

Jobsheet 09 – Abstract Class and Interface

I. Competence

After completing this jobsheet students be able to:

1. Explain the purpose of using Abstract Class;
2. Explain the purpose of using the Interface;
3. Applying Abstract Class and Interface in making programs.

II. Introduction

Abstract Class

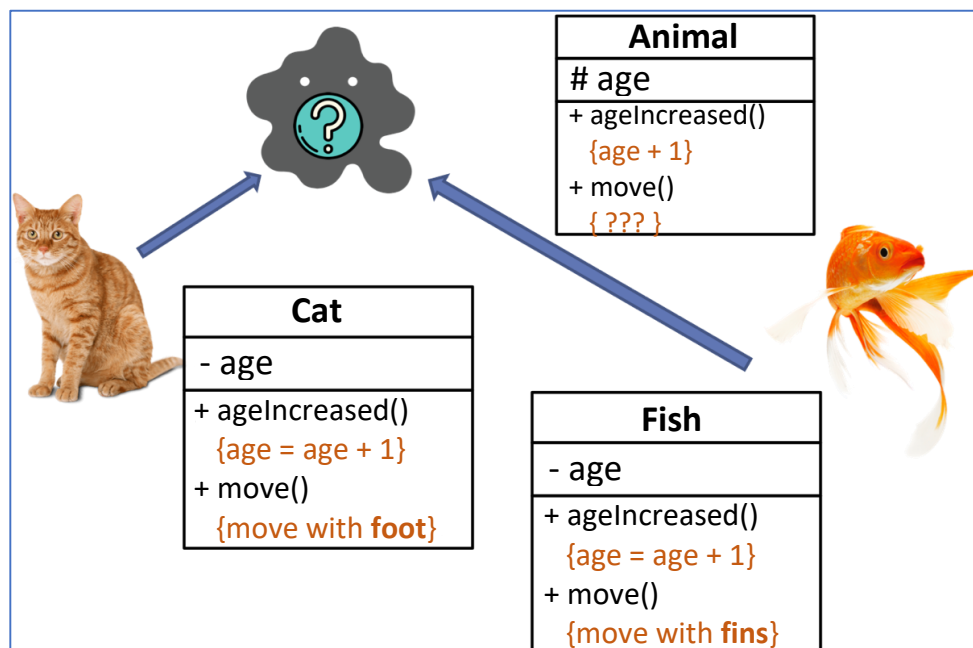
Abstract Class is a class that cannot be instantiated but can be extend. Abstract class only use when on extend.

Characteristics:

- a. Can have properties and methods like ordinary classes;
- b. Always have methods that don't have a body (only declarations), also called abstract methods;
- c. Always declared using the `abstract class` keyword.

Use:

Describe something that is general in nature, which can only function after it has been described in a more specific form.



Interface

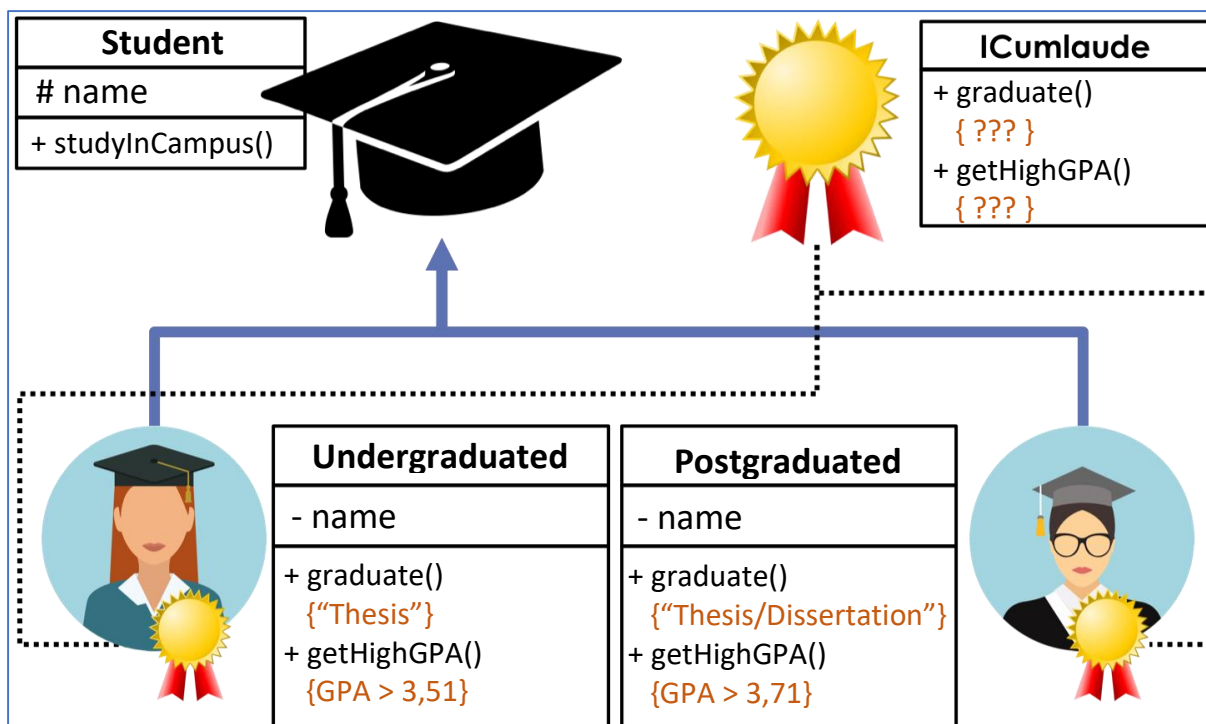
Interface is data structure that contains only abstract methods. There is nothing but abstract methods on the interface, including the getter and setter attributes.

Characteristics:

- That contains only abstract methods;
- At the conventions of the Java programming language, the name is recommended to always begin with a capital letter 'I';
- Always declared using keyword `interface`;
- Implemented using keyword `implements`.

Use:

Acting like a contract / condition that **MUST** be fulfilled for a class so that the class can be considered as 'something else'.

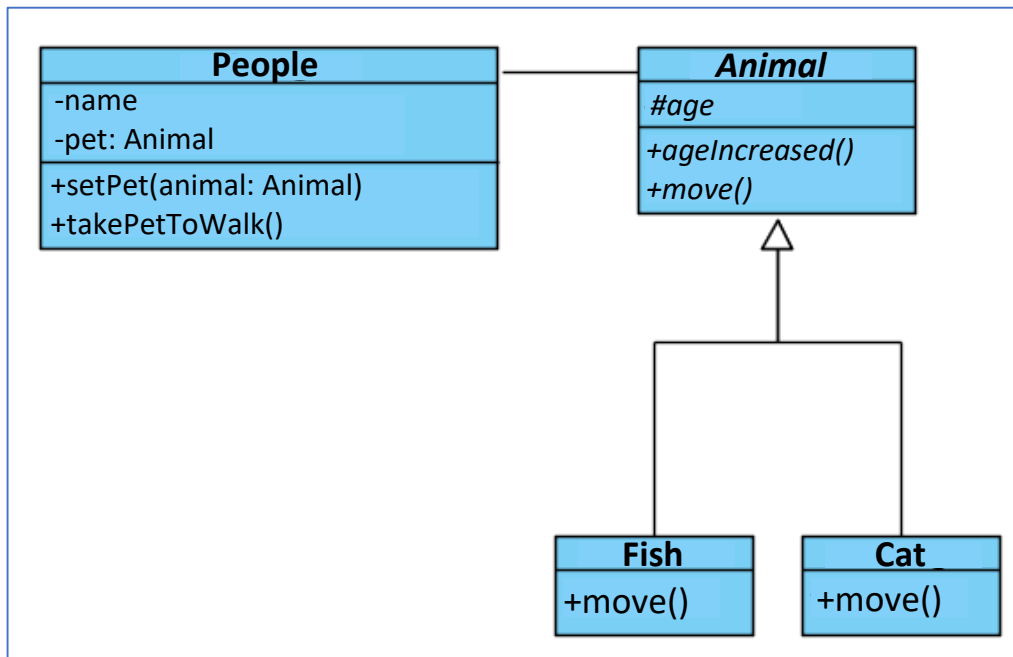


III. Practice

Experiment 1: Abstract Class

In this world there are many types of animals. All animals have some of the same characteristics, for example all animals have an age, whatever animal it is, an age will increase in number each year.

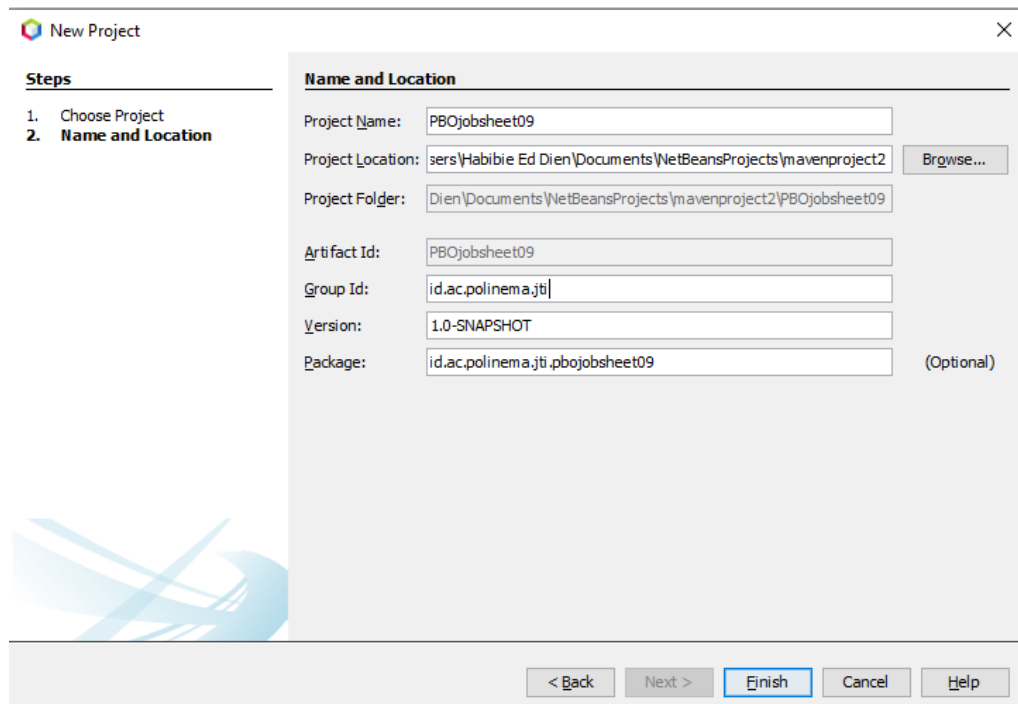
In addition to the same characteristics, each animal also has different characteristics from one another. For example in terms of moving. The way a cat moves is different from the way a fish moves. Cats move by moving their legs while fish move by moving their fins.



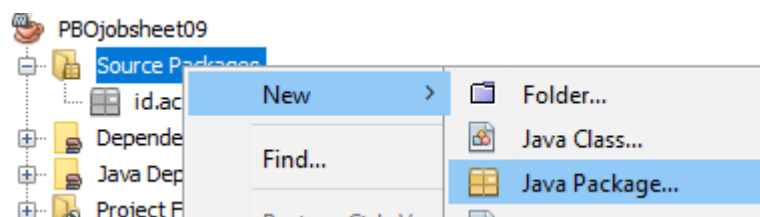
Every person who raises animals can take their pets to walk (make the pet move). But people who raise different animals, it will also differ the way their pets move.

In this first experiment we will create a program that illustrates the above scenario using **abstract class**.

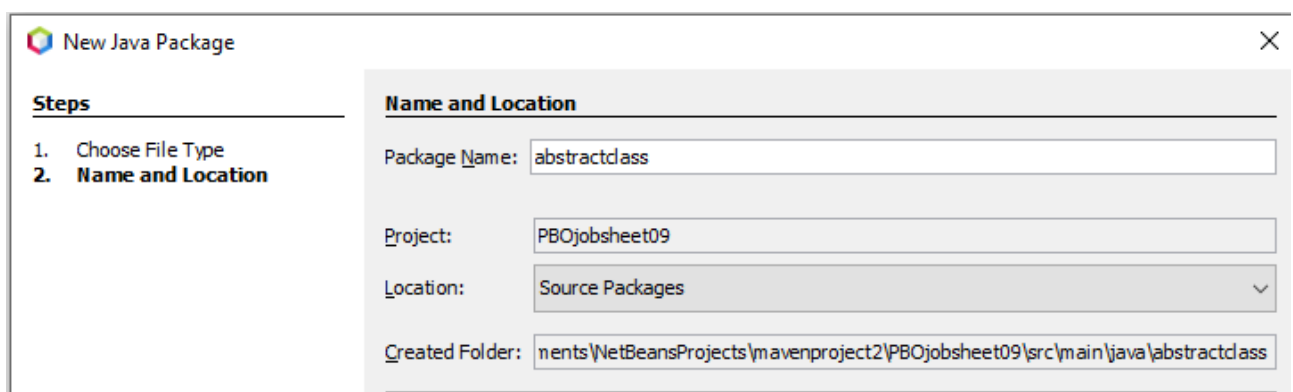
1. Create a new project on NetBeans with the name **PBOjobsheet09**



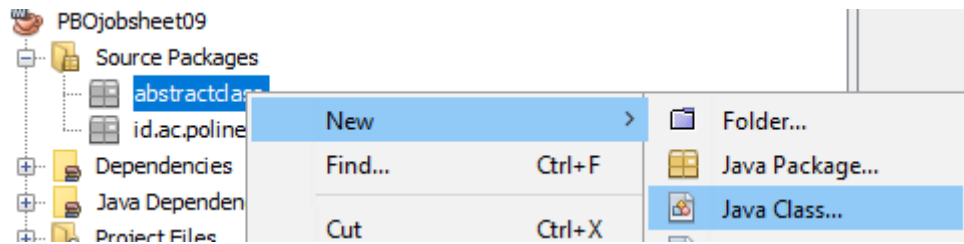
2. In **pbojobsheet09** package, add a new package by right-clicking the package name → New → Java Package...



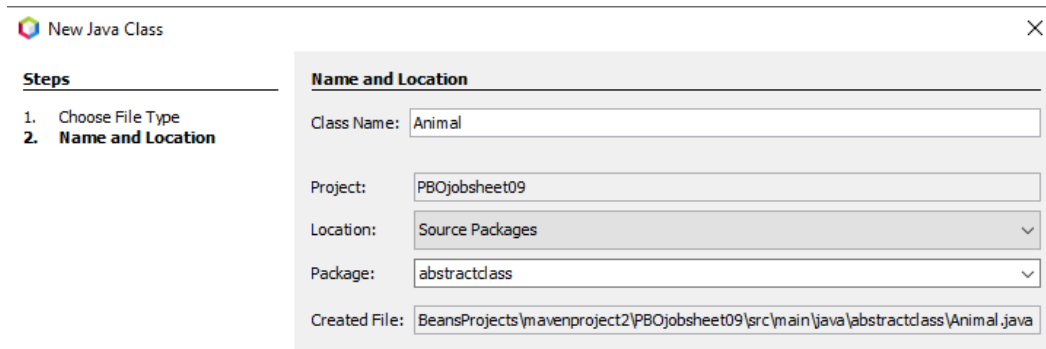
3. Name the package with the name **abstractclass**. All classes created in experiment 1 are **placed in the same package**, that is abstractclass package,



4. In that new package, add new class



5. Add name of that new class, that is **Animal** class.

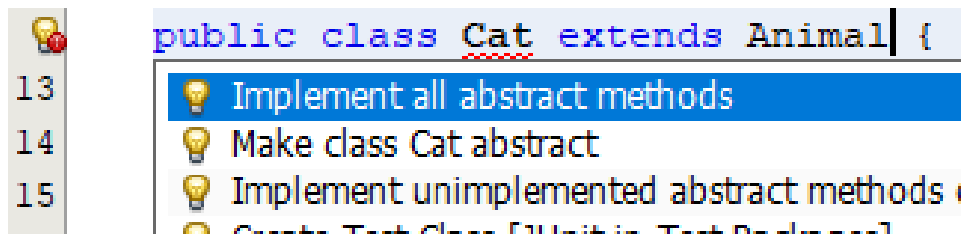


6. On that Hewan class, type this code below:

```
public abstract class Animal {  
  
    private int age;  
  
    protected Animal() {  
        this.age = 0;  
    }  
  
    public void ageIncreased() {  
        this.age += 1;  
    }  
  
    public abstract void move();  
}
```

The Animal class is an abstract class containing ordinary properties and methods, and an abstract method named **move()**. The method has an **abstract** keyword in front of it and does not have a function body. This method will be overridden by any class that is a class derived from Hewan class.

7. In the same way, create a class named **Cat** that extends the **Animal** class. In the Cat class, after you write the code as below, a warning light icon will appear. Click the light and then select **implement all abstract methods**.



8. Then a function will automatically be created that overrides the **move()** abstract function that exists in the Animal class.

```
public class Cat extends Animal {  
  
    @Override  
    public void move() {  
        throw new UnsupportedOperationException("Not supported yet.");  
    }  
  
}
```

9. Change the body of the function by replacing the code in it as follows.

```
@Override  
public void move() {  
    System.out.println("Walking with foot, \"Tap..tap..\"");  
}
```

10. In the same way as when you created the Cat class, create a new Animal class named **Fish** and make the code as shown below.

```
public class Fish extends Animal {  
  
    @Override  
    public void move() {  
        System.out.println("Swimming with fins, \"wush..wush..\"");  
    }  
  
}
```

11. Next, create a new ordinary class called the **People** class. This class is a class that is a user of Animal abstract class that created previously. Type in that People class, lines of code as below.

```

public class People {

    private String name;
    private Animal pet;

    public People(String name) {
        this.name = name;
    }

    public void setPet(Animal pet) {
        this.pet = pet;
    }

    public void takePetToWalk() {
        System.out.println("My name is " + this.name);
        System.out.println("My pet move by: ");
        this.pet.move();
        System.out.println("-----");
    }
}

```

12. Finally, make a new Main Class in the same package. Name the new class the name **Program** class. Type in it like the code below.

```

public class Program {
    public static void main(String[] args) {
        Cat asiaCat = new Cat();
        Fish dolphin = new Fish();

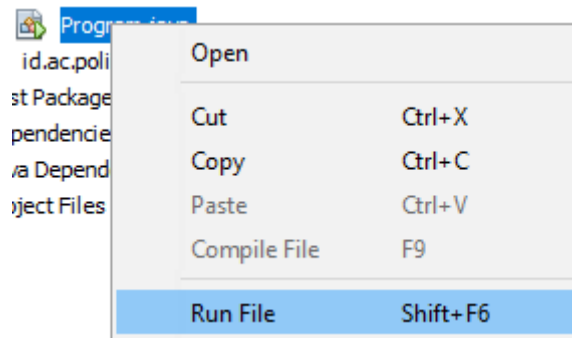
        People john = new People("John");
        People ben = new People("Ben");

        john.setPet(asiaCat);
        ben.setPet(dolphin);

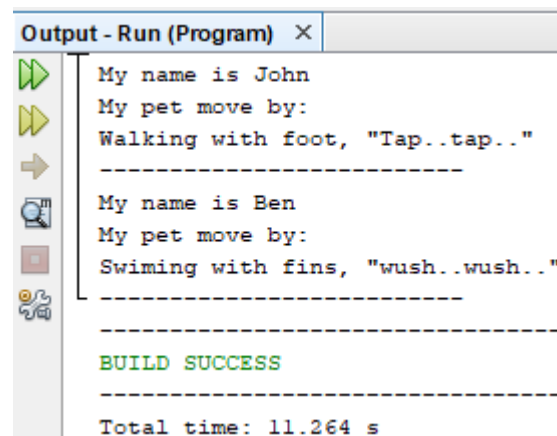
        john.takePetToWalk();
        ben.takePetToWalk();
    }
}

```

13. Run the class by right-clicking on the Program class then select **Run File** (Shift + F6).



14. Watch and observe the results!

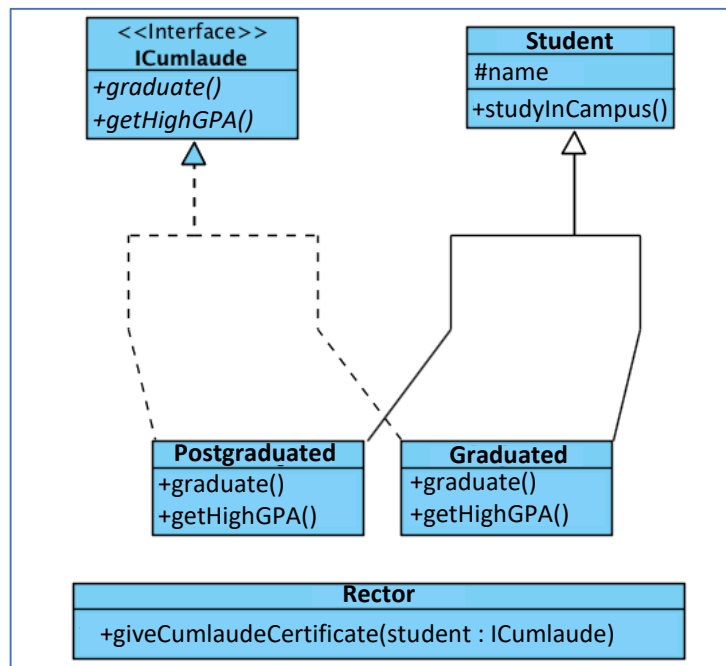


15. Discussion questions:

Can it be possible if a class that extends an abstract class does not implement abstract methods in its parent class? Prove it!

Experiment 2: Interface

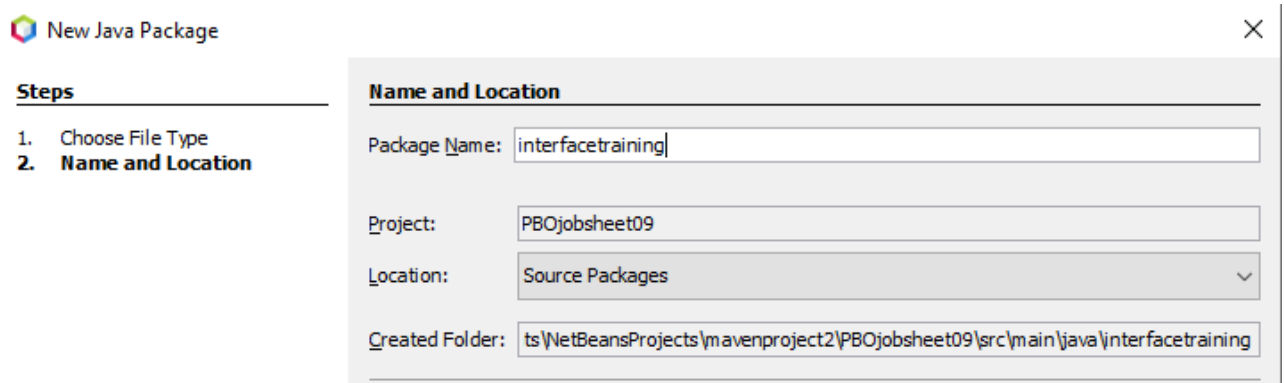
At a graduation ceremony, a Rector will award Cum laude certificates to all students who meet the requirements. The requirements for a student to be called Cum laude vary between Undergraduated and Postgraduated students.



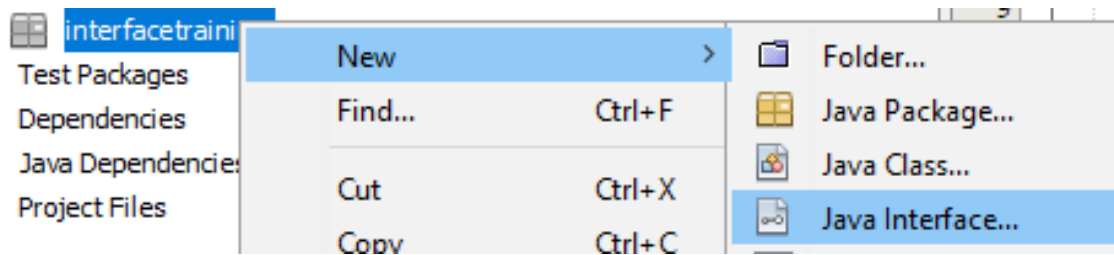
To become a cum laude, Undergraduated students must work on a thesis and have a GPA higher than 3.51. Whereas for postgraduated students, they must work on a thesis and achieve a higher GPA of 3.71

In this experiment we will try to translate the above scenario into a simple application that utilizes the interface.

1. In **the same project**, make a new package called **interfacetraining**.



2. In **that** newly created package, add a new interface by right-clicking on the package → **New** → **Java Interface ...** Give that new interface with the name **ICumlaude**.



3. In that ICumlaude interface, add 2 abstract methods named **graduate()** and **getHighGPA()**

```
public interface ICumlaude {  
    public abstract void graduate();  
    public abstract void getHighGPA();  
}
```

4. Next, create a new class called **Student** with lines of code as below.

```
public class Student {  
  
    protected String name;  
  
    public Student(String name) {  
        this.name = name;  
    }  
  
    public void studyInCampus() {  
        System.out.println("I am a student, my name is " + this.name);  
        System.out.println("I study in cammpus.");  
    }  
}
```

5. Next, create a new class called **Undergraduated** which is a **derivative** of the Student class. The Undergraduated Class is made to implement the interface ICumlaude that was created earlier. Type the code below in the class. **Clue:** You can use the automatic override facility in the same way by clicking on the warning light icon as in experiment 1.

```

public class Undergraduated extends Student implements ICumlaude {

    public Undergraduated(String name) {
        super(name);
    }

    @Override
    public void graduate() {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void getHighGPA() {
        throw new UnsupportedOperationException("Not supported yet.");
    }

}

```

6. Next, adjust the contents of method **graduate()** and **getHighGPA()** to be the same as the line of code below.

```

public class Undergraduated extends Student implements ICumlaude {

    public Undergraduated(String name) {
        super(name);
    }

    @Override
    public void graduate() {
        System.out.println("I have finished the thesis for Undergraduated");
    }

    @Override
    public void getHighGPA() {
        System.out.println("My GPA is more than 3,51");
    }

}

```

Note in the above line of code, the Undergraduated class extends the Student class, this means, the Undergraduated is a Student so that all objects of this Undergraduated class can later be called **Cumlaude** so it must **implement** the **ICumlaude** interface.

7. Then in the same way create a new class named **Postgraduated** with a line of code like below

```

public class Postgraduated extends Student implements ICumlaude {

    public Postgraduated(String name) {
        super(name);
    }

    @Override
    public void graduate() {
        System.out.println("I have finished the thesis for Postgraduated");
    }

    @Override
    public void getHighGPA() {
        System.out.println("My GPA is more than 3,71");
    }

}

```

8. Then make a new class called **Rector**. This class is a class that utilizes Student class that have been previously created.

```

public class Rector {
    public void giveCumlaudeCertificate(ICumlaude student){
        System.out.println("I am a Rector, give a Cumlaude Certificate");
        System.out.println("Congratulations! Please introduce yourself..");

        student.graduate();
        student.getHighGPA();

        System.out.println("-----");
    }
}

```

9. Finally, create a new class named **Program** that is placed in the same package. Add the following line of code:

```

public class Program {

    public static void main(String[] args) {
        Rector MrRector = new Rector();

        Student ordinaryStudent = new Student("Charlie");
        Undergraduated undergraduatedCumlaude = new Undergraduated("Joe");
        Postgraduated postgraduatedCumlaude = new Postgraduated("Ben");

        MrRector.giveCumlaudeCertificate(ordinaryStudent);
        MrRector.giveCumlaudeCertificate(undergraduatedCumlaude);
        MrRector.giveCumlaudeCertificate(postgraduatedCumlaude);
    }
}

```

10. In that line of code, if you type all classes correctly, there will be an error and the Program class cannot be executed. Revise your code so that the program you created will show outputs the following:

```

--- exec-maven-plugin:1.5.0:exec (default-cli)
I am a Rector, give a Cumlaude Certificate
Congratulations! Please introduce yourself..
I have finished the thesis for Undergraduated
My GPA is 3,1
-----
I am a Rector, give a Cumlaude Certificate
Congratulations! Please introduce yourself..
I have finished the thesis for Undergraduated
My GPA is more than 3,51
-----
I am a Rector, give a Cumlaude Certificate
Congratulations! Please introduce yourself..
I have finished the thesis for Postgraduated
My GPA is more than 3,71
-----
BUILD SUCCESS

```

11. Discussion questions:
- Why did step 9 cause an error? Explain!
 - Can the **studyInCampus()** method be called from the **undergraduatedCumlaude** object in the Program class? Why?
 - Can the **studyInCampus()** method be called from the student parameters in the **giveCumlaudeCertificate()** method in the Rector class? Why?

- d. Modify the **giveCumlaudeCertificate()** method on the **Rector** class so that the results of the **Program** class execution become as follows:

```

Output - Run (Program) X
--- exec-maven-plugin:1.5.0:exec (default-cli)
I am a Rector, give a Cumlaude Certificate
Congratulations! Please introduce yourself..
I am a student, my name is Charlie
I study in cammpus.
I have finished the thesis for Undergraduated
My GPA is 3,1

-----
I am a Rector, give a Cumlaude Certificate
Congratulations! Please introduce yourself..
I am a student, my name is Joe
I study in cammpus.
I have finished the thesis for Undergraduated
My GPA is more than 3,51

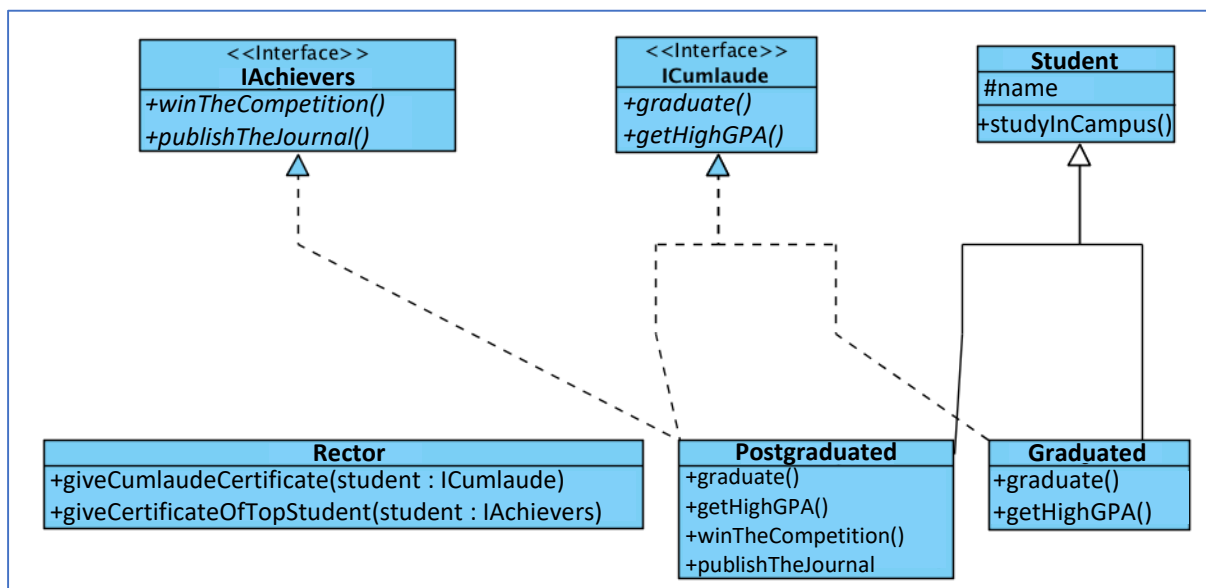
-----
I am a Rector, give a Cumlaude Certificate
Congratulations! Please introduce yourself..
I am a student, my name is Ben
I study in cammpus.
I have finished the thesis for Postgraduated
My GPA is more than 3,71

-----
BUILD SUCCESS

```

Experiment 3: Multiple Interfaces Implementation

In this experiment we will modify the program created in Experiment 2 so that the program will have a class that implements more than one interface.



Imagine the previous scenario, where a rector would also give a **CertificateOfTopStudent()** at a graduation ceremony. College students who are entitled to receive the award are certainly outstanding college students, where the criteria for achievement here differ between Undergraduate

students and Postgraduate students. In this experiment, we will determine the achievement criteria, namely: must **winTheCompetition()** and **publishTheJournal()**.

1. In the **same package as the package in Experiment 2**, add a new interface called **IAchievers**. Add lines of code as follows in it.

```
public interface IAchievers {  
    public abstract void winTheCompetition();  
    public abstract void publishTheJournal();  
}
```

2. Next, modify the **Postgraduated** class by adding a **IAchievers** new interface behind the **implements** keyword. Then in the same way as before, click the warning light icon to **generate** all abstract methods from the **IAchievers** interface in the **Postgraduated** class.

```
13 public class Postgraduated extends Student implements ICumlaude, IAchievers {
```

3. Modify the method that has generated by NetBeans as follows

```
@Override  
public void winTheCompetition() {  
    System.out.println("I have won an INTERNATIONAL competition");  
}  
  
@Override  
public void publishTheJournal() {  
    System.out.println("I publish articles in INTERNATIONAL journals");  
}
```

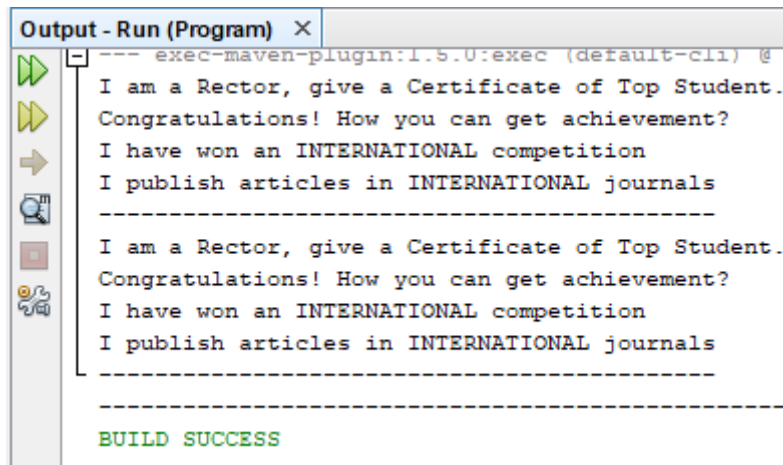
4. Add **giveCertificateOfTopStudent()** method with a line of code as below, to the **Rector** class.

```
public void giveCertificateOfTopStudent(IAchievers student) {  
    System.out.println("I am a Rector, give a Certificate of Top Student.");  
    System.out.println("Congratulations! How you can get achievement?");  
  
    student.winTheCompetition();  
    student.publishTheJournal();  
  
    System.out.println("-----");  
}
```

5. Finally, modify the **main()** method in your **Program** class. Comment all the lines that contain the method **giveCumlaudeCertificate()**, then add a new line of code like in the picture below:

```
public class Program {  
  
    public static void main(String[] args) {  
        Rector MrRector = new Rector();  
  
        //Student ordinaryStudent = new Student("Charlie");  
        Undergraduated undergraduatedCumlaude = new Undergraduated("Joe");  
        Postgraduated postgraduatedCumlaude = new Postgraduated("Ben");  
  
        //MrRector.giveCumlaudeCertificate(ordinaryStudent);  
        //MrRector.giveCumlaudeCertificate(undergraduatedCumlaude);  
        //MrRector.giveCumlaudeCertificate(postgraduatedCumlaude);  
  
        MrRector.giveCertificateOfTopStudent(undergraduatedCumlaude);  
        MrRector.giveCertificateOfTopStudent(postgraduatedCumlaude);  
    }  
}
```

6. There will be an error in step-5, so the program cannot be executed. Please correct the program code, so that the execution results are **same** as in the screenshot below.



```
Output - Run (Program) X  
--- exec-maven-plugin:1.5.0:exec (default-cli) @  
I am a Rector, give a Certificate of Top Student.  
Congratulations! How you can get achievement?  
I have won an INTERNATIONAL competition  
I publish articles in INTERNATIONAL journals  
-----  
I am a Rector, give a Certificate of Top Student.  
Congratulations! How you can get achievement?  
I have won an INTERNATIONAL competition  
I publish articles in INTERNATIONAL journals  
-----  
BUILD SUCCESS  
-----
```

*** End Of Jobsheet-09 ***