary = salary;

workhours = 0;

}

public Teacher(string name, int age, string phone, double salary, int workhours) : base(name, age, phone)

{

this.salary = salary;

this.workhours = workhours;

}

public int CalSalary(int workhours)

{

return 10 \* workhours;

}

public void status()

{

Console.Write($"Infomation of teacher {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tSalary: {salary}\n\tWork hours: {workhours}\n\tSalary get: {CalSalary(workhours)}\n");

}

}

**Code Main function of Polymorphism:**

class Program

{

static void Main(string[] args)

{

Student myStudent1 = new Student("Quang Huy", 24, "0795541090", 9, 10);

Student myStudent2 = new Student("Hieu Nguyen", 20, "012333232", 4, 10);

Teacher myTeacher1 = new Teacher("Vinh Hoang", 28, "0987654321", 400, 48);

myStudent1.status();

myStudent2.status();

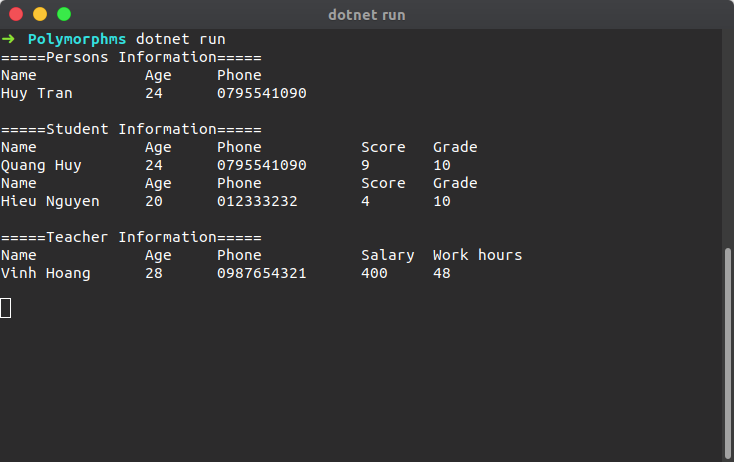
myTeacher1.status();

Console.WriteLine();

Console.ReadKey();

}

}

**Result:**

Picture . Result of Polymorphism code

## **Abstraction**

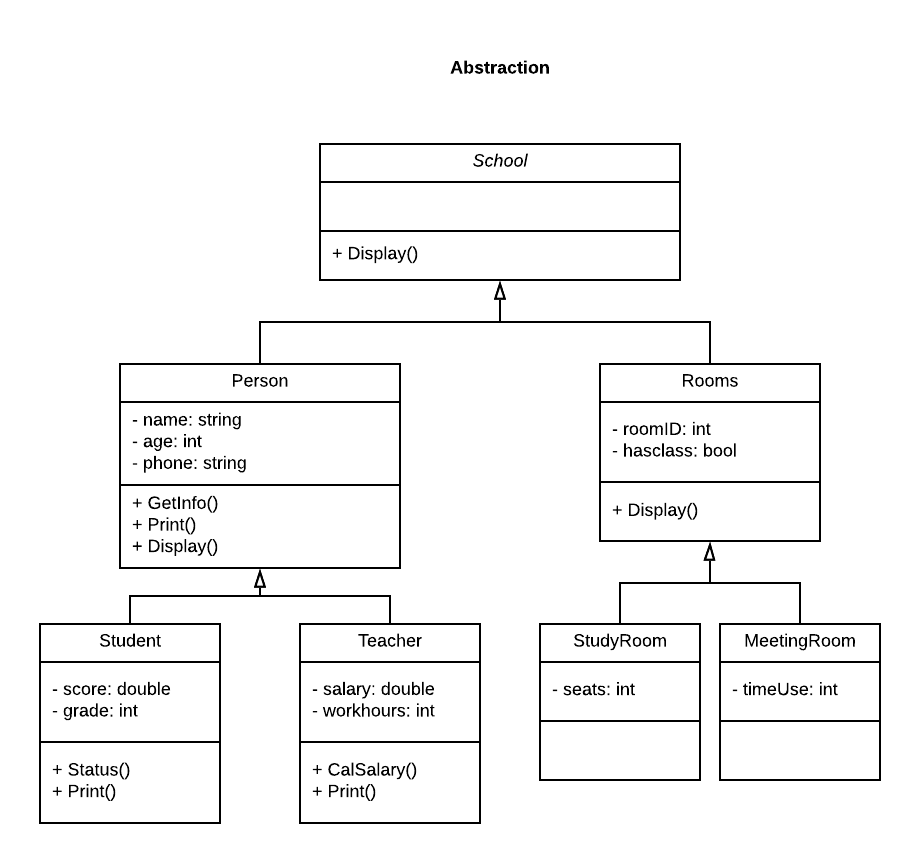
**Class Diagram of Abstraction:**

Figure . Abstraction class diagram

**Source code of Abstraction:**

class Persons : school

{

private string name;

private int age;

private string phone;

public virtual string Name

{

get { return name; }

set { name = value; }

}

public virtual int Age

{

get { return age; }

set { age = value; }

}

public virtual string Phone

{

get { return phone; }

set { phone = value; }

}

abstract class school

{

public virtual void Display()

{

}

}

class Room : school

{

private int roomID;

private bool hasClass;

public int roomID

{

get { return roomID; }

set { roomID = value; }

}

public bool HasClass

{

get { return hasClass; }

set { hasClass = value; }

}

public Room(int roomID, bool hasClass)

{

this.roomID = roomID;

this.hasClass = hasClass;

}

public override void Display()

{

Console.WriteLine("==== Welcome to University of Greenwich ====");

Console.WriteLine("==== Advance programing ====");

Console.WriteLine($"==== Room ID: {roomID} \t ====");

Console.WriteLine($"==== Room status: {hasClass} ====");

}

}

public Persons(string name, int age, string phone)

{

this.Name = name;

this.Age = age;

this.Phone = phone;

}

public void getInfo()

{

Console.WriteLine($"{name} is {age} years old and has phone number: {phone}");

}

public virtual void Print()

{

Console.WriteLine("=====Persons Information=====");

Console.WriteLine("Name\t\tAge\tPhone");

Console.WriteLine($"{Name}\t{Age}\t{Phone}");

}

public override void Display()

{

Console.WriteLine("==== Welcome to University of Greenwich ====");

Console.WriteLine("==== Advance programing ====");

Console.WriteLine($"==== Person Name: {Name} ====");

Console.WriteLine($"==== Person Age: {Age} ====");

Console.WriteLine($"==== Person Phone: {Phone} ====\n");

}

}

class Student : Persons

{

private double score;

private int grade;

public Student(string name, int age, string phone) : base(name, age, phone)

{

score = 0;

grade = 0;

}

public Student(string name, int age, string phone, double score) : base(name, age, phone)

{

this.score = score;

}

public Student(string name, int age, string phone, double score, int grade) : base(name, age, phone)

{

this.score = score;

this.grade = grade;

}

public void status()

{

if (score >= 5)

Console.Write($"Infomation of student {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tScore: {score}\n\tMove to next grade({grade} => {grade + 1})\n\n");

else Console.Write($"Infomation of student {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tScore: {score}\n\tCant move to next grade({grade} => {grade + 1})\n\n");

}

public override void Print()

{

Console.WriteLine("Name\t\tAge\tPhone\t\tScore\tGrade");

Console.WriteLine($"{Name}\t{Age}\t{Phone}\t{score}\t{grade}");

}

}

class Teacher : Persons

{

private double salary;

private int workhours;

public Teacher(string name, int age, string phone) : base(name, age, phone)

{

salary = 0;

workhours = 0;

}

public Teacher(string name, int age, string phone, double salary) : base(name, age, phone)

{

this.salary = salary;

workhours = 0;

}

public Teacher(string name, int age, string phone, double salary, int workhours) : base(name, age, phone)

{

this.salary = salary;

this.workhours = workhours;

}

public int CalSalary(int workhours)

{

return 10 \* workhours;

}

public void status()

{

Console.Write($"Infomation of teacher {Name}:\n\tAge: {Age}\n\tPhone Number: {Phone}\n\tSalary: {salary}\n\tWork hours: {workhours}\n\tSalary get: {CalSalary(workhours)}\n");

}

public override void Print()

{

Console.WriteLine("=====Teacher Information=====");

Console.WriteLine("Name\t\tAge\tPhone\t\tSalary\tWork hours");

Console.WriteLine($"{Name}\t{Age}\t{Phone}\t{salary}\t{workhours}");

}

}

**Code Main function of Abstraction:**

class Program

{

static void Main(string[] args)

{

school mySchool;

mySchool = new Student("QUang Huy", 24, "0795541090");

mySchool.Display();

mySchool = new Room(10, true);

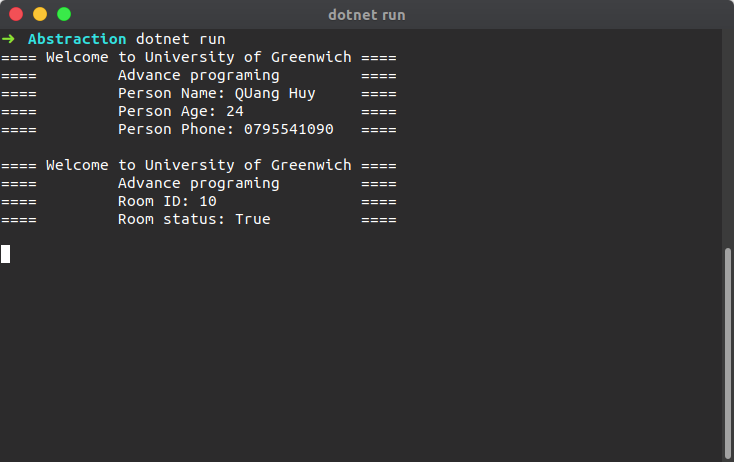
mySchool.Display();

Console.WriteLine();

Console.ReadKey();

}

}

**Result:**

Picture . Result of Abstraction code

**The different between Encapsulation and Abstraction in C#**

|  |  |
| --- | --- |
| **Encapsulation** | **Abstraction** |
| It is used to bind data members and member functions into single unit to prevent outsiders to access it directly. | It is used to hide unwanted data and show only required properties and methods. |

Table 1. The different between Encapsulation and Abstraction

# 