Java™ Education & Technology Services

Object Oriented programming Using C++

Class Relations

Class Relations

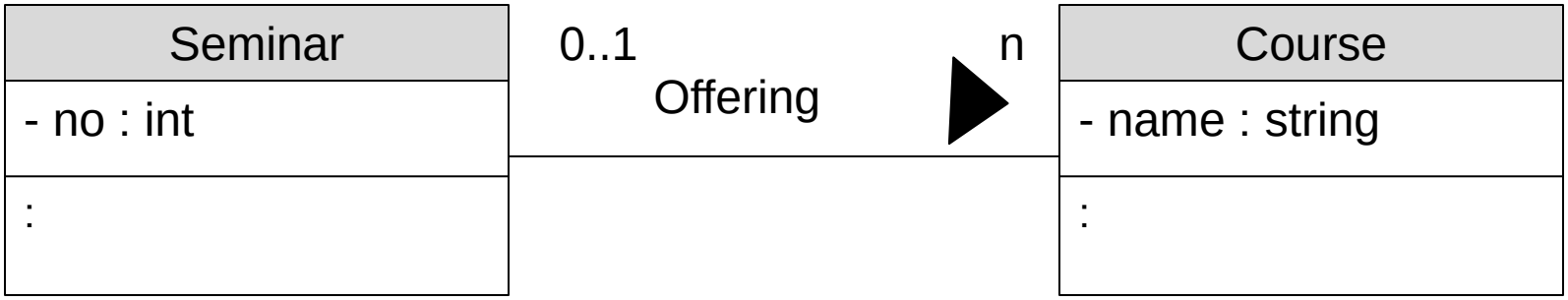
I. Association

- Is a relation between two classes.
- It allows one object instance to "use" another to perform an action on its behalf.
- Two objects have their own lifecycle and there is no owner .
- Both can created and deleted independently.
- **Examples:**
  - Students are "on waiting list " for a seminar.
  - Professors "instruct" a seminar.
  - Seminar is an "offering" of courses

Class Relations

I. Association

- **Examples:**
  - Seminar is an "offering" of courses



  - How many courses does seminar offer? None, one , many?

0..1	1	0..*	1..*	n	0..n	1..n

Class Relations

I. Association

- **Examples:**
  - class **Course**{

```
:
Course();
}
```

  - class **Seminar**{

```
    Course * x;
public:
    Seminar();
    void offer ( Course *c);
}
```

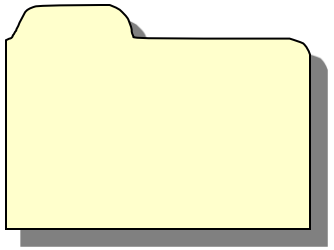
```
int main() {
        Course c1;
        Seminar s1;
        s1.offer(&c1);
```

ClassA may be linked to ClassB in order to show that one of its methods include a reference to the another one as a parameter

Class Relations

I. Association

- Create a folder called " links" .
- Create shortcuts inside this folder to Google , YouTube and Facebook
- Delete this folder.
- Open your browser to check Google , YouTube and Facebook are still exit or not



- **No owner :** Association is a relation where all the object have different lifecycle.

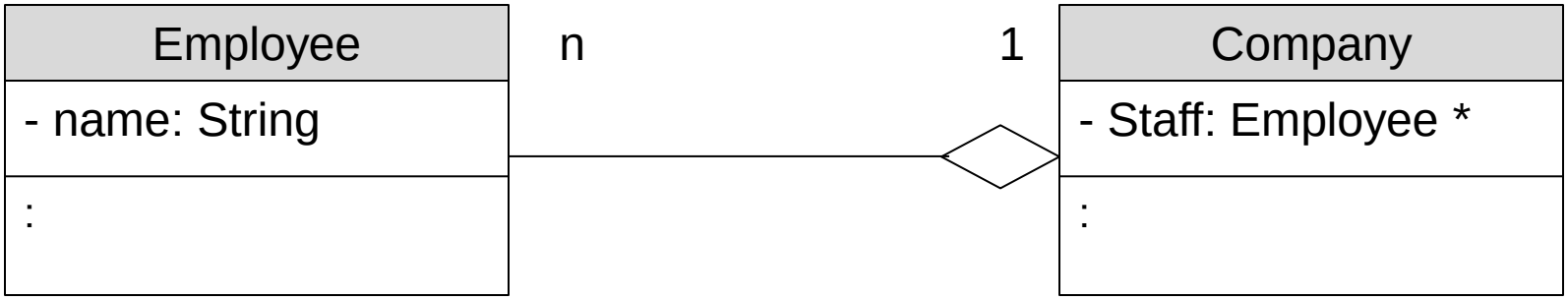
Class Relations

II. Aggregation

- Is a special type of association. [strong Association]
- Object of one class " has" an object of the another one
- Two objects have their own lifecycle and there is one owner at atime .
- Both can created and deleted independently.
- **Examples:**
  - Room contains many Tables.

II. Aggregation

- **Examples:**
  - A single Employee can not belong to multiple companies .



II. Aggregation

- **Examples:**
  - class **Employee**{

```
:
Employee();
}
```

  - class **Company**{

```
    Employee* staff;
public:
    Company (Employee *x){
        staff= x;
    }
}
```

```
int main() {
    Employee emp;
    Company c1(&emp);
```

ClassA may be linked to **ClassB** in order to show that one of its constructors include a reference to the another one as a parameter

II. Aggregation

- Create a file called " file.txt" .
- Make a simple Application to open the file.txt in rw mode.
- Run an instance of the application.
- Try to run another instance of this application.
- Sure the application and the file has a separated lifecycles .
- However this file can be opened only by one application. One parent at a time.

- **All Aggregations are associations but not all associations are aggregations**

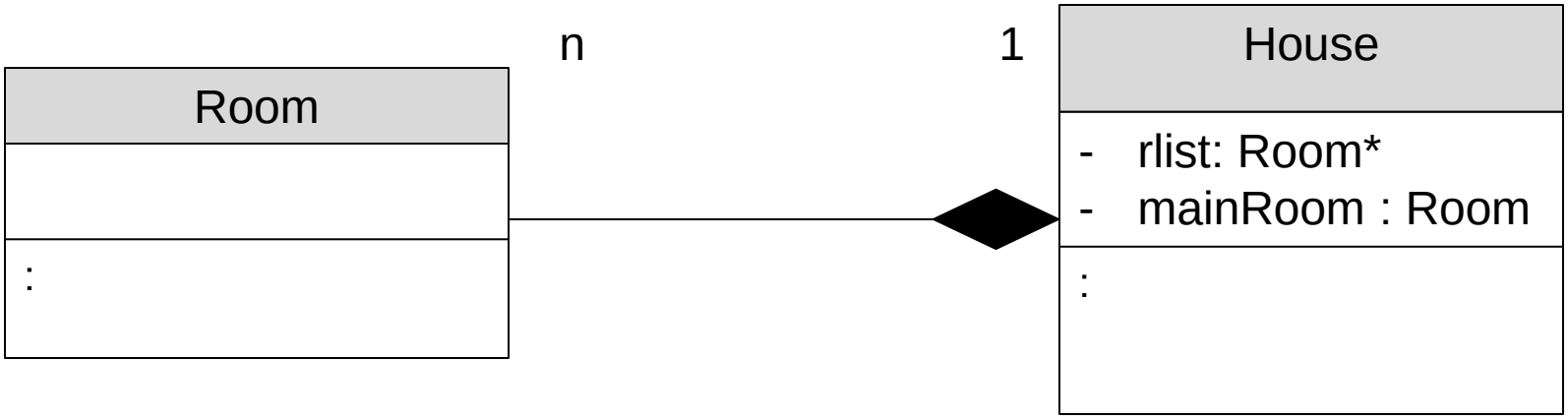
Class Relations

III. Composition

- Is a Strong type of Aggregation.
- Part object does not have its own lifecycle.
- If the whole object gets deleted , all of its parts will also deleted.
- Typically use normal member variables
- Can use pointer values if the composition class automatically handle allocation / de-allocation of these values
- **Examples:**
  - House contains multiple Rooms.
  - Any Room can not be belong to two Houses
  - If we delete the house , all rooms will be deleted

III. Composition

- **Examples:**



III. Composition

- **Examples:**
  - class **Room**{

```
:
Room();
}
```

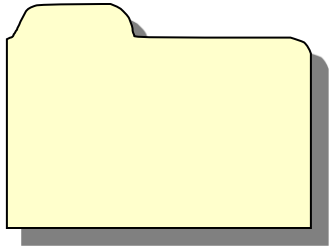
  - class **House**{

```
    Room* rlist;
    Room mainRoom;
public:
    House():mainRoom() {
        rlist= new Room[4];
    }
}
```

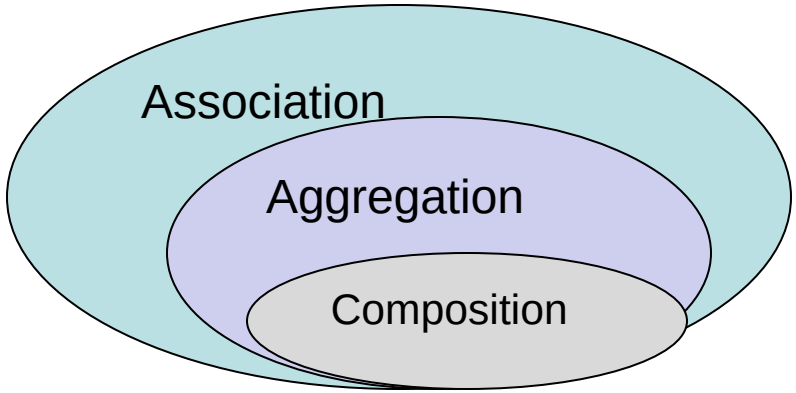
```
int main() {
        House H;
```

III. Composition

- Create a folder called " mainFolder" .
- Create two word files inside this folder to todoList and contacts
- Delete this folder.

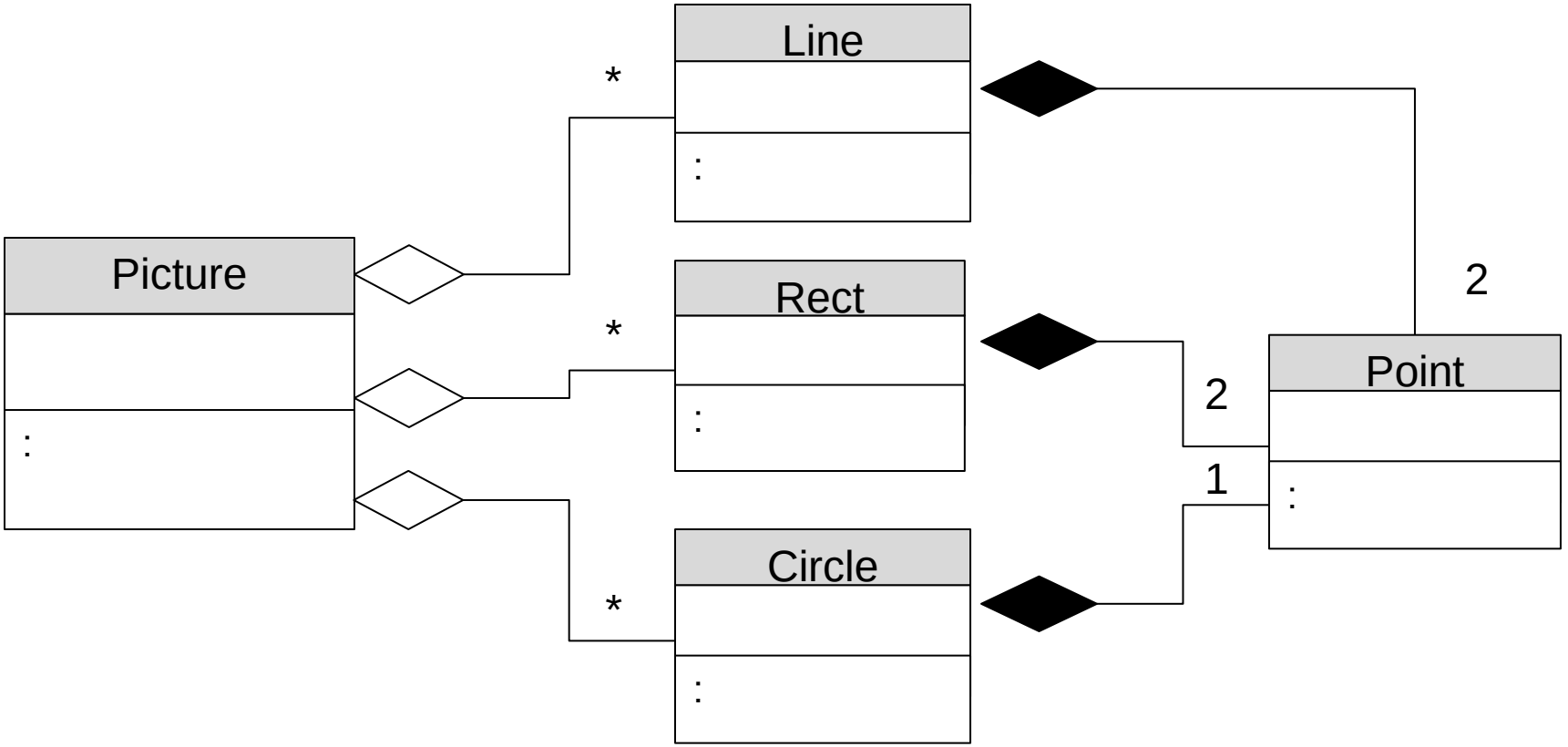


- **All Compositions are Aggregations but not all Aggregations are Compositions**

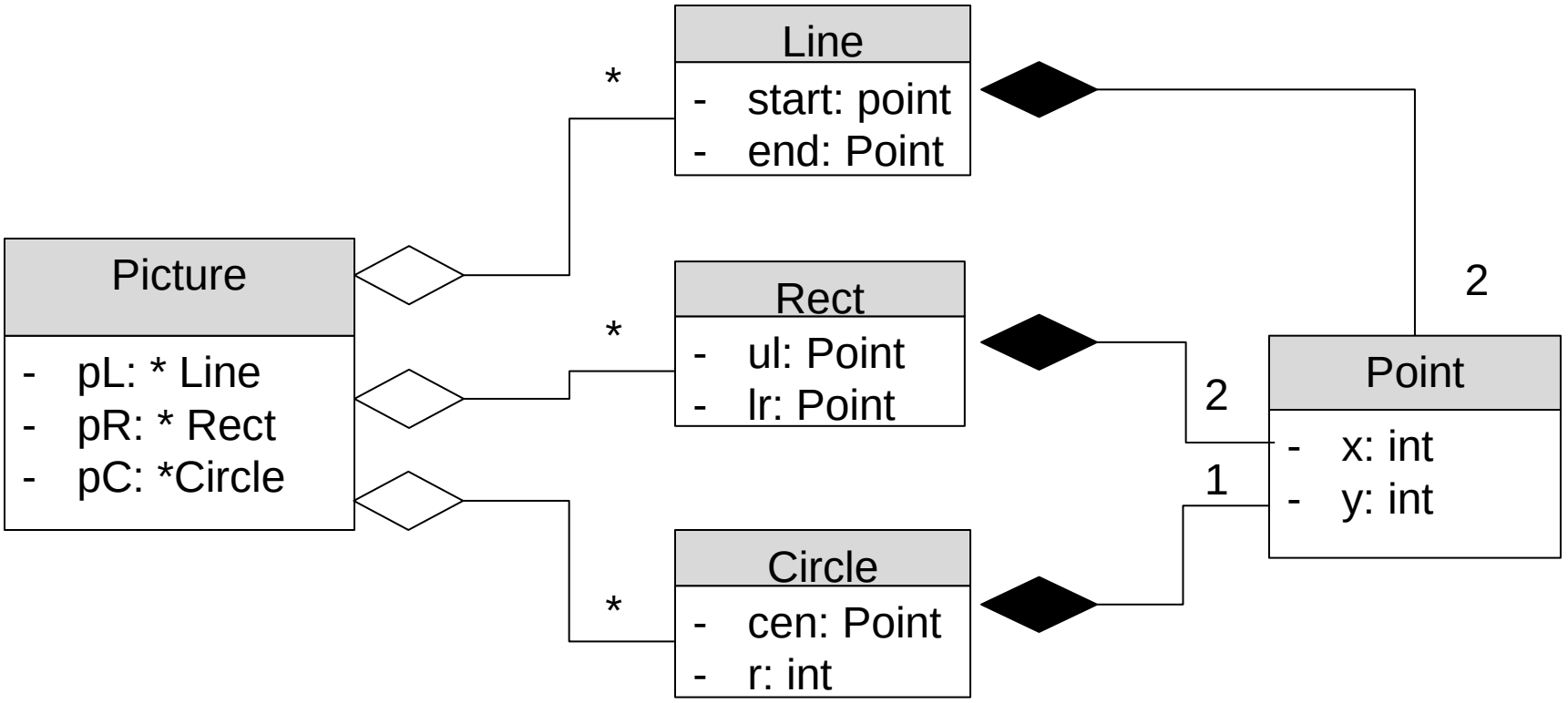


Example

- Create an application to draw a picture of lines , Rectangles and circles.
- Point is the main component of all shapes



Example



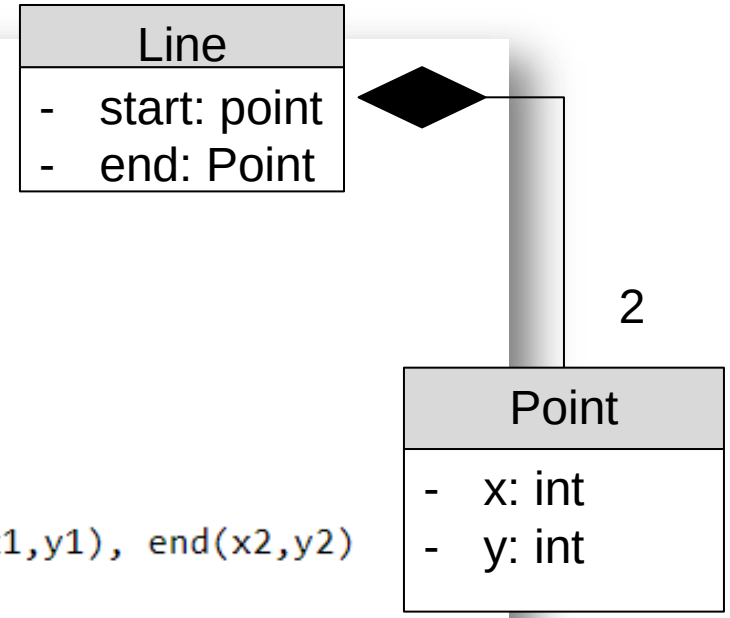
Example

```
class Point
{
    int x ;
    int y ;

  public:
    Point();
    Point(int m, int n);

    void setX(int m);
    void setY(int n);

    int getX();
    int getY();

};
```

Point
- x: int - y: int

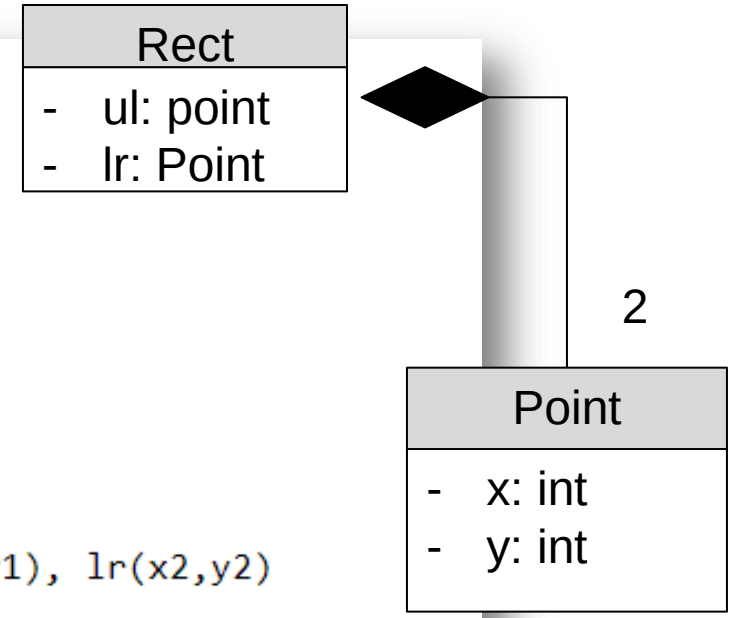
Example

```
class Line
{

    Point start;
    Point end;

  public:
    Line() : start(), end()
    { cout<<"At line Const.";   }

    Line(int x1, int y1, int x2, int y2) : start(x1,y1), end(x2,y2)
    { cout<<"At line Const.";   }

    void draw()
    {
        line(start.getX(), start.getY(), end.getX(), end.getY()) ;
    }
};
```



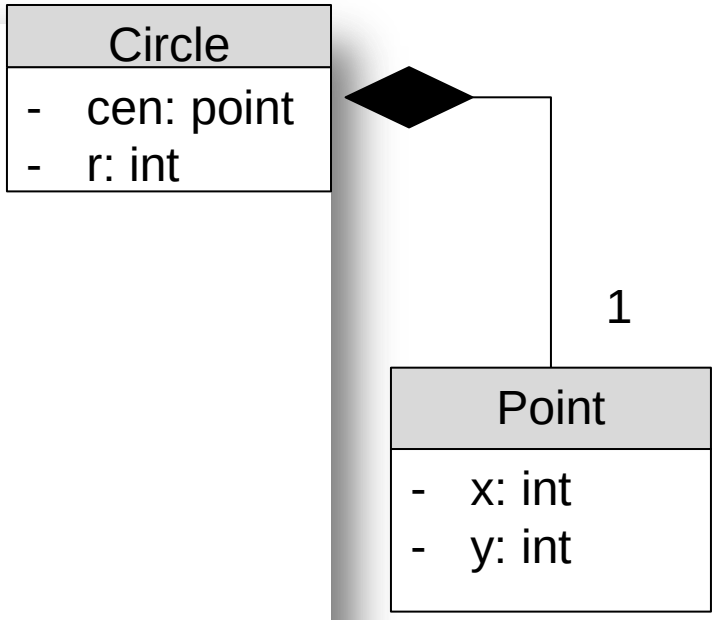
Example

```
class Rect
{
  private:
    Point ul;
    Point lr;

  public:
    Rect() : ul(), lr()
    { cout<<"At Rect Const.";   }

    Rect(int x1, int y1, int x2, int y2) : ul(x1,y1), lr(x2,y2)
    { cout<<"At Rect Const.";   }

    void draw()
    {
        rectangle(ul.getX(), ul.getY(), lr.getX(), lr.getY()) ;
    }
};
```



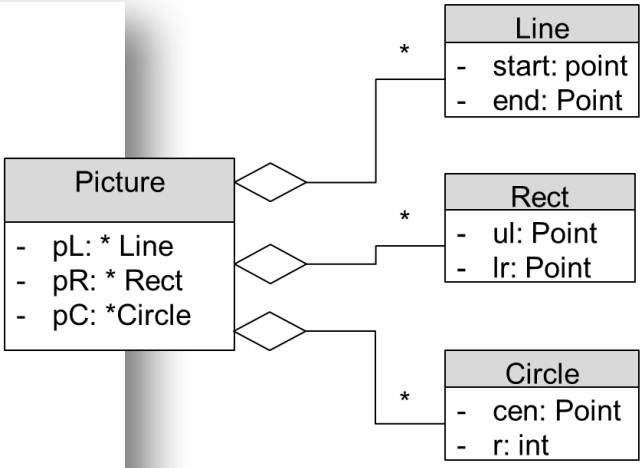
Example

```
class Circle
{
  private :
    Point center;
    int   radius;

  public :
    Circle() : center()
    {
        radius = 0 ; cout<<"At Circle Const.";
    }

    Circle(int m, int n, int r) : center(m,n)
    {
        radius = r ; cout<<"At Circle Const.";
    }
    void draw()
    {
        circle(center.getX(), center.getY(), radius) ;
    }
};
```



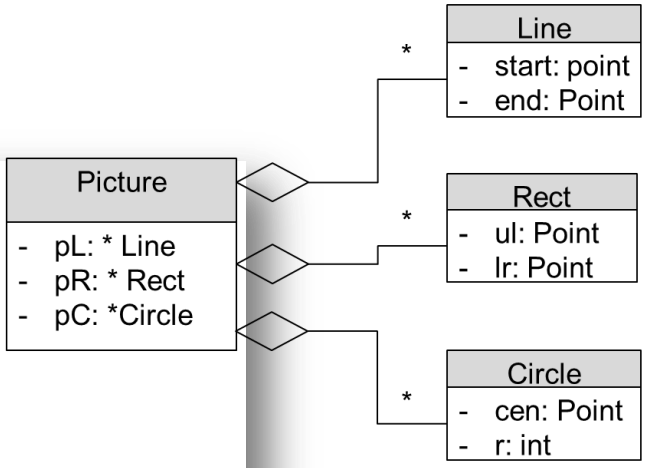
Example

```
class Picture
{
    int cNum ;
    int rNum ;
    int lNum ;
    Circle *pCircles;
    Rect *pRects;
    Line *pLines;
  public :
    Picture()
    {
        cNum=0;
        rNum=0;
        lNum=0;
        pCircles = NULL ;
        pRects = NULL ;
        pLines = NULL ;
    }

    Picture(int cn, int rn, int ln, Circle *pC, Rect *pR, Line *pL)
    {
        cNum = cn;
        rNum = rn;
        lNum = ln;
        pCircles = pC ;
        pRects = pR ;
        pLines = pL ;
    }
```



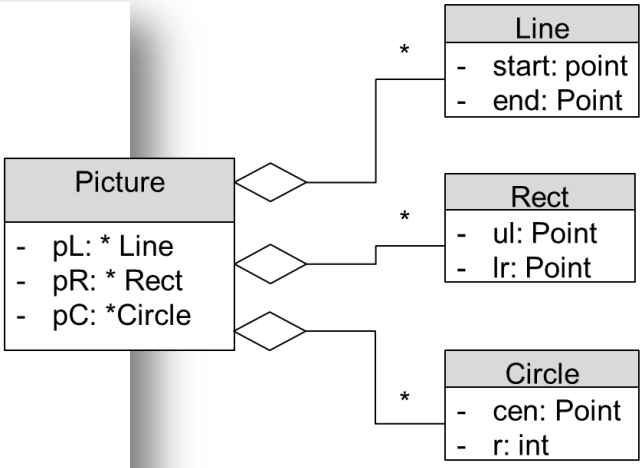
Example

```
    void setCircles(int, Circle *);
    void setRects(int, Rect *);
    void setLines(int, Line *);
    void paint();
};
```



Example

```
void Picture::setLines(int ln, Line * lptr)
{
    lNum = ln ;
    pLines = lptr ;
}

void Picture::paint()
{
    int i;
    for(i=0; i<cNum ; i++)
    {
        pCircles[i].draw();
    }

    for(i=0 ; i<rNum ; i++)
    {
        pRects[i].draw();
    }

    for(i=0 ; i<lNum; i++)
    {
        pLines[i].draw();
    }
}
```



Example

```
//simple main( )
int main()
{
    // Graphic Mode
        Picture myPic;

        Circle cArr[3]={Circle(50,50,50), Circle(200,100,100), Circle(420,50,30)};
        Rect rArr[2]={Rect(30,40,170,100), Rect(420,50,500,300)};
        Line lArr[2]={Line(420,50,300,300), Line(40,500,500,400)};

        myPic.setCircles(3,cArr) ;
        myPic.setRects(2,rArr) ;
        myPic.setLines(2,lArr) ;

        myPic.paint() ;

return 0;
}
```

Lab Exercise

Lab Exercise

- **1st Assignment :**
  - Create an application to draw a nice picture of lines , Rectangles and circles with colors.
  - ☺