Cylinder.h

```
#ifndef CYLINDER_H_INCLUDED
#define CYLINDER_H_INCLUDED

#include "Polyhedron.h"

class Cylinder : public Polyhedron {
    private:
        double height;
        double radius;

    public:
        /**
         * Default Constructor
         */
        Cylinder();

        /**
         * Construct a cylinder with specified height and radius
         *
         * @param r desired radius
         * @param h desired height
         */
        Cylinder(double r, double h);

        /**
         * Destructor
         */
        ~Cylinder() = default;

        // Use the compiler generated version
        Cylinder(const Cylinder& src) = default;

        // Use the compiler generated version
        Cylinder& operator=(const Cylinder& rhs) = default;

        /**
         * Retrieve the radius
         */
        double getRadius() const;

        /**
         * Retrieve the height
         */
```

```
        double getHeight() const;

        /**
         * Update the radius
         */
        void setRadius(double r);

        /**
         * Update the height
         */
        void setHeight(double h);


        /**
         * Compute and return the diameter
         */
        double getDiameter() const;

        virtual Polyhedron* clone() const;
        virtual void read(std::istream& ins);
        virtual void display(std::ostream& outs) const;
        virtual void scale(double scalingFactor);
};

//-----------------------------------------------------------------------------
inline
double Cylinder::getRadius() const
{
    return this->radius;
}

//-----------------------------------------------------------------------------
inline
double Cylinder::getHeight() const
{
    return this->height;
}

//-----------------------------------------------------------------------------
inline
double Cylinder::getDiameter() const
{
    return 2 * radius;
}
```

```
#endif
```

Cylinder.cpp

```cpp
#include "Cylinder.h"

//-----------------------------------------------------------------------------
Cylinder::Cylinder()
    :Cylinder(1, 1)
{
}

//-----------------------------------------------------------------------------
Cylinder::Cylinder(double r, double h)
    :Polyhedron("Cylinder"),
     height(h),
     radius(r)
{
    double d = this->getDiameter();
    boundingBox.setUpperRightVertex(d, d, height);

    // **Note** the upper-right vertex of the bounding box must be set to
    // (diameter, diameter, height).
    //
    // The z-axis is treated as the height of the
    // cylinder and the x-y plane is the "floor" where the circular face of the
    // cylinder rests.
}

//-----------------------------------------------------------------------------
void Cylinder::read(std::istream& ins)
{
    // Implement this function
    ins >> this->height >> this->radius;
    double d = this->getDiameter();
    boundingBox.setUpperRightVertex(d, d, height);
}

//-----------------------------------------------------------------------------
void Cylinder::display(std::ostream& outs) const
{
    // Implement this function
    outs << "[Cylinder] "
    << boundingBox.getUpperRightVertex()
    << "->" << "Radius: " << radius << " Height: " << height;

}
```

```
//-----------------------------------------------------------------------------
void Cylinder::scale(double scalingFactor)
{
    radius *= scalingFactor;
    height *= scalingFactor;

    boundingBox.scale(scalingFactor);
}

//-----------------------------------------------------------------------------

void Cylinder::setRadius(double r)
{
    // Implement this function
    radius = r;
    double d = this->getDiameter();
    boundingBox.setUpperRightVertex(d,d, height);

}
//-----------------------------------------------------------------------------
void Cylinder::setHeight(double h)
{
    // Implement this function
    height = h;
    double d = (radius * 2);
    boundingBox.setUpperRightVertex(d,d, height);
}

//-----------------------------------------------------------------------------
Polyhedron* Cylinder::clone() const
{
    return new Cylinder(*this);
}
```