XIAMEN UNIVERSITY MALAYSIA
厦門大學马来西亚分校

CST209

Object-oriented Programming C++

(Week 15)

Prepared by Dr. Teo Bee Guan

Presented By Mr. Venantius Kumar Sevamalai

Content

- UML Class Diagram
- https://www.lucidchart.com/pages/

UML Class Diagram

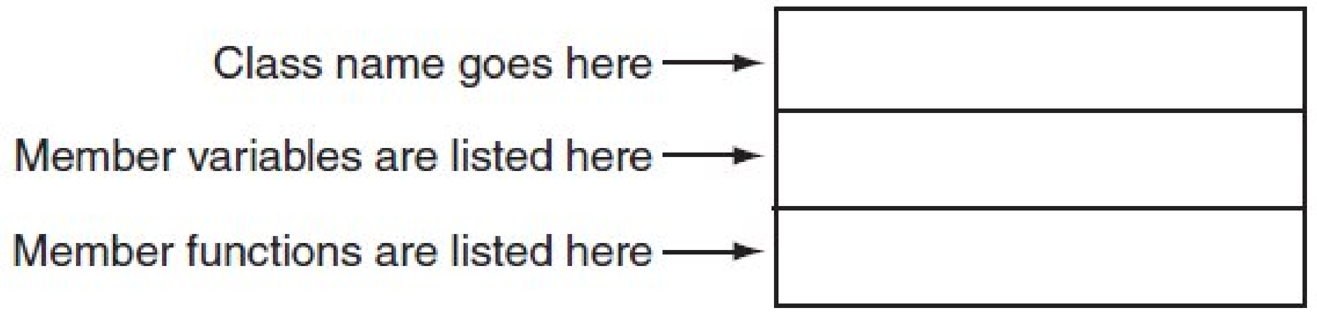
- The Unified Modeling Language (UML) provides a standard method for graphically depicting an object-oriented system.

- When designing a class it is often helpful to draw a UML Class diagram.

- A UML Class diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

UML Class Diagram

The general layout of a UML Class diagram for a class is shown below.

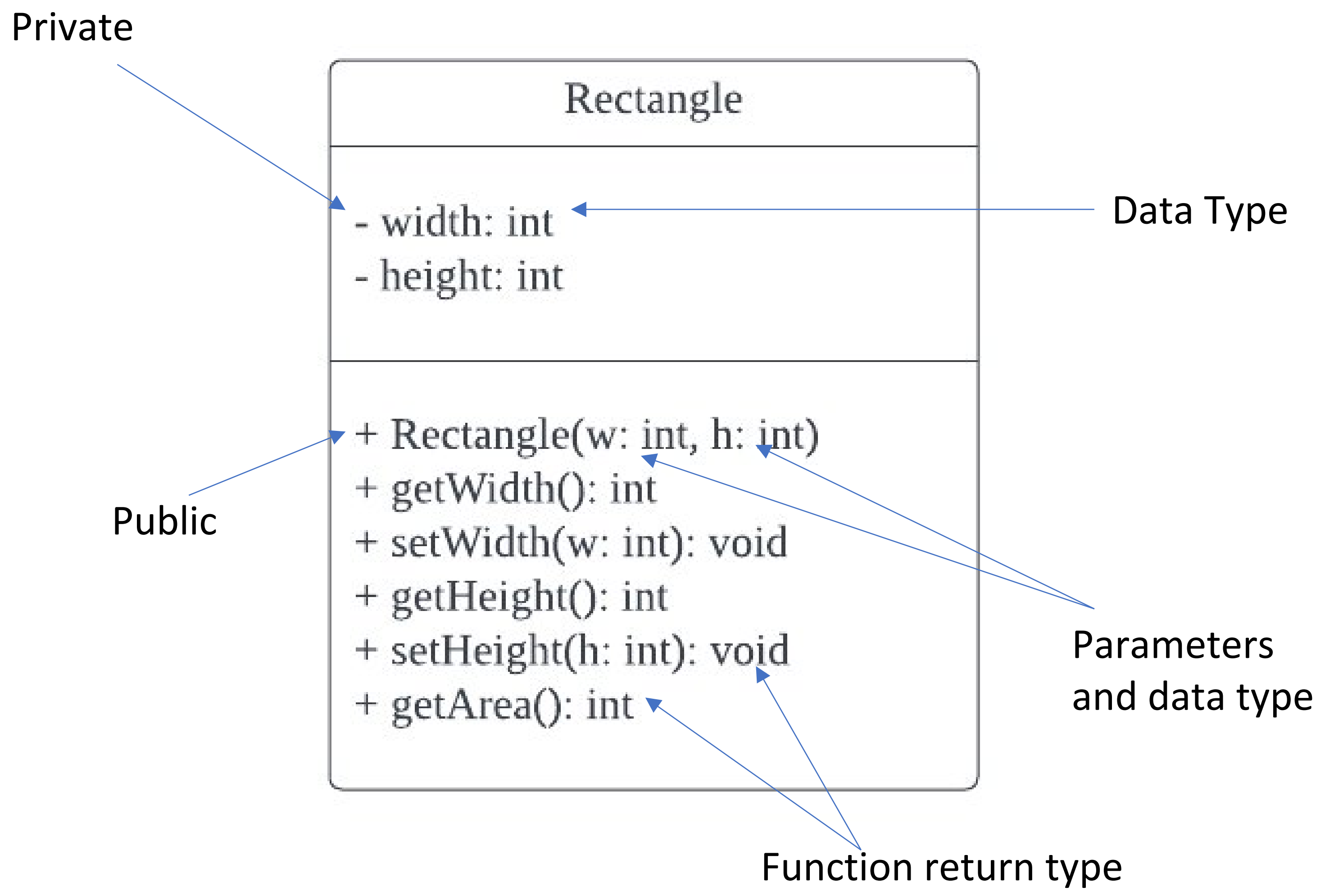

Class name goes here
Member variables are listed here
Member functions are listed here

Notice that the diagram is a box that is divided into three sections.

- The top section is where you write the name of the class.
- The middle section holds a list of the class' member variables.
- The bottom section holds a list of the class' member functions.

UML Class Diagram (A Basic Class)

```
class Rectangle{
    private:
        int width;
        int height;

    public:
        Rectangle(int w, int h){
            width = w;
            height = h;
        }

        int getWidth(){
            return width;
        }

        void setWidth(int w){
            width = w;
        }

        int getHeight(){
            return height;
        }

        void setHeight(int h){
            height = h;
        }

        int getArea(){
            return width * height;
        }

};
```



In-class Exercise 1

Create a UML Class Diagram for the class Employee

```cpp
class Employee{
    private:
        int staffId;
        int salary;

    public:
        Employee(int id, int s){
            staffId = id;
            salary = s;
        }

        int getStaffId(){
            return staffId;
        }

        void setStaffId(int id){
            staffId = id;
        }

        int getSalary(){
            return salary;
        }

        void setSalary(int s){
            salary = s;
        }

        double calcEmployeeEPF(){
            double epf = salary * 0.11;
            return epf;
        }
};
```

UML Class Diagram Relationships

- Relationships in UML diagram are used to represent a connection between various things.

- A relationship is a connection amongst things such as structural, behavioral, or grouping things in the unified modeling language.

- There are different types of standard relationships in UML, for example:
    i. Association
    ii. Composition

UML Class Diagram Association

- Association relationship is denoted using an arrow.
- Association can exist between two or more classes in UML.
- There can be one-one, one-many, many-one, and many-many association present between the association classes.

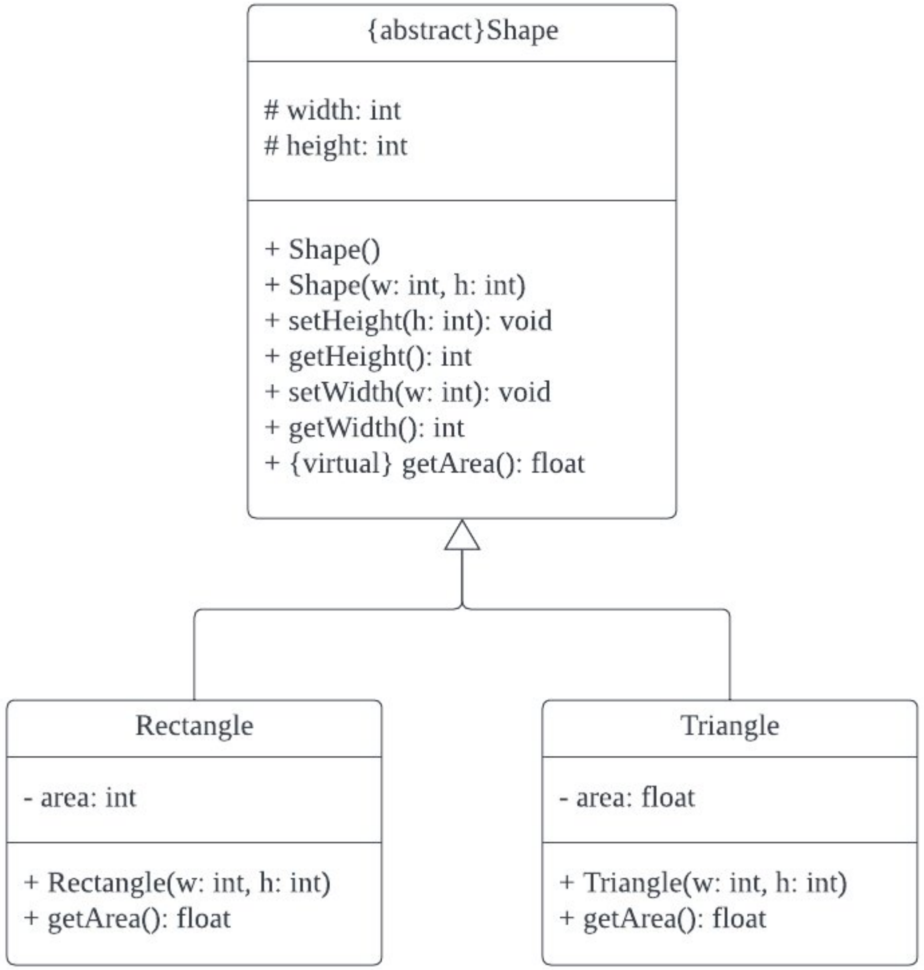
XIAMEN UNIVERSITY MALAYSIA
厦門大學马来西亚分校

UML Class Diagram (Inheritance)

```
class Shape{
    protected:
        int height;
        int width;

    public:
        Shape() = default;

        Shape(int h, int w){
            height = h;
            width = w;
        }

        void setHeight(int h){
            height = h;
        }

        int getHeight(){
            return height;
        }

        void setWidth(int w){
            width = w;
        }

        int getWidth(){
            return width;
        }

        virtual float getArea()=0;

};
```

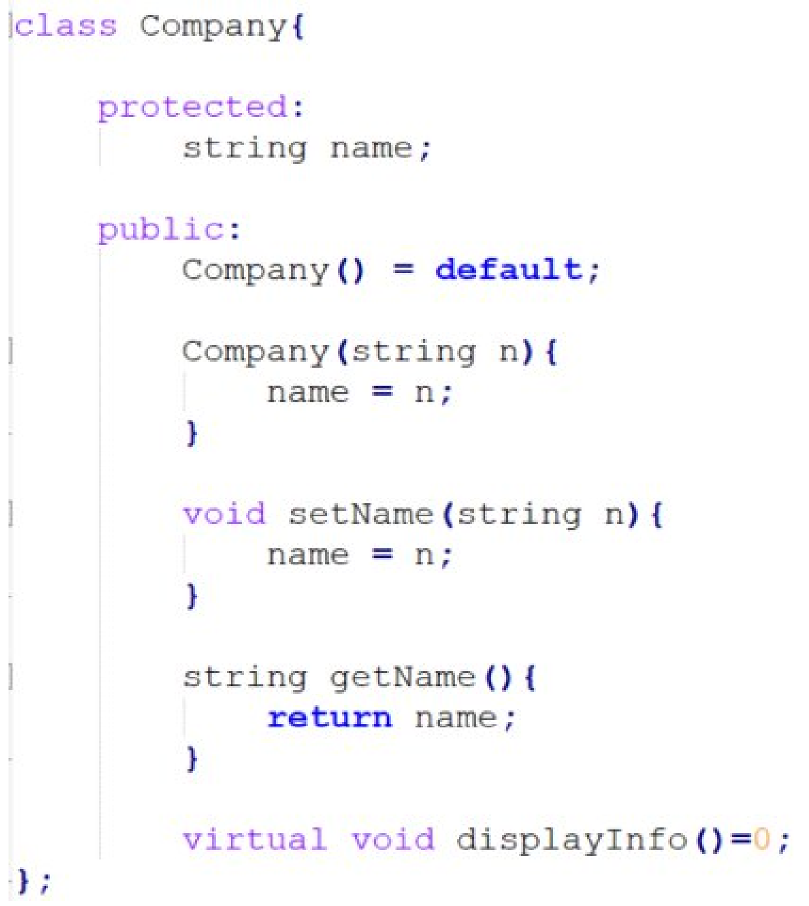
```
class Rectangle: public Shape{
    private:
        int area;

    public:
        Rectangle(int w, int h){
            width = w;
            height = h;
        }

        float getArea(){
            return width * height;
        }

};
```

```
class Triangle: public Shape{
    private:
        float area;

    public:
        Triangle(int w, int h){
            width = w;
            height = h;
        }

        float getArea(){
            return 0.5 * width * height;
        }
};
```

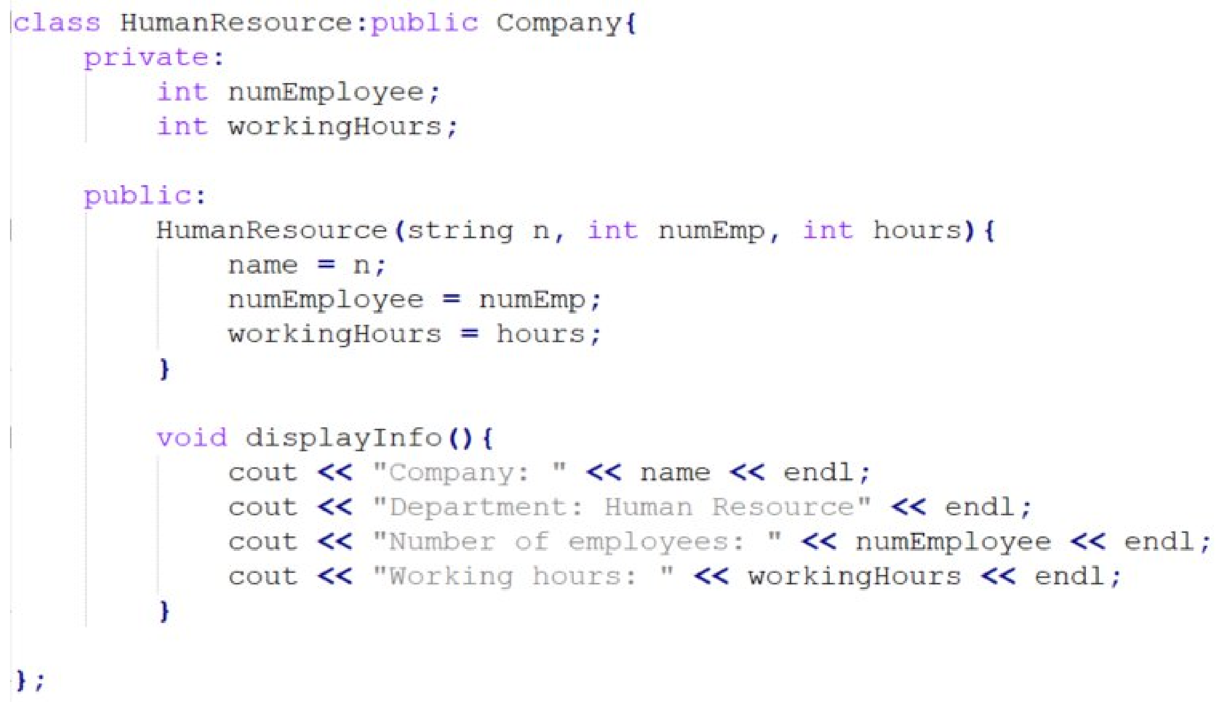


In-class Exercise 2

Create a UML Class Diagram to illustrate the inheritance relationship of the classes below



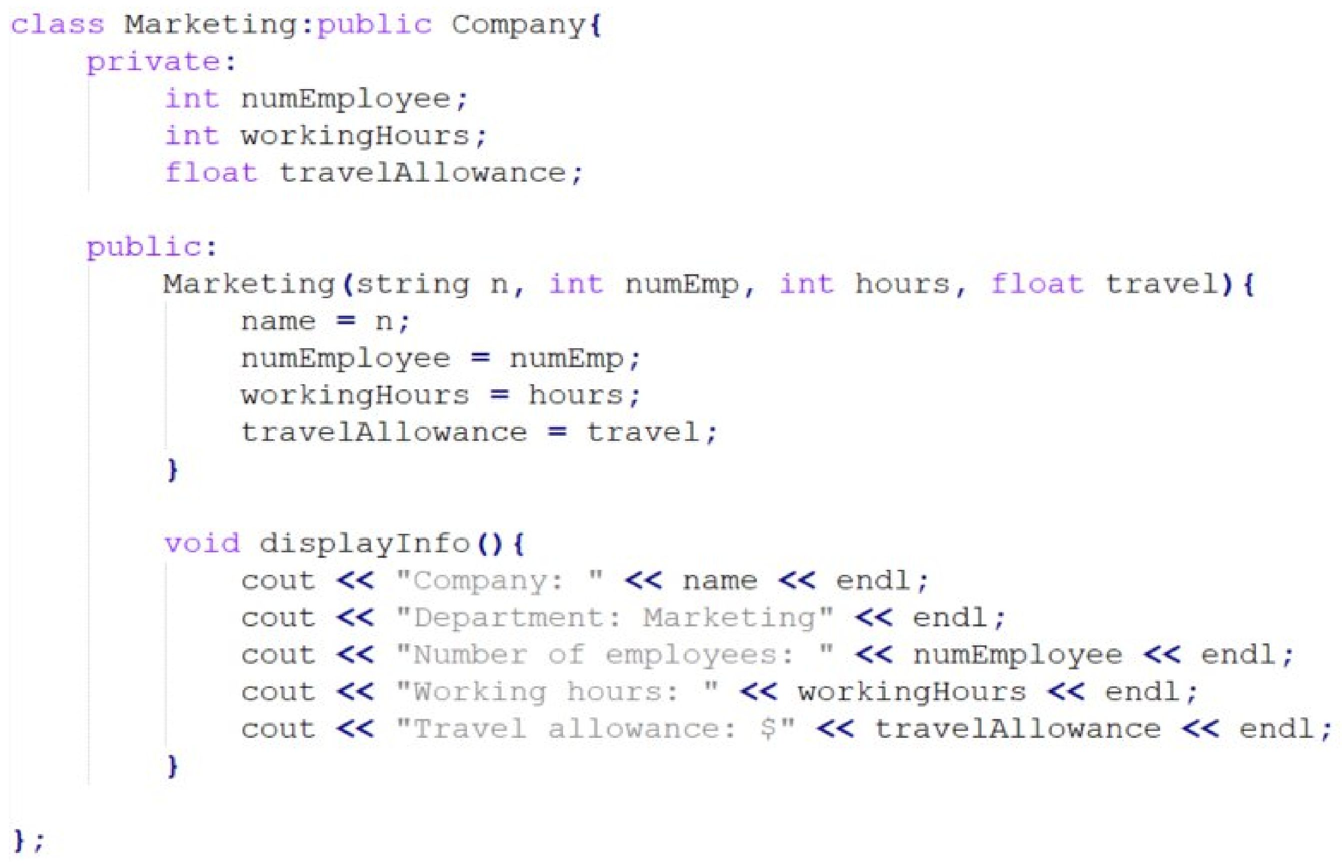
```
class Company{

    protected:
        string name;

    public:
        Company() = default;

        Company(string n){
            name = n;
        }

        void setName(string n){
            name = n;
        }

        string getName(){
            return name;
        }

        virtual void displayInfo()=0;
};
```



```
class HumanResource:public Company{
    private:
        int numEmployee;
        int workingHours;

    public:
        HumanResource(string n, int numEmp, int hours){
            name = n;
            numEmployee = numEmp;
            workingHours = hours;
        }

        void displayInfo(){
            cout << "Company: " << name << endl;
            cout << "Department: Human Resource" << endl;
            cout << "Number of employees: " << numEmployee << endl;
            cout << "Working hours: " << workingHours << endl;
        }

};
```

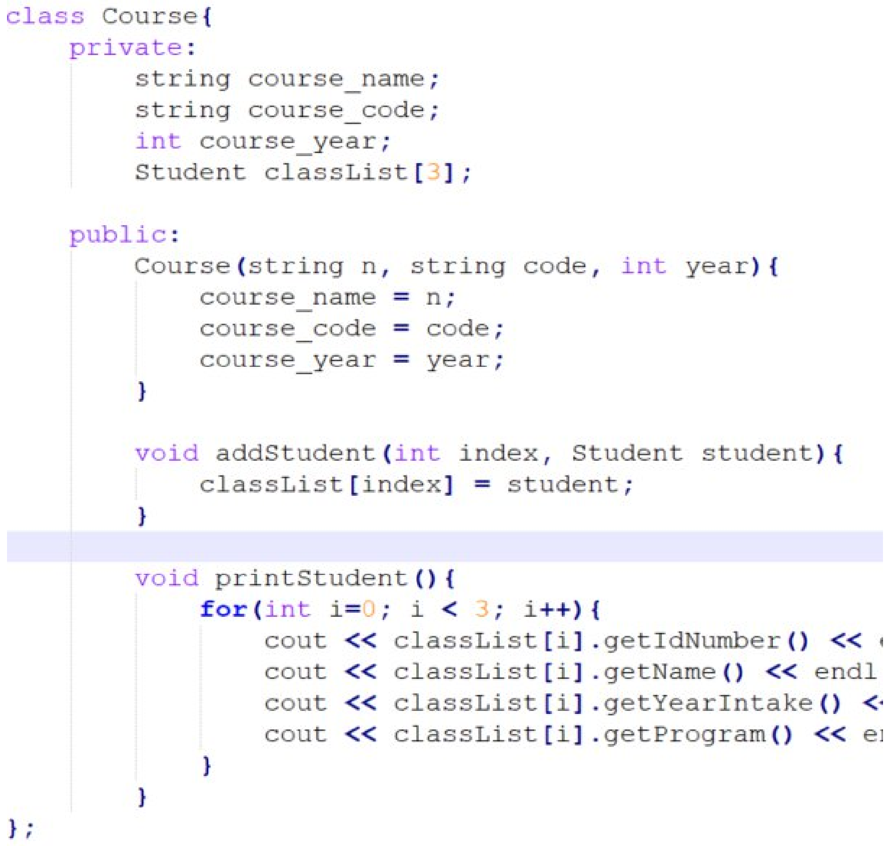
Continue..

In-class Exercise 2

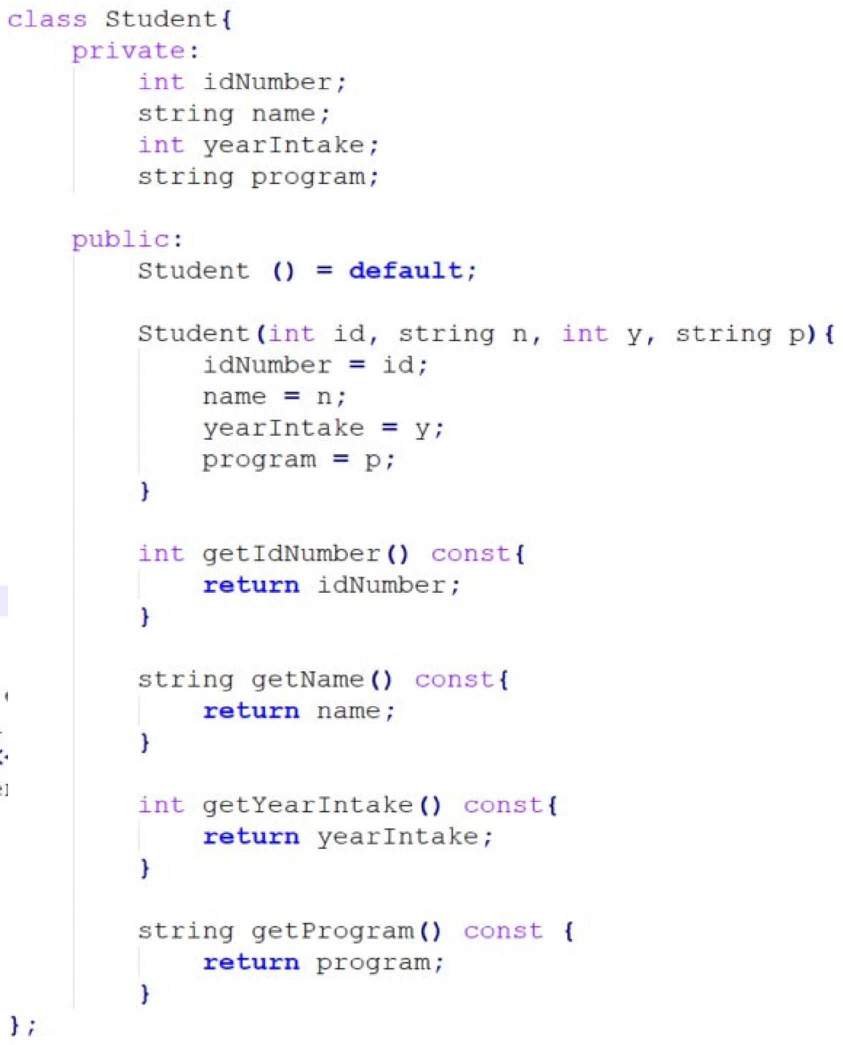


```
class Marketing:public Company{
    private:
        int numEmployee;
        int workingHours;
        float travelAllowance;

    public:
        Marketing(string n, int numEmp, int hours, float travel){
            name = n;
            numEmployee = numEmp;
            workingHours = hours;
            travelAllowance = travel;
        }

        void displayInfo(){
            cout << "Company: " << name << endl;
            cout << "Department: Marketing" << endl;
            cout << "Number of employees: " << numEmployee << endl;
            cout << "Working hours: " << workingHours << endl;
            cout << "Travel allowance: $" << travelAllowance << endl;
        }

};
```

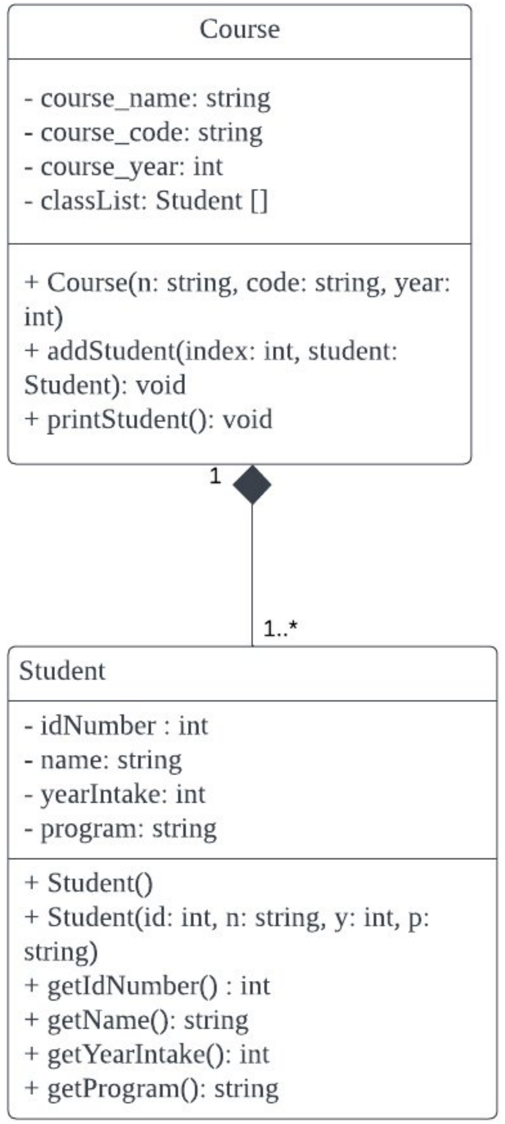
UML Class Diagram (Composition)



```
class Course{
    private:
        string course_name;
        string course_code;
        int course_year;
        Student classList[3];

    public:
        Course(string n, string code, int year){
            course_name = n;
            course_code = code;
            course_year = year;
        }

        void addStudent(int index, Student student){
            classList[index] = student;
        }

        void printStudent(){
            for(int i=0; i < 3; i++){
                cout << classList[i].getIdNumber() <<
                cout << classList[i].getName() << endl
                cout << classList[i].getYearIntake() <
                cout << classList[i].getProgram() << e
            }
        }
};
```

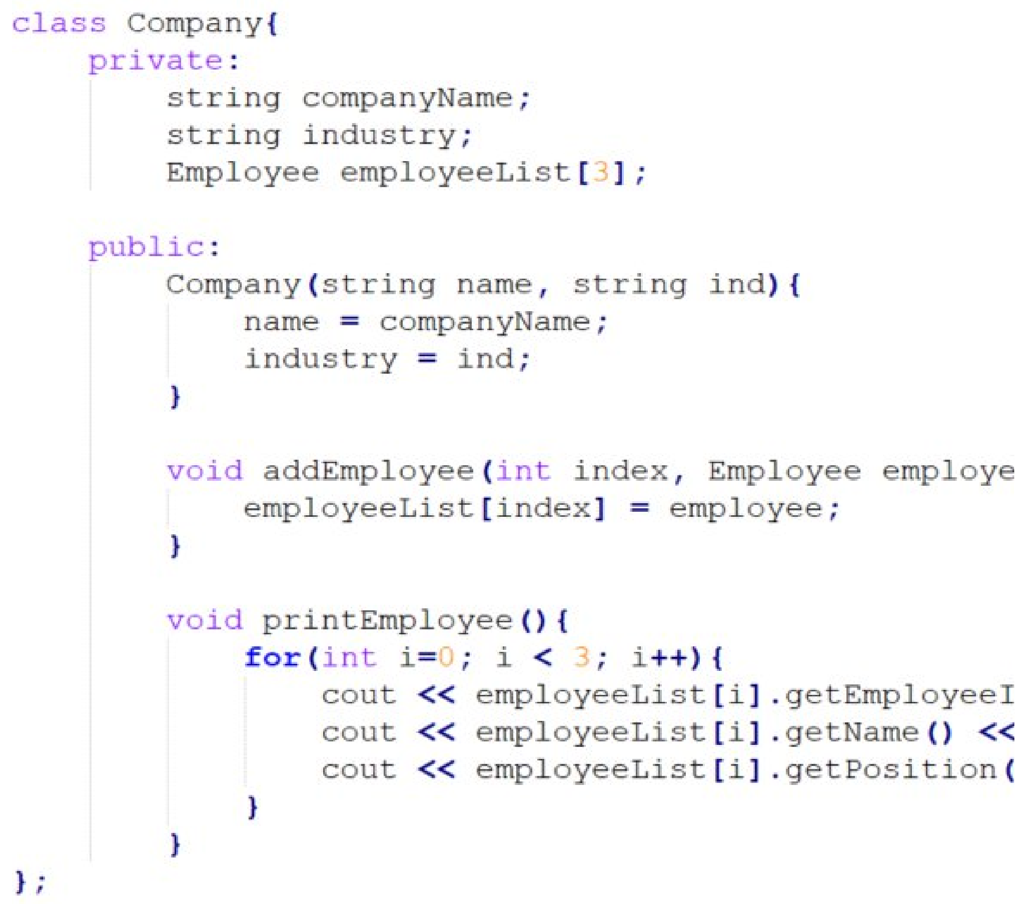


```
class Student{
    private:
        int idNumber;
        string name;
        int yearIntake;
        string program;

    public:
        Student () = default;

        Student(int id, string n, int y, string p){
            idNumber = id;
            name = n;
            yearIntake = y;
            program = p;
        }

        int getIdNumber() const{
            return idNumber;
        }

        string getName() const{
            return name;
        }

        int getYearIntake() const{
            return yearIntake;
        }

        string getProgram() const {
            return program;
        }
};
```

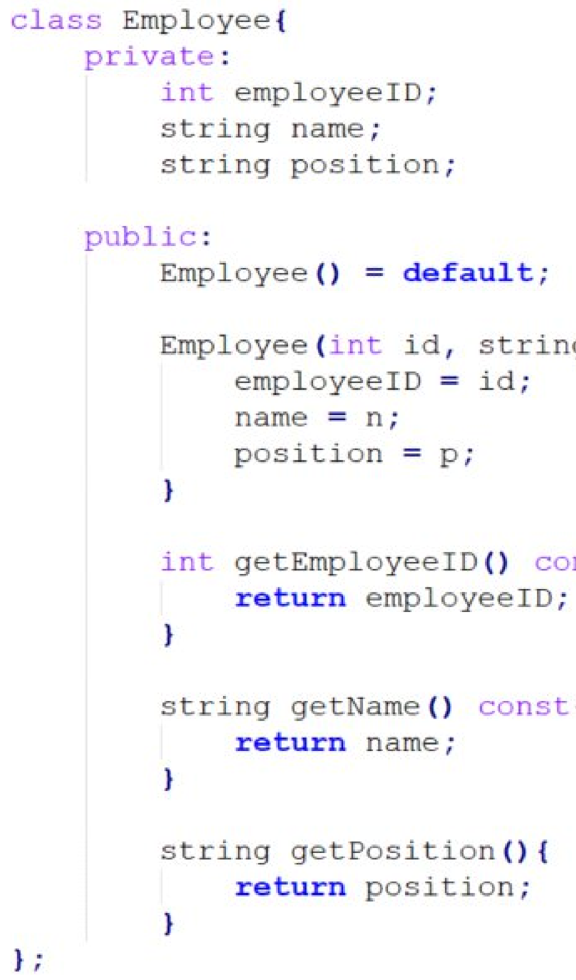


In-class Exercise 3

Create a UML Class Diagram to illustrate the composition relationship of the classes below

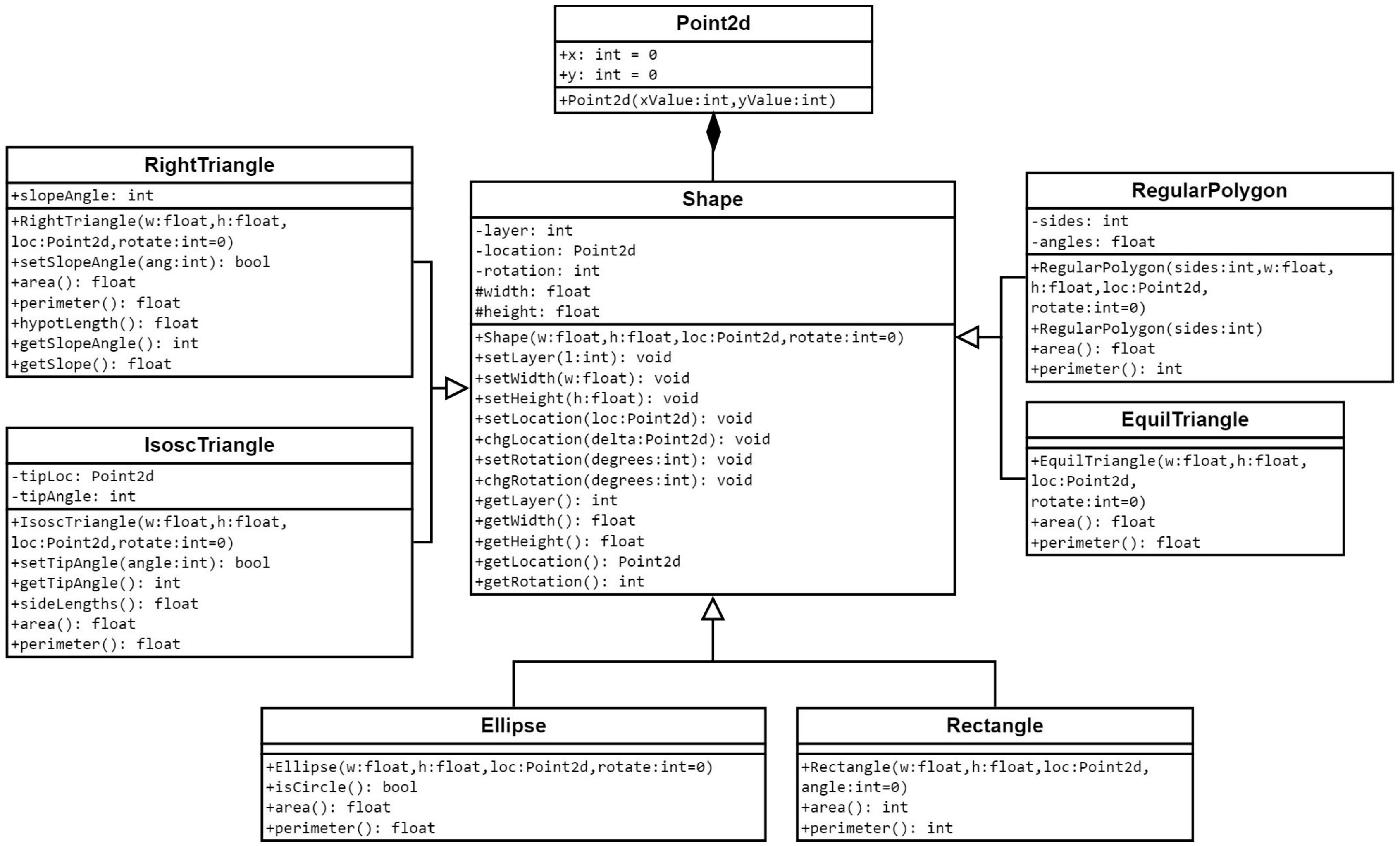


```
class Company{
    private:
        string companyName;
        string industry;
        Employee employeeList[3];

    public:
        Company(string name, string ind){
            name = companyName;
            industry = ind;
        }

        void addEmployee(int index, Employee employe
            employeeList[index] = employee;
        }

        void printEmployee(){
            for(int i=0; i < 3; i++){
                cout << employeeList[i].getEmployeeI
                cout << employeeList[i].getName() <<
                cout << employeeList[i].getPosition(
            }
        }
};
```



```
class Employee{
    private:
        int employeeID;
        string name;
        string position;

    public:
        Employee() = default;

        Employee(int id, string n, string p){
            employeeID = id;
            name = n;
            position = p;
        }

        int getEmployeeID() const{
            return employeeID;
        }

        string getName() const{
            return name;
        }

        string getPosition(){
            return position;
        }
};
```

UML Class Diagram



```
Point2d
+x: int = 0
+y: int = 0
+Point2d(xValue:int,yValue:int)
```

```
RightTriangle
+slopeAngle: int
+RightTriangle(w:float,h:float,
loc:Point2d,rotate:int=0)
+setSlopeAngle(ang:int): bool
+area(): float
+perimeter(): float
+hypotLength(): float
+getSlopeAngle(): int
+getSlope(): float
```

```
IsoscTriangle
-tipLoc: Point2d
-tipAngle: int
+IsoscTriangle(w:float,h:float,
loc:Point2d,rotate:int=0)
+setTipAngle(angle:int): bool
+getTipAngle(): int
+sideLengths(): float
+area(): float
+perimeter(): float
```

```
Shape
-layer: int
-location: Point2d
-rotation: int
#width: float
#height: float
+Shape(w:float,h:float,loc:Point2d,rotate:int=0)
+setLayer(l:int): void
+setWidth(w:float): void
+setHeight(h:float): void
+setLocation(loc:Point2d): void
+chgLocation(delta:Point2d): void
+setRotation(degrees:int): void
+chgRotation(degrees:int): void
+getLayer(): int
+getWidth(): float
+getHeight(): float
+getLocation(): Point2d
+getRotation(): int
```

```
RegularPolygon
-sides: int
-angles: float
+RegularPolygon(sides:int,w:float,
h:float,loc:Point2d,
rotate:int=0)
+RegularPolygon(sides:int)
+area(): float
+perimeter(): int
```

```
EquilTriangle
+EquilTriangle(w:float,h:float,
loc:Point2d,
rotate:int=0)
+area(): float
+perimeter(): float
```

```
Ellipse
+Ellipse(w:float,h:float,loc:Point2d,rotate:int=0)
+isCircle(): bool
+area(): float
+perimeter(): float
```

```
Rectangle
+Rectangle(w:float,h:float,loc:Point2d,
angle:int=0)
+area(): int
+perimeter(): int
```

All the best