Class relations: has-a

Victor Eijkhout, Susan Lindsey

Fall 2023
`last formatted: September 12, 2023`

1. Has-a relationship

A class usually contains data members. These can be simple types or other classes. This allows you to reflect relations between things you are modeling.

```
class Person {
  string name;
  ....
};
class Course {
private:
  Person the_instructor;
  int year;
};`
```

This is called the has-a relation:

Course has-a *Person*

2. Literal and figurative has-a

A line segment has a starting point and an end point. *LineSegment* code design:

Store both points:

```
class Segment {
private:
  Point p_start,p_end;
public:
  Point end_point() {
    return p_end; };
}
int main() {
  Segment seg;
  Point somepoint =
    seg.end_point();
```

or store one and derive the other:

```
class Segment {
private:
  Point starting_point;
  float length,angle;
public:
  Point end_point() {
    /* some computation
       from the
       starting point */ };
}
```

Implementation vs API: implementation can be very different from user interface.

3. Constructors in has-a case

Class for a person:

```
class Person {
private:
  string name;
public:
  Person( string name ) {
    /* ... */
  };
};
```

Class for a course, which contains a person:

```
class Course {
private:
  Person instructor;
  int enrollment;
public:
  Course( string instr,int n )
    {
    /* ???? */
  };
};
```

Declare a `Course` variable as: `Course("Eijkhout",65);`

4. Constructors in the has-a case

Possible constructor:

```
Course( string teachername,int nstudents ) {
  instructor = Person(teachername);
  enrollment = nstudents;
};
```

Preferred:

```
Course( string teachername,int nstudents )
  : instructor(Person(teachername)),
    enrollment(nstudents) {
};
```

5. Rectangle class

Rectangle with sides parallel to the x/y axes.
Two designs possible. For the function:

```
float Rectangle::area();
```

it is most convenient to store width and height;
for inclusion testing:

```
bool Rectangle::contains(Point);
```

it would be convenient to store bottomleft/topright points.

Exercise 1

1. Make a class `Rectangle` (sides parallel to axes) with a constructor:

   ```
   Rectangle(Point botleft,float width,float height);
   ```

   The logical implementation is to store these quantities. Implement methods:

   ```
   float area(); float rightedge_x(); float topedge_y();
   ```

   and write a main program to test these.
2. Add a second constructor

   ```
   Rectangle(Point botleft,Point topright);
   ```

   Can you figure out how to use member initializer lists for the constructors?

Optional exercise 2

Make a copy of your solution of the previous exercise, and redesign your class so that it stores two `Point` objects. Your main program should not change.