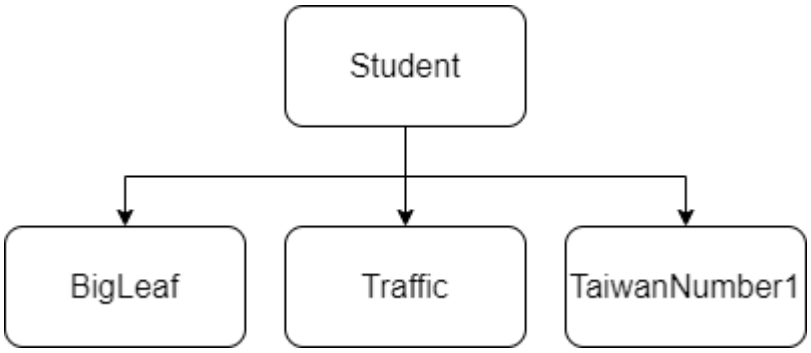
Lab 6

Role

For this problem, we will implement inheritance and polymorphism.

The architecture is shown as the following figure.

We have a base class Student, and three derived classes.



● Class Student

- Student(string ID, int score, int money, float background)
  Public constructor:
  - ID: Student's name.
  - score: Student's score.
  - money: Amount of money that the Student has.
  - background: Student's background.
- void PrintAllInfo()
  Print this Student's name, score, money and background. You can see the format in the sample output.
- virtual void ShowSchool()
  Pure virtual function.
- String getID()
  Public function to get the Student's ID.

● Class BigLeaf

- BigLeaf(string ID, int score, int money, float background)
  Public constructor. This class inherits the class Student.
- void ShowSchool()

Print “Student’s school is BigLeaf."

● **Class Traffic**

- Traffic(string ID, int score, int money, float background)
  Public constructor. This class inherits the class Student.
- void ShowSchool()
  Print “Student’s school is Traffic."

● **Class TaiwanNumber1**

- TaiwanNumber1(string ID, int score, int money, float background)
  Public constructor. This class inherits the class Student.
- void ShowSchool()
  Print “Student’s school is TaiwanNumber1."

Class PorkBall

Private:
Bool isTraffic(Student* student)
Check whether the dynamic casting is successful.

Public:
PorkBall(string name, float speed)
name: The porkball’s name.
speed: The porkball’s speed.
Bool canRide(Student* student);
Check whether the student can ride this porkball.

● **Function**

- bool isTraffic(Student* student)
  In this function, we would like to assign a pointer of a base class type (Student) to a pointer of its derived class type (Traffic). The advantage of dynamic casting is that it can be used to check whether casting is performed successfully.

  Implement dynamic casting, then return true and print ”Traffic Student” if casting success. Return False and print out “bad cast” if casting failed. Using this function to check whether the student is a Traffic student.

- bool canRide(Student* student)
  In this function, you need to use isTraffic to check whether the student is a Traffic student, if so, then he/she can ride the porkball.

  Example :

```
Student* bigleaf = new BigLeaf("Steve", 60, 50, 0.6);
bigleaf->printAllInfo();
bigleaf->ShowSchool();
cout << endl;

Student* traffic = new Traffic("Legolas", 99, 20, 0.9);
traffic->printAllInfo();
traffic->ShowSchool();
cout << endl;

PorkBall* porkball = new PorkBall("bb", 15.5);
cout << "Check whether the student can ride this porkball." << endl;
porkball->canRide(bigleaf);
porkball->canRide(traffic);
```

Output will be :

```
ID : Steve
Score : 60
Money : 50
Background : 0.6
Student Steve is BigLeaf student.

ID : Legolas
Score : 99
Money : 20
Background : 0.9
Student Legolas is Traffic student.

Check whether the student can ride this porkball.
error bad_cast
Legolas can ride porkball bb, the speed is 15.5.
```

* **Hint:**

1. We will provide a template (main.cpp and PorkBall.h) .
2. Please declare the class in the header file respectively. (Named Student.h, BigLeaf.h, …)