```
sales = new double [numDays]; // Dynamic Memory allocation
or // double *sales = new double [numDays];
```

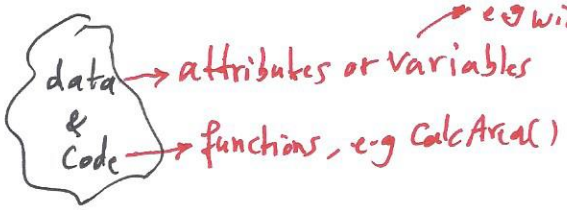
OOP: We have a class and object. Please look at the Word lecture in Canvas.
Class is a blueprint (template) for an object. An object is an instance of the class.

We tried three examples: 1- class Person 2- class BankAccount 3- class Rectangle

Week 2, Sat.:

OOP has the following features:

1- Encapsulation (Data and Code)



2- Inheritance: A class can inherite from another class. e.g. Student class can inherite from Person class.
Employee " " " " " " .

3- Polymorphism (Many forms): Many forms.

ADT (Abstract Data Type) is programmer's defined data type.

Class ≡ Struct ≡ ADT *{ int age; Person david;

Information Hiding: We declare data members a private members.

Access Specifiers: public, private, protected, abstract, or final.

Constructors: They are for initialization purpose. When we create objects we can initialize private members. We have default Constructor and Constructor with arguements

Friends: Friends are functions or classes declared with the "friend" keyword. "friend" is a non-member function but can access private and protected members of a class.

functions

Example:

```
Class Rectangle
{ ...
  friend double getCost(Rectangle); // friend func. prototype
```

```
};
double getCost ( Rectangle rect)
{  double Cost;
   Cost = rect.length * rect.width * 15;
   return Cost;
}
int main ( )
{
   Rectangle floor ( 20 , 4 ); // using Constructor
   cout<< "Expense: "<< getCost (floor) <<endl; // 1200
    // We are not using an object to access it.
   return 0;
} // end main
```

(w = 20, len = 4; #15 per sqft of Carpet)

Inheritance: A class can inherite from other class.

- Parent class ≡ base class ≡ Super class
- Child class ≡ inherited class ≡ sub class

- Inherited class (child class) can access private and public members of the base class (parent class). Of course we need to declare private members in parent class as protected members otherwise we cannot access them.

e.g.
```
class Rectangle : public Shape // Rectangle class inherites
{  . . .                       // from Shape class
   . . .
   . . .
}
```

Inheritance is specialization, this means inherited class has all attributes of parent class plus its own attributes (private members).
e.g. Student has all attributes of Person class plus its own attributes like gpa, number_of_units, and year_in_school.

In Inheritance we have "is a" relationship.
for example:

Student is a Person.
Employee is a Person.
Rose is a Flower
Tiger is an Animal

In Composition we have "has a" relationship.

Composition means a class can contain another class as its member.

e.g. Person has a Name. → Name is a class.

Multiple Inheritance:

Class A : public B
⋮ ⋮
Class B : public C
⋮ ⋮

- We did class activity: Rectangle and Box

- Friend Classes: One class member func. can access the private and protected member of other class.
We do this by declaring a class as friend of other class.

```
#include <iostream>
using - . . .

class CostCalculator; // friend class prototype
class Rectangle
{  . . .
   . . .
   . . .
   friend class CostCalculator; // friend of class Rectangle
   . . . .
}; //end Rectangle
class CostCalculator
{  public:
```

↘ ↘ ↘

```
double getCost(Rectangle rect)
{
    return rect.length * rectowidth * 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;
    floor.setData(20, 3);
    CostCalculator calc;
    cout << "The cost for Carpet is $" << calc.getCost(floor) << endl;
    return 0;
}//end main
```

floor: w = 20, len = 3

#1040 object