Lecture notes: week 7: Working with Classes.

A Class contains data AND code. This is called "encapsulation".

Objects are compiled from Class definitions, and "instantiated" at runtime.

The structure of an object:

class classname

{

// all members are private by default (private, protected, public, friend)

private:

int private\_int1 ;

string privateString ;

public:

// "getter" method

int get\_private\_int(void)

{

return this->privateint1 ;

} // I'm a "getter"

// "setter" method

void set\_private\_int(int s)

{

this->private\_int1 = s) ;

}

// constructor (no type, same name as class)

classname()

{

// setup and housekeeping

}

classname(int s)

{

//another version of the constructor

this->private\_int1 = s ;

}

} ; // note semicolon

This is an oversimplified class definition, but it shows the bones.

Classes run a "constructor" function when they are created. If you

don't write one, the compiler will, and it won't do anything.

You can have multiple constructors with different argument lists.

This is called "polymorphism". In fact, you can have multiple versions

of any function, so long as it can be distinguished by its arguments in

type or number or both.

OVERLOADING:

Overloading means, several methods (functions) have the same name, but

different argument lists. Simple example:

int sum (int a, int b) ;

int sum (int a, int b, int c) ;

int sum (int a, int b, int c, int d) ;

int sum (int a, ...) ;

double sum (double a, double b) ;

double sum (double a, double b, double c) ;

These could all exist in the same program, and the compiler would compile in

the ones that were indicated by the way the program used them.

/\*-----------------------------------------------------------------

\* cardclass01.cpp - show class objects as a deck of cards

Mon Mar 10 15:33:44 PDT 2014

-----------------------------------------------------------------\*/

#include <cstdio>

#include <cctype>

#include <iostream>

#include <iomanip>

#include <cstdlib>

using namespace std ;

class carddeck

{

public:

string deck[52] ;

string suit[4];

string face[13];

string deal\_one(void)

{

unsigned int rs = (rand() % 4) ;

unsigned int rf = (rand() % 13) ;

// cout << "rf: " << rf << " rs: " << rs << endl ;

string t = face[rf] + " of " + suit[rs] ;

return t ;

} // deal\_one

carddeck() // constructor

{

srand(time(NULL)) ;

suit[0] = "Spades" ;

suit[1] = "Hearts" ;

suit[2] = "Diamonds" ;

suit[3] = "Clubs" ;

face[0] = "Ace" ;

face[1] = "Two" ;

face[2] = "Three" ;

face[3] = "Four" ;

face[4] = "Five" ;

face[5] = "Six" ;

face[6] = "Seven" ;

face[7] = "Eight" ;

face[8] = "Nine" ;

face[9] = "Ten" ;

face[10] = "Jack" ;

face[11] = "Queen" ;

face[12] = "King" ;

} // constructor

void print\_deck()

{

string t ;

int s , f ;

for (s = 0 ; s < 4 ; ++s)

{

for (f = 0 ; f < 13 ; ++f)

{

t = face[f] + " of " + suit[s] ;

cout << t << endl ;

} // for f

} // for s

} // print\_deck

} ; // class

int main (int argc, char \*argv[], char \*\*env)

{

carddeck c ;

// cout << "Card dealt is: " << c.deal\_one() << endl ;;

for (int i = 0 ; i < 7 ; ++i)

cout << c.deal\_one() << endl ;

// c.print\_deck() ;

} // main ends