Class relations: has-a

Victor Eijkhout, Susan Lindsey

COE 322 Fall 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.

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. Default Constructors in initialization

```
class Inner { /* ... */ };
class Outer {
private:
  Inner inside_thing;
```

Two possibilities for constructor:

```
Outer( Inner thing )
: inside_thing(thing) {};
```

The *Inner* object is copied during construction of *Outer* object.

```
Outer( Inner thing ) {
  inside_thing = thing;
};
```

The *Outer* object is created, including construction of *Inner* object, then the argument is copied into place: $\Rightarrow$ needs default constructor on *Inner*.

Exercise 1

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

   ```
   Rectangle(Point bl,float w,float h);
   ```

   The logical implementation is to store these quantities. Implement methods

   ```
   float area(); float rightedge(); float topedge();
   and write a main program to test these.
   ```

2. Add a second constructor

   ```
   Rectangle(Point bl,Point tr);
   ```

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

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

4. 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.