Class relations: has-a

Victor Eijkhout and Charlie Dey

spring 2017

Literal and figurative has-a

Compare:

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

Versus:

```
class Segment {
private:
  Point starting_point;
  float length,angle;
}
```

Implementation vs API.

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!

Exercise 1

Make a class LinearFunction with two constructors:

```
LinearFunction( Point input_p2 );
LinearFunction( Point input_p1,Point input_p2 );
```

where the first stands for a line through the origin.
Implement again the evaluate function so that

```
LinearFunction line(p1,p2);
cout << "Value at 4.0: " << line.evaluate_at(4.0) << endl;
```

Exercise 2

Make a class `Rectangle` (sides parallel to axes) with two constructors:

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

and functions

```
float area(); float width(); float height();
```

Let the `Rectangle` object store two `Point` objects.

Then rewrite your exercise so that the `Rectangle` stores only one point (say, lower left), plus the width and height.