```
/*
 * Shapes.h
 *
 *  Created on: Nov 9, 2013
 *      Author: Nathaniel Gallinger
 */

#ifndef SHAPES_H_
#define SHAPES_H_

namespace NathanielGallinger
{
  // parent class
  class Shapes
  {
  public:
    virtual ~Shapes();
    virtual void display() const = 0;
  };

  // inherit from shapes
  class TwoDimensionalShape : public Shapes
  {
  public:
    virtual ~TwoDimensionalShape();
    virtual double getArea() const = 0;
  };

  // inherit from TwoDimensionalShape
  class Circle : public TwoDimensionalShape
  {
  public:
    Circle(double radius);
    void display() const;
    double getArea() const;
  private:
    double radius;
  };

  // inherit from TwoDimensionalShape
  class Square : public TwoDimensionalShape
  {
  public:
    Square(double lengthOfSide);
    void display() const;
    double getArea() const;
  private:
    double lengthOfSide;
  };
```

```
  // inherit from shapes
  class ThreeDimensionalShape : public Shapes
  {
  public:
    virtual ~ThreeDimensionalShape();
    virtual double getSurfaceArea() const = 0;
    virtual double getVolume() const = 0;
  };

  // inherit from ThreeDimensionalShape
  class Sphere : public ThreeDimensionalShape
  {
  public:
    Sphere(double radius);
    void display() const;
    double getSurfaceArea() const;
    double getVolume() const;
  private:
    double radius;
  };

  // inherit from ThreeDimensionalShape
  class Cube : public ThreeDimensionalShape
  {
  public:
    Cube(double lengthOfSide);
    void display() const;
    double getSurfaceArea() const;
    double getVolume() const;
  private:
    double lengthOfSide;
  };
}

#endif /* SHAPES_H_ */


/*
 * Shapes.cpp
 *
 *  Created on: Nov 9, 2013
 *      Author: Nathaniel Gallinger
 */

#include "Shapes.h"
#include <cmath>
#include <iostream>
using std::cout;
using std::pow;
```

```
const double PI = 3.14159;

// Shapes destructor
NathanielGallinger::Shapes::~Shapes()
{
  // Empty Destructor
}

// TwoDimensionalShape destructor
NathanielGallinger::TwoDimensionalShape::~TwoDimensionalShape()
{
  // Empty Destructor
}

// TwoDimensionalShape destructor
NathanielGallinger::ThreeDimensionalShape::~ThreeDimensionalShape()
{
  // Empty Destructor
}

// Circle constructor
NathanielGallinger::Circle::Circle(double radius)
{
  this->radius = radius;
}

// Circle display function
void
NathanielGallinger::Circle::display()  const
{
  cout << "Circle with radius " << radius << " has area " <<
getArea();
}

// Circle get area
double
NathanielGallinger::Circle::getArea() const
{
  return PI * pow(radius, 2);
}

// Square constructor
NathanielGallinger::Square::Square(double lengthOfSide)
{
  this->lengthOfSide = lengthOfSide;
}

// Square display function
void
NathanielGallinger::Square::display()  const
```

```
{
  cout << "Square with length of side " << lengthOfSide << " has area
" << getArea();
}

// Square get area
double
NathanielGallinger::Square::getArea() const
{
  return pow(lengthOfSide, 2);
}

// Sphere constructor
NathanielGallinger::Sphere::Sphere(double radius)
{
  this->radius = radius;
}

// Sphere display fuction
void
NathanielGallinger::Sphere::display()  const
{
  cout << "Sphere with radius " << radius << " has surface area " <<
getSurfaceArea() << " and volume " << getVolume();
}

// Sphere get surface area
double
NathanielGallinger::Sphere::getSurfaceArea() const
{
  return 4 * PI * pow(radius, 2);
}

// Sphere get volume
double
NathanielGallinger::Sphere::getVolume() const
{
  return (4/3) * PI * pow(radius, 3);
}

// Cube constructor
NathanielGallinger::Cube::Cube(double lengthOfSide)
{
  this->lengthOfSide = lengthOfSide;
}

// Cube display function
void
NathanielGallinger::Cube::display()  const
{
```

```
  cout << "Cube with length of side " << lengthOfSide << " has surface
area " << getSurfaceArea() << " and volume " << getVolume();
}

// Cube get surface area
double
NathanielGallinger::Cube::getSurfaceArea() const
{
  return 6 * pow(lengthOfSide, 2);
}

// Cube get volume
double
NathanielGallinger::Cube::getVolume() const
{
  return pow(lengthOfSide, 3);
}


/*
 * hw5.cpp
 *
 *  Created on: Nov 9, 2013
 *      Author: Nathaniel Gallinger
 */

#include "Shapes.h"
using namespace NathanielGallinger;


int main()
{
  const char NUM_SHAPES = 4;
  enum shapes {
    CIRCLE = 0,
    SQUARE = 1,
    SPHERE = 2,
    CUBE   = 3
  };
  Shapes *pShapes[4];

  // create a shape of each type
  Circle ci(4);
  Square sq(10);
  Sphere sp(15);
  Cube cu(60);

  // add shapes to array
  pShapes[CIRCLE] = &ci;
  pShapes[SQUARE] = &sq;
```

```
  pShapes[SPHERE] = &sp;
  pShapes[CUBE]   = &cu;

  // loop through, call display on each shape, then delete
  for (int idx = 0; idx < NUM_SHAPES; idx++) {
    pShapes[idx]->display();
    delete pShapes[idx];
  }
}
```

```
Output:
Circle with radius 4 has area 50.26544
Square with length of side 10 has area 100
Sphere with radius 15 has surface area 2827.431 and volume 14137.155
Cube with length of side 60 has surface area 21600 and volume 216000
```