Exercise: Graphical Shapes

In this exercise the beginning class library from the slides should be completed. Implement a Point class and a Rectangle class with width and height properties. Also implement a Composite class that can contain Shape classes. The print() method of the Composite class should iterate over the shapes and call their print() methods individually. The composite class should have methods for adding, removing, clearing, capacity-queries and accessing elements. The Composite object is responsible for deleting objects contained within it. Delete the objects in the destructor. Use a vector<Shape*> internally in the Composite class for storing the shape pointers and use heap allocated objects as shown in the following example:

```
Rectangle* rectangle = new Rectangle(0.0, 0.0);
```

For larger object-hierarchies heap allocation is often the only option. Members of heap objects are accessed with the -> operator.

The Shape and Circle classes can be downloaded at:

https://dl.dropbox.com/u/2888586/advprog_exercises/shapes.tar.gz

A typical main program using the library could look something like the code below:

```
int main()
{
    Composite* composite = new Composite(0.0, 0.0);

    cout << "adding objects ---" << endl;

    composite->add(new Point(0.0, 0.0));
    composite->add(new Circle(1.0, 0.0, 2.0));
    composite->add(new Rectangle(0.0, 1.0, 2.0, 1.0));

    for (int i=0; i<composite->count(); i++)
        composite->at(i)->print();

    composite->remove(composite->at(0));

    cout << composite->count() << endl;

    composite->clear();

    cout << composite->count() << endl;

    delete composite;
}
```