MA122 - Computer Programming and Applications

Indian Institute of Space Science and Technology

March 23, 2017

Lecture 23

1 Classes

example 1

```cpp
#include <iostream>

using namespace std;

class Box {
public:
  double length; // Length of a box
  double breadth; // Breadth of a box
  double height; // Height of a box
};
```

example

```cpp
int main( ) {
  Box Box1;        // Declare Box1 of type Box
  Box Box2;        // Declare Box2 of type Box
  double volume = 0.0;   // Store the volume of a box here

  // box 1 specification
  Box1.height = 5.0;
  Box1.length = 6.0;
  Box1.breadth = 7.0;

  // box 2 specification
  Box2.height = 10.0;
  Box2.length = 12.0;
  Box2.breadth = 13.0;
```

example

```

  // volume of box 1
  volume = Box1.height * Box1.length * Box1.breadth;
  cout << "Volume of Box1 : " << volume <<endl;

  // volume of box 2
  volume = Box2.height * Box2.length * Box2.breadth;
  cout << "Volume of Box2 : " << volume <<endl;

  return 0;
}
```

example 2

```
// classes example
#include <iostream>
using namespace std;

class Rectangle {
    int width, height;
public:
    void set_values (int,int);
    int area() {return width*height;}
};
```

example 2

```
void Rectangle::set_values (int x, int y) {
  width = x;
  height = y;
}

int main () {
  Rectangle rect;
  rect.set_values (3,4);
  cout << "area: " << rect.area();
  return 0;
}
```

example 3

```
#include <iostream>
class Test
{
private:
    int data1;
    float data2;

public:

    void insertIntegerData(int d)
    {
        data1 = d;
        std::cout << "Number: " << data1;
    }
```

example 3

```

    float insertFloatData()
    {
        cout << "\nEnter data: ";
        cin >> data2;
        return data2;
    }
};
```

example 3

```

int main()
{
    Test o1, o2;
    float secondDataOfObject2;

    o1.insertIntegerData(12);
    secondDataOfObject2 = o2.insertFloatData();

    cout << "You entered " << secondDataOfObject2;
    return 0;
}
```

example 4

```
#include <iostream>
#include <string>
class Stock // class declaration
  {
  private:
    std::string company;
    long shares;
    double share_val;
    double total_val;
    void set_tot() { total_val = shares * share_val; }
  public:
    void acquire(const std::string & co, long n,
        double pr);
    void buy(long num, double price);
    void sell(long num, double price);
    void update(double price);
    void show();
  };    // note semicolon at the end
```

example 4

```
void Stock::acquire(const std::string & co, long n,
     double pr)
{
  company = co;
  if (n < 0)
  {
    std::cout << "Number of shares cant be negative; "
    << company << " shares set to 0.\n";
    shares = 0;
  }
  else
    shares = n;
  share_val = pr;
  set_tot();
}
```

example 4

```cpp
void Stock::buy(long num, double price)
{
  if (num < 0)
  {
    std::cout << "Number of shares purchased cant be negative. "
    << "Transaction is aborted.\n";
  }
  else
  {
    shares += num;
    share_val = price;
    set_tot();
  }
}
```

example 4

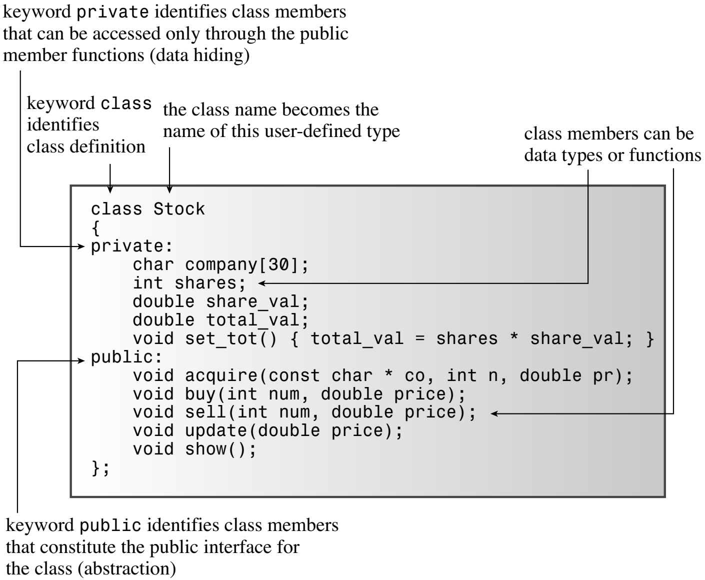
```
void Stock::sell(long num, double price)
{
  using std::cout;
  if (num < 0)
  {
    cout << "Number of shares sold cant be negative. "
    << "Transaction is aborted.\n";
  }
  else if (num > shares)
  {
    cout << "You cant sell more than you have! "
    << "Transaction is aborted.\n";
  }
  else
  {
    shares -= num;
    share_val = price;
    set_tot();  } }
```

example 4

```cpp
void Stock::update(double price)
{
  share_val = price;
  set_tot();
}
void Stock::show()
{
  std::cout << "Company: " << company <<
  "  Shares: " << shares << std::endl<<
  "  Share Price: $" << share_val
  << "  Total Worth: $" << total_val << std::endl;
}
```

example 4

```
int main()
{
  Stock fluffy_the_cat;
  fluffy_the_cat.acquire("NanoSmart", 20, 12.50);
  fluffy_the_cat.show();
  fluffy_the_cat.buy(15, 18.125);
  fluffy_the_cat.show();
  fluffy_the_cat.sell(400, 20.00);
  fluffy_the_cat.show();
  fluffy_the_cat.buy(300000,40.125);
  fluffy_the_cat.show();
  fluffy_the_cat.sell(300000,0.125);
  fluffy_the_cat.show();
  return 0;
}
```



keyword `private` identifies class members that can be accessed only through the public member functions (data hiding)

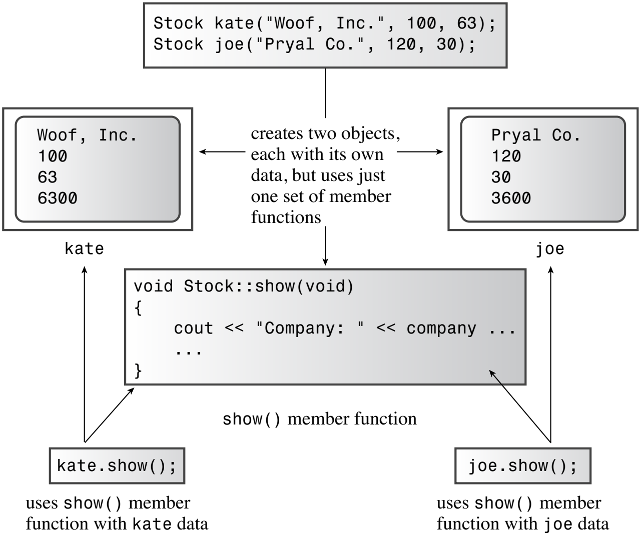
keyword `class` identifies class definition

the class name becomes the name of this user-defined type

class members can be data types or functions

```
class Stock
{
private:
    char company[30];
    int shares;
    double share_val;
    double total_val;
    void set_tot() { total_val = shares * share_val; }
public:
    void acquire(const char * co, int n, double pr);
    void buy(int num, double price);
    void sell(int num, double price);
    void update(double price);
    void show();
};
```

keyword `public` identifies class members that constitute the public interface for the class (abstraction)



```
Stock kate("Woof, Inc.", 100, 63);
Stock joe("Pryal Co.", 120, 30);
```

creates two objects, each with its own data, but uses just one set of member functions

```
Woof, Inc.
100
63
6300
```

kate

```
Pryal Co.
120
30
3600
```

joe

```
void Stock::show(void)
{
    cout << "Company: " << company ...
    ...
}
```

show() member function

```
kate.show();
```

uses show() member function with kate data

```
joe.show();
```

uses show() member function with joe data