Project 2

Software Engineering

CSIT at UDC

Source: https://plantuml.com/class-diagram

Class vs. Struct

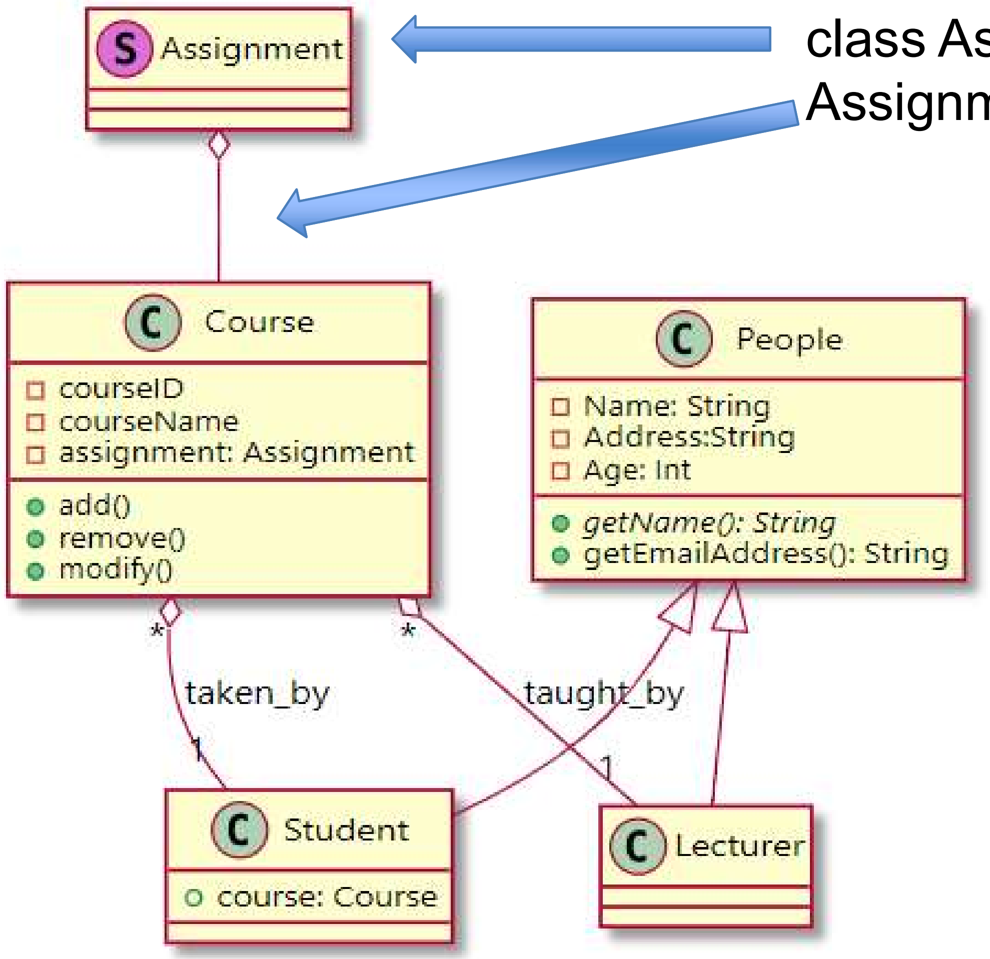
- ✧ A ‘Class’ is like a struct that defaults to ‘private’ instead of ‘public’.
- ✧ Structures are user defined data types which are used to store the group of items of non-similar data types.

Example

- Also known as ‘structs’ and ‘types’.
  - C

```
struct resident {
  char initials[2];
  int ss_number;
  bool married;
};
```

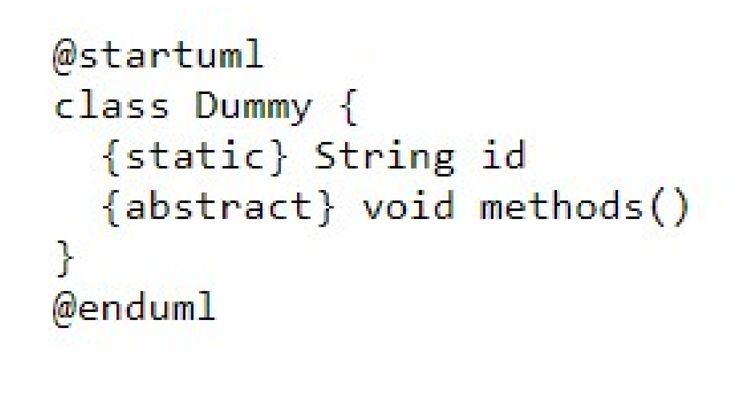
Struct in Project 1



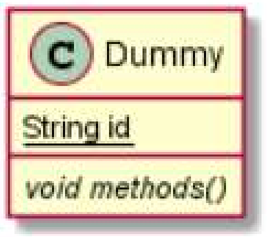
class Assignment << (S,orchid) >>
Assignment o-- Course

Abstract and Static

You can define static or abstract methods or fields using the {static} or {abstract} modifier.

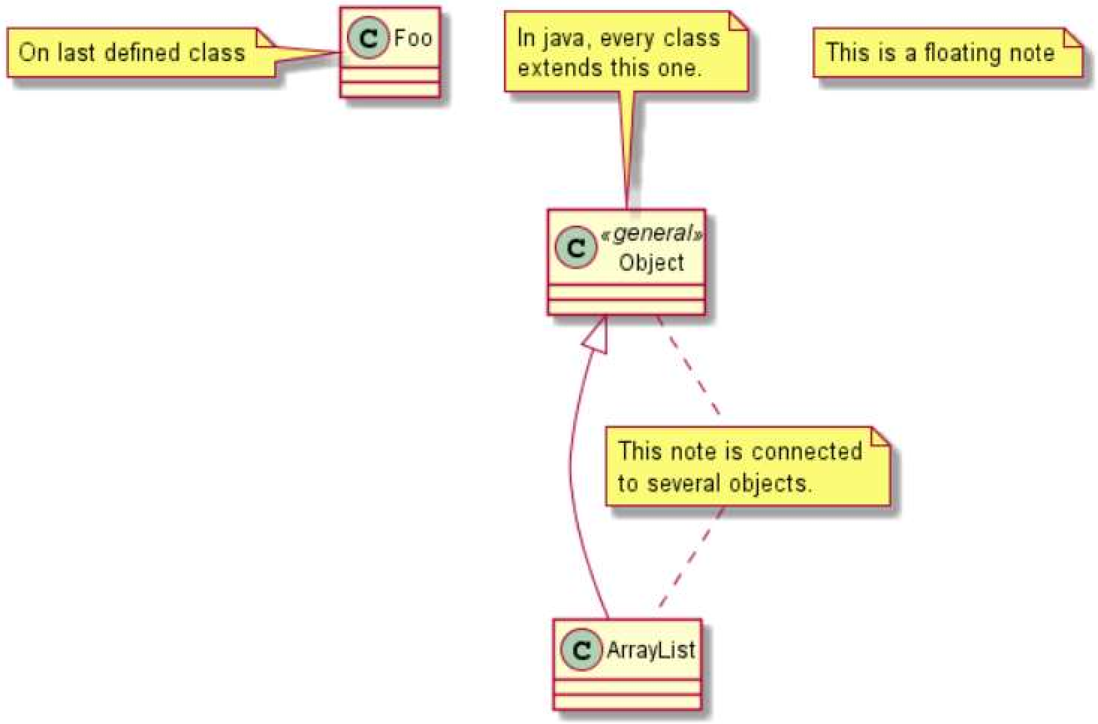


```
@startuml
class Dummy {
  {static} String id
  {abstract} void methods()
}
@enduml
```

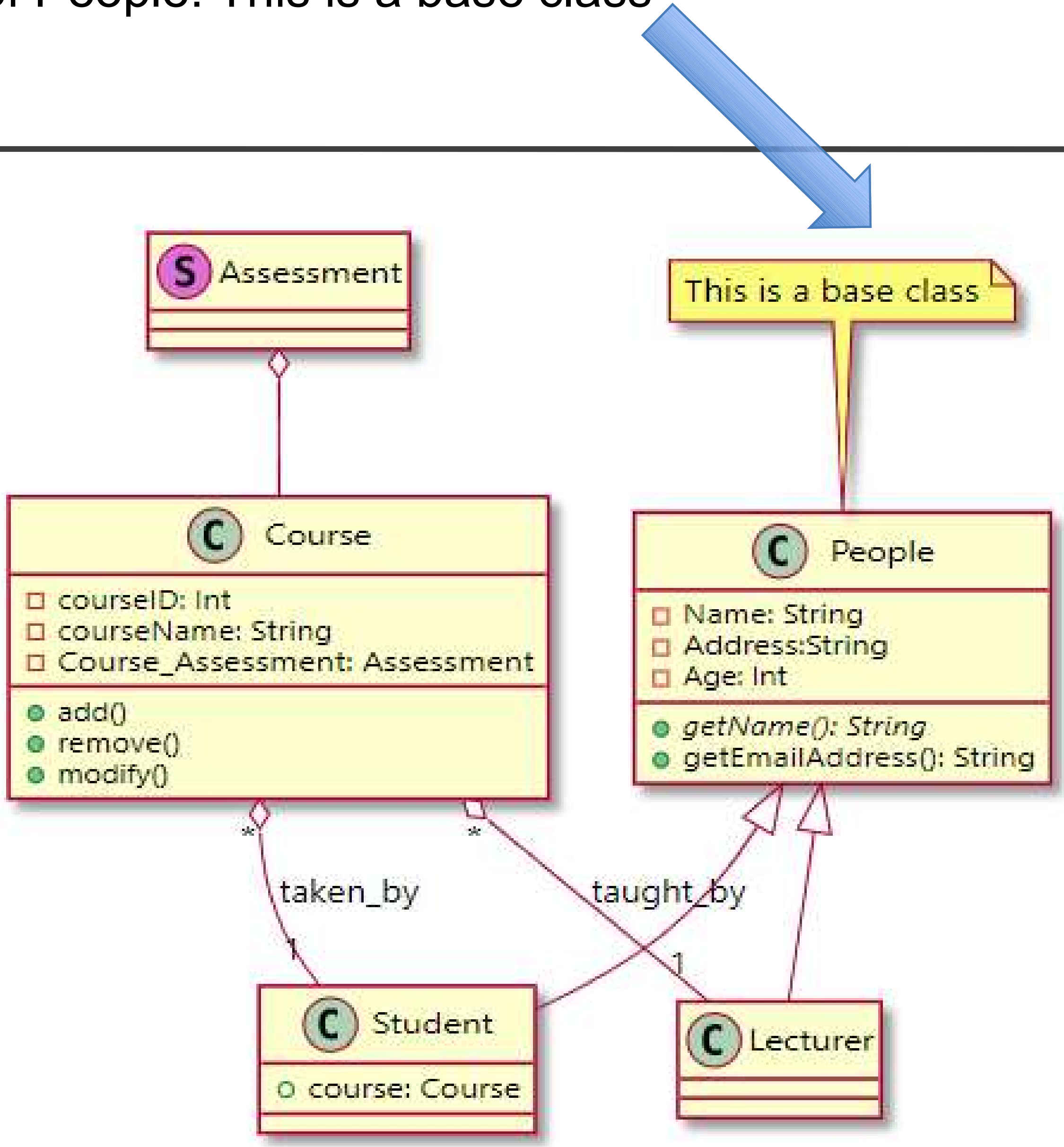


Notes

```
@startuml
class Object << general >>
Object <|--- ArrayList

note top of Object : In java, every class\nextends this one.

note "This is a floating note" as N1
note "This note is connected\nto several objects." as N2
Object .. N2
N2 .. ArrayList

class Foo
note left: On last defined class

@enduml
```


On last defined class
Foo
In java, every class extends this one.
This is a floating note
«general» Object
This note is connected to several objects.
ArrayList

note top of People: This is a base class



Abstract Class

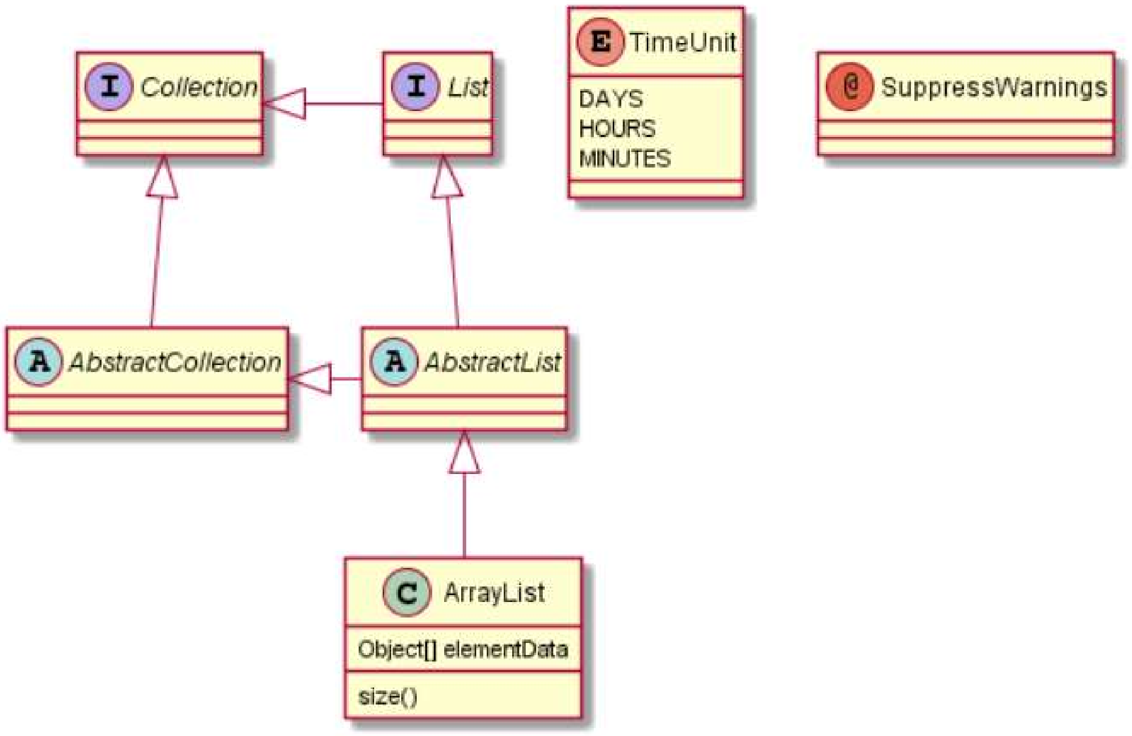
```
// Base class
class Shape {
  public:
    // pure virtual function providing interface framework.
    virtual int getArea() = 0;
    void setWidth(int w) {
      width = w;
    }

    void setHeight(int h) {
      height = h;
    }
  protected:
    int width;
    int height;
};


// Derived classes
class Rectangle: public Shape {
  public:
    int getArea() {
      return (width * height);
    }
};
```

enum

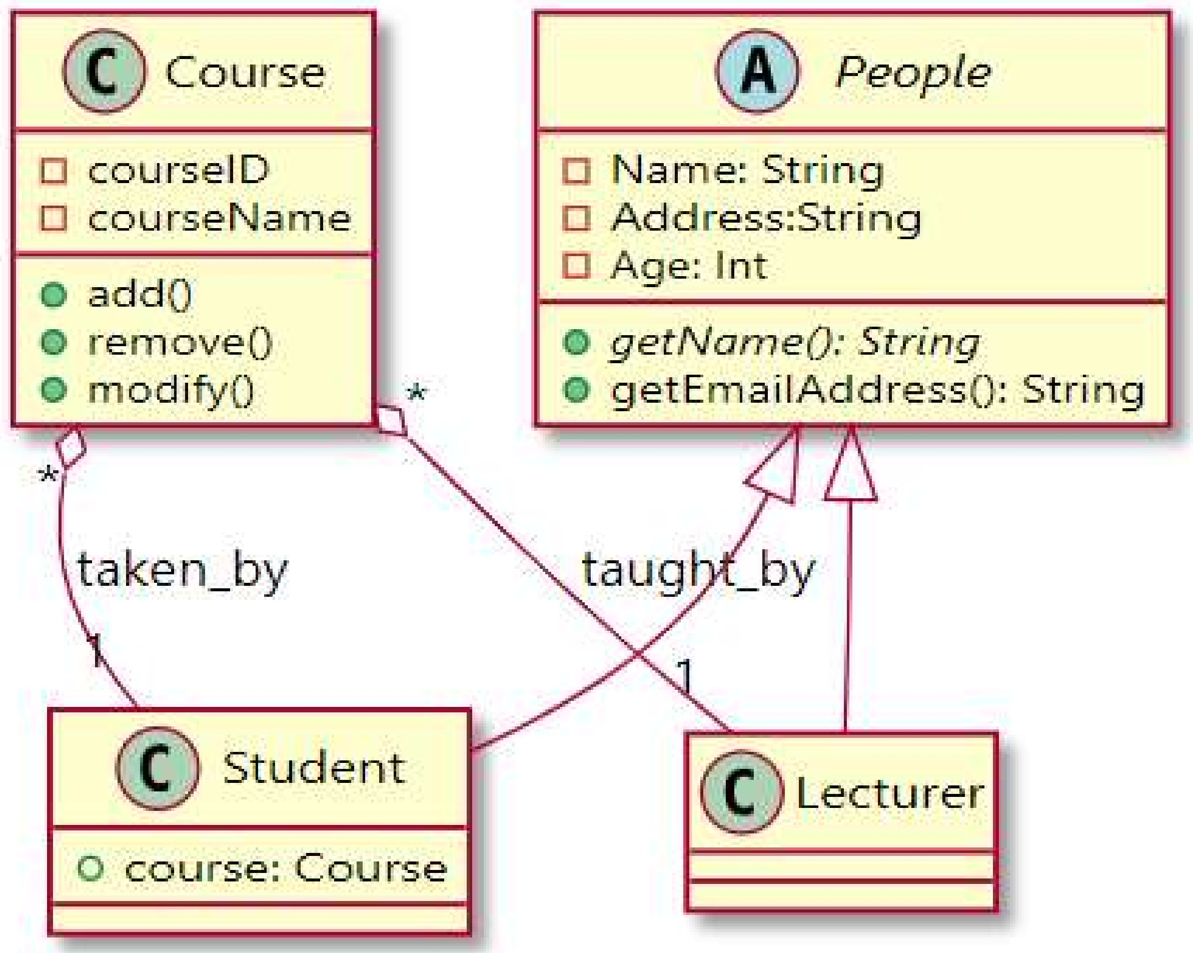
```
enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};
int main()
{
    enum week day;
    day = Wed;
    printf("%d",day);
    return 0;
}
```

Abstract Class and Interface

```
@startuml

abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection

List <|-- AbstractList
Collection <|-- AbstractCollection

Collection <|- List
AbstractCollection <|- AbstractList
AbstractList <|-- ArrayList

class ArrayList {
  Object[] elementData
  size()
}

enum TimeUnit {
  DAYS
  HOURS
  MINUTES
}

annotation SuppressWarnings

@enduml
```



Abstract Class in Project 1

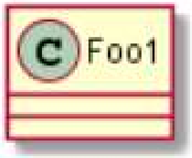


Extend in JAVA

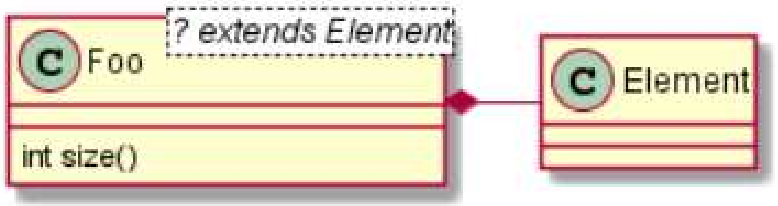
```
abstract class myAbstractClass {
    abstract void myAbstractFun();
     void fun() {
          System.out.println("Inside My fun");
  }
}
public class myClass extends myAbstractClass {
  public void myAbstractFun() {
     System.out.println("Inside My fun");
  }
}
```

Hide and Generics

```
@startuml

class Foo1
class Foo2

Foo2 *-- Foo1

hide Foo2

@enduml
```



```
@startuml

class Foo<? extends Element> {
  int size()
}
Foo *- Element

@enduml
```



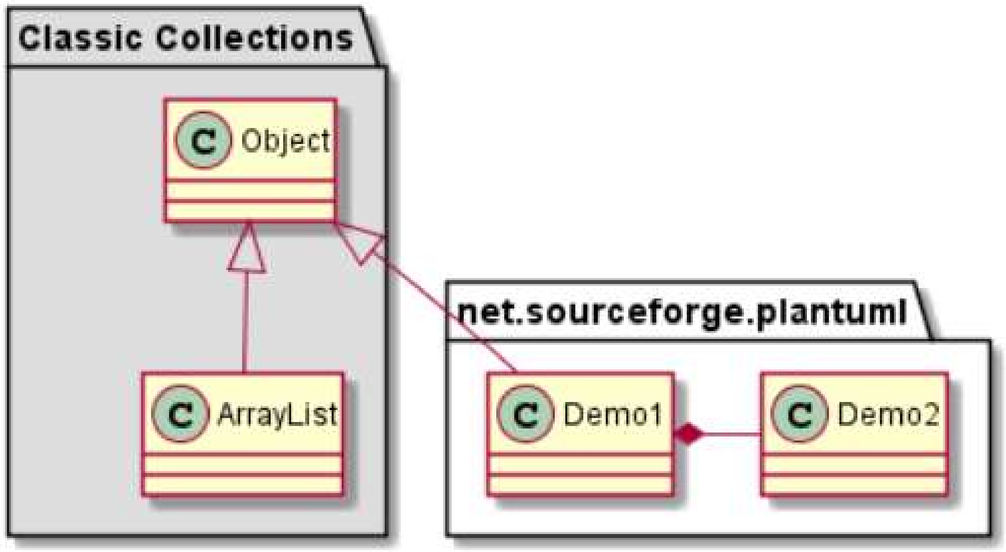
Package

```
// Name of the package must be same as
the directory under which this file is saved

package myPackage;

public class MyClass

{

    public void getNames(String s)

    {

        System.out.println(s);

    }

}

/* import 'MyClass' class from 'names'
myPackage */

import myPackage.MyClass;
```

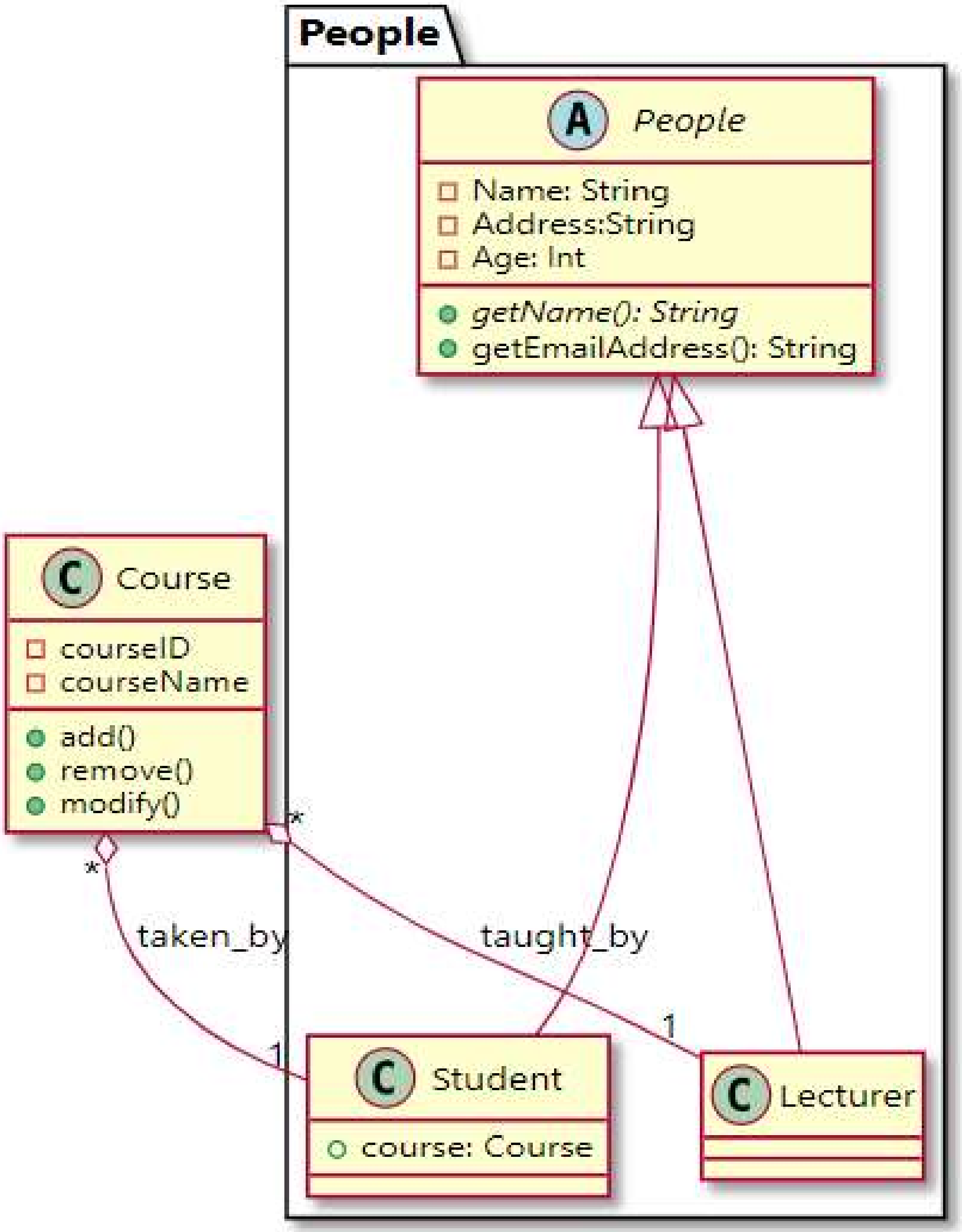
```
public class PrintName

{

   public static void main(String args[])

   {

 // Initializing the String variable  with a value

      String name = “Software Engineering";

      // Creating an instance of class MyClass
in the package.

      MyClass obj = new MyClass();

      obj.getNames(name);

   }

}
```

Packages

```
@startuml

package "Classic Collections" #DDDDDD {
  Object <|-- ArrayList
}

package net.sourceforge.plantuml {
  Object <|-- Demo1
  Demo1 *- Demo2
}

@enduml
```

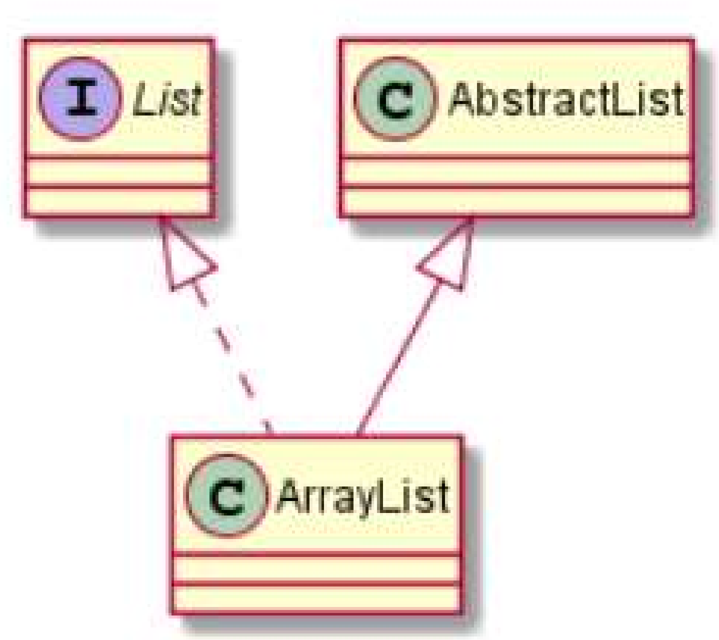


People Package In Project 1



Extends and Implements

```
@startuml
class ArrayList implements List
class ArrayList extends AbstractList
@enduml
```

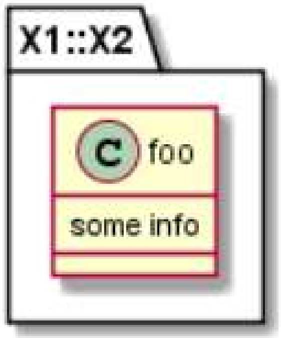


Namespace

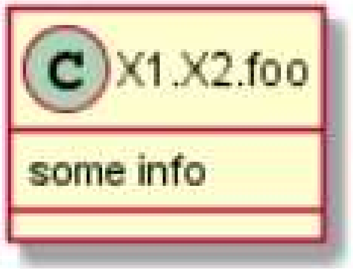
```
#include <iostream>
using namespace std;
// first name space
namespace first_space {
   void func() {
      cout << "Inside first_space" << endl;
   }
}
// second name space
namespace second_space {
   void func() {
      cout << "Inside second_space" << endl;
   }
}
int main () {
   // Calls function from first name space.
   first_space::func();
   // Calls function from second name space.
   second_space::func();
   return 0;
}
```

Namespace creation

```
@startuml

set namespaceSeparator ::
class X1::X2::foo {
  some info
}

@enduml
```



```
@startuml

set namespaceSeparator none
class X1.X2.foo {
  some info
}

@enduml
```



Project 2 (Design Class Diagram)

✧ Enhance the class diagram of **specification 2** that you designed in Project 1

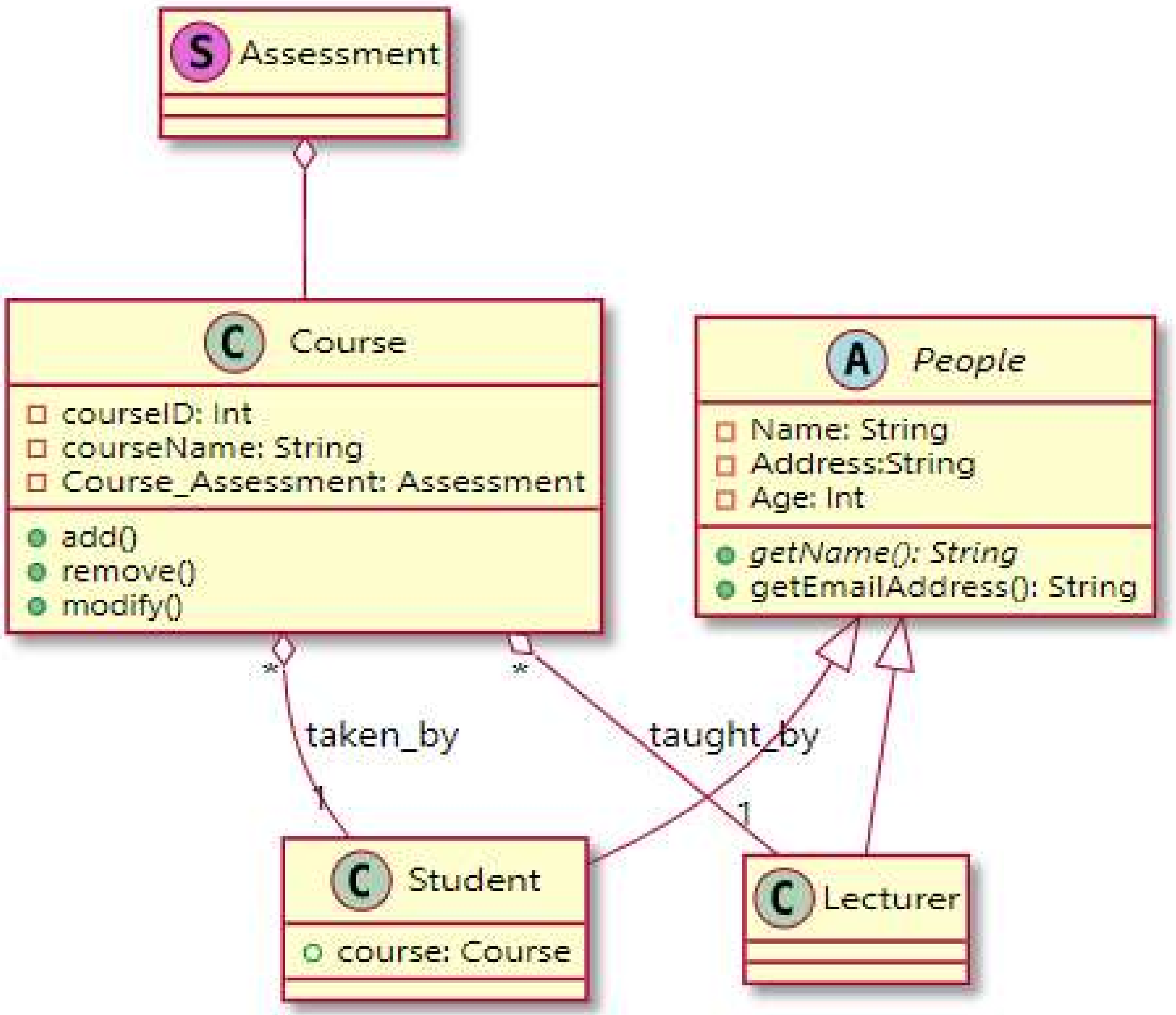
1. An “assessment” is declared as a structure, and *Course_assessment* is defined as an attribute in the Course class. (lab)
2. As designing #1, an assignment is declared as a structure and Course_Assignment is defined as an attribute in the Course class (10%)
3. *Midterm* and *Final exam*s are defined in the Assessment structure. (10%)
4. *Online, Hybrid, and Standard* courses are defined as enumeration types in the Course class. The name of the enumeration is *Section*. *Section_number* is defined as an enumeration in Course(10%)
5. Students can take maximum 10 courses. Lecturers can teach maximum 3 courses (10%).
6. The People, Student and Lecturer classes must be combined as a package (10%).
7. As a static attribute. an ID is defined in People and used for Student and Lecturer (10%)
8. An “Enroll” operation is defined in Student with the Course class parameter (20%)
9. Two types of courses can be defined as Undergraduate and Graduate courses. Identify two courses using namespace. (20%)

Midterm Exam on 10/13

- In class exam
- Close Book
- 20 questions in Evaluation questions

LAB (Part of Project 2)

- ✧Design the following specification
  - Open your *uml* design for project 1’s specification 2.
  - An “*Assessment*” structure is declared in the Course class
  - *Course_assessment* is defined as an attribute in the Course class.



S Assessment
C Course
□ courseID: Int
□ courseName: String
□ Course_Assessment: Assessment
● add()
● remove()
● modify()
A People
□ Name: String
□ Address:String
□ Age: Int
● getName(): String
● getEmailAddress(): String
*
taken_by
1
*
taught_by
1
C Student
○ course: Course
C Lecturer