```
double getCost(Rectangle rect)
{
    return rect.length * rect.width * 18;   → Cost per Sqft of Carpet
}
```

private members of Rectangle class that can be accessed by the CostCalculator class, because it is friend of Rectangle class.

```
};// end class CostCalculator
// Driver of classes
int main()
{
    Rectangle floor;
                   w   len
    floor.setData(20, 3);
    CostCalculator calc;                    #1040   object
    cout << "The cost for Carpet is $" << calc.getCost(floor) << endl;
    return 0;
} //end main
```

Week 3, Sat: Polymorphism, Virtual functions, and Abstract class:

Poly → Many, morphism → forms

- pointer of base class type can point to an object of its derived class.

```
Class Shape
{  protected:
           double width, height;

   public:  Void Set_data(double a, double b)
            { width = a;
              height = b;
            }
}; // end Shape

Class Rectangle : public Shape // Rectangle inherites from Shape class
{  public:
           double area()
           { return width * height; }
}; // end Rectangle
// Driver for classes
int main()
{  Shape *Sptr; // sptr is a pointer of type Shape
Rectangle rect; // using default Constructor which is provided by C++
Sptr = &rect;   // Compiler
Sptr->Set_data (2.6, 3.8);
Cout << Sptr->area() << endl; // Error
 return 0;
} //end main
```

Because the base class pointer Cannot access additional member function of its derived class.

* To fix this error one way is type Casting: converts

```
Cout << static_cast< Rectangle *> (Sptr) -> area() << endl; ✓
```

- Virtual Functions and Polymorphism:

We use virtual functions to support polymorphism behavior in C++.

```
Virtual double area()
{ return 0; }
```

This goes before } of Shape class.

- A member of a class that can be redefined in its derived class is know as Virtual member.
- The advantage of having virtual function is that we are able to access member function "area" of the derived class.
- pure virtual function: It is also called "abstract function", and a Class with at least one pure virtual function is called an "abstract class". We cannot instantiate (Creat) an object of an abstract class.

```
Virtual double area() = 0; // pure virtual function
```

```
Class Shape          protected: double width, height;
{  public:
        Virtual void draw() = 0; // pure virtual function
}; // end Shape "abstract class"        virtual double area()
                                        { return 0; }
Class Circle: public Shape
{
   public:  void print()
            { Cout << "I am a Circle." << endl; }
};// end circle
class Rectangle : public Shape
{   public :   void draw()
               { cout <<"Drawing a rectangle."<<endl;}
                         double area()
                         { return width * height ;}
};// end Rectangle
// Driver for classes
int main()
{   Rectangle rect; // Good
     rect.draw(); // Good
     Circle C1;    // Error
      C1.print();// Error
      return 0;
  } //end main
            Shape *sptr;
            sptr = &rect;
            // sptr-> set_data (2.6, 3.8);
            cout << sptr->area() << endl; // Good
                                  Good
```

Because Circle is an abstract class, we didn't redefine (override) pure virtual function of its base class.

Because pointer of base class can access a member function of a child class which was virtual function in the parent class.

---

final class:

```
Class myClass final // This class cannot be a base class.
{    - - - //body of class       // We cannot inherite from this class
};
```

In C++, a base class virtual function that is declared "final" in its prototype:

e.g.   Virtual CalcName (parameters) final ; // Cannot be overriden in
                                              // any derived classes

- This guarntees that the base class's final member func. definition will be used by all base class objects and by all objects of the base class's direct and indirect derived classes.

- Static Data Member: It is one variable for all objects. It is created and initialized once. Initialization of a static data member is done outside of the class.

Example:
```
#include <iostream>
using ... std;
Class Circle
{  private:  double radius;

   public:  static int count;
   // Constructor with one agrument
   Circle ( double r)
   {  radius = r;
      count++; //We count # of objects
   }
   double getArea( )
   {  return 3.14 *radius *radius; }
 }; //end class
int Circle:: count= 0; // Initialization
 // Driver for Circle
 int main( )
 {
     Circle  C1 (2.7);
     Circle  C2 (4.5);
   // Circle  C3; // Error
     Cout << " Total objects : " << Circle :: Count << endl; //2
     return 0;
 }
```

- Static Member function: The static func. can access only the static data member of a class and Cannot call non-static funcs. of the class.

```
class Circle
{ private:  static int count;
            double radius;
  public:
        // Constructor
        Circle (double r)
        {  radius = r;
           count++;
        }
        double getArea()
        { return 3.14*radius*radius; }
        static int getCount()
        { return count; }
}; //end Circle
int Circle:: count=0; //initialization is outside of the class
// Driver for class Circle
int main ()
{   Circle C1 (2.4);
    Circle C2 (3.8);
    cout << "Total objects: " << Circle::getCount() <<endl; //2
    return 0;
 } //end main
```