Class relations: has-a

Victor Eijkhout, Susan Lindsey

Fall 2021

last formatted: September 20, 2021

1. Has-a relationship

A class usually contains data members. These can be simple types or other classes. This allows you to make structured code.

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

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.

A Segment class can store those points:

```
class Segment {
private:
  Point starting_point,
    ending_point;
public:
  Point get_the_end_point() {
    return ending_point; };
}
int main() {
  Segment somesegment;
  Point somepoint =
    somesegment.
    get_the_end_point();
```

or store one and derive the other:

```
class Segment {
private:
  Point starting_point;
  float length,angle;
public:
  Point get_the_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 Person {
private:
  string name;
public:
  Person( string name ) {
    /* ... */
  };
};
```

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

Use as `Course("Eijkhout",65);`

4. Constructors in the has-a case

Possible constructor:

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

Preferred:

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

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.

5. Polymorphism in constructors

You have to decide what to store and what to derive, but you can construct two ways:

```
class Segment {
private:
 // up to you how to implement!
public:
  Segment( Point start,float length,float angle )
    { .... }
  Segment( Point start,Point end ) { ... }
```

Advantage: with a good API you can change your mind about the implementation without changing the calling code.