Notes From Class

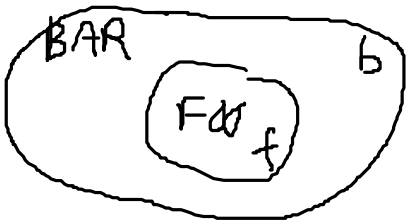
Thursday, March 10, 2016 4:00 PM

Project 7 Submission URL Is Available Now

- Due: Friday At 9 PM

Student Questions:

- No You Don't Need To Comment Or Test Code That That Skeleton Provided. Just Document And Test Player And Game
- `char * myString = "Hello";` /// is myString an object?
  `string myOtherString( "Hello" );` /// is myOtherString an object?
- Is `string` in the namespace `std`? Yes or No
  Can you get string to work even if you don't say #include <string>
- Foo Is A Subclass Of Bar

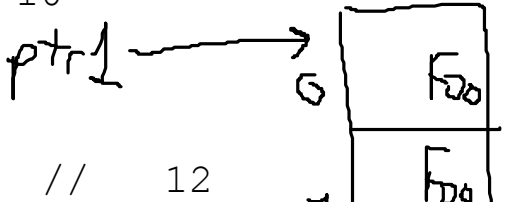



```
Foo f;
Bar b;
b = f;    /////  legal or not???
f = b;    /////  legal or not???
```

- Bar * ptrBar = &b;
- ptrBar = &f;

- Is This Legal Runnable Code? Let's Correct The Mistakes, If There Are Any
  How Many Times Is The Constructor Of Foo Called?
  Where Would We Need To Properly Call `delete` and `delete[ ]`

```
int main( )
{
   Foo array2[ 5 ];      /// Foo( ):  5
   Foo * ptr;
   ptr = new Foo( );     ////         6

   Foo * ptr1;
   ptr1 = new Foo( );    ////         7

   Foo x;                ////         8
   x = Foo( );           ////         9
   Foo f;                ////        10
   delete ptr;
   ptr = & f;
   delete ptr1;
   /* Foo * */ ptr1 = new Foo[ 2 ];   //   12
```

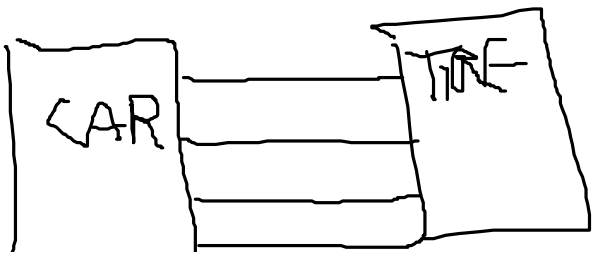


```
        delete ptr1;
        /* Foo * */ ptr1 = new Foo[ 2 ];   //   12
        // an error:    delete ptr;
        delete [] ptr1;

        Foo * newOne = new Foo( 1, 2, 3, 4 );
        Foo * newArray = new Foo[ 56 ];
    }
```

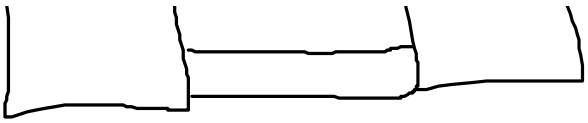
Review The Online Sample Final Exam Problems
Thanks For A Wonderful Class!
See You Saturday @ 11:30 Right Here…

```
class Tire
{
public:
    Tire( );
     void roll( int amount );
private:
     int myDistance;
};

Tire::Tire( ) : myDistance( 0 )
{ // empty... }

void Tire::roll( int amount )
{  myDistance += amount;  }
```

Car c; //// have 4 Tires



Car c; //// have 4 Tires

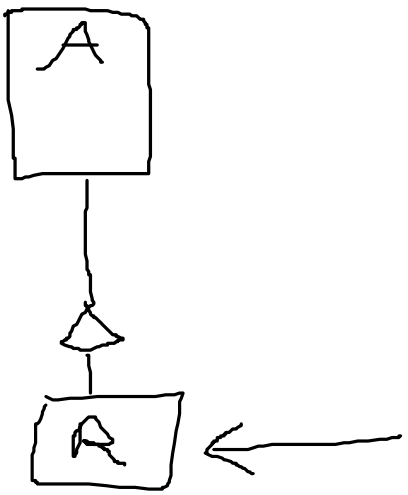


"AGGREGATION"

```
#ifndef CAR_H
#define CAR_H
#include "Tire.h"
Class Car
{
Public:
  Car( );
  void drive(  int amount );
Private:
  Tire myFrontLeft, myFrontRight, myRearLeft, myRearRight;
};
#endif

Car::Car( )  { }

Void Car::drive( int amount )
{
// four tires…. Roll them each amount
  myFrontLeft.roll( amount );
  myFrontRight.roll( amount );
  myRearLeft.roll( amount );
  myRearRight.roll( amount );
}
```

```
#include <iostream>
class Athlete
{
public:
    Athlete( std::string name, std::string sport);
```

```
    virtual void play(  );
private:
    std::string mName, mSport;
};


Athlete::Athlete( std::string name, std::string
sport ) : mName( name ), mSport( sport )
{ // empty... }


void Athlete::play( )
{  std::cout << mName << " is playing " << mSport
<< std::endl;  }
```




```
#include "Athlete.h"

Class Rower : public Athlete
{
Public:
  Rower( std::string name );
 virtual void play(  );

Private:
/// bad…
/// private:
////   std:string mSport, mName;
};


Rower::Rower(  std::string name  ) : Athlete( name  , "Rowing"  )
{
// empty
```

```
 }
```

```
Void Rower::play( )   {   cout << "Ready, All, Row!" << endl;  }
```

```
struct Candy
{
  std::string mMaker;
  std::string mName;
  double mCost;
};
```

```
Candy c = { "Mar's", "M&M's", 1.59 };
cout << c.mMaker << c.mName << c.mCost << endl;
```

```
Candy * ptrCandy = &c;
cout << ptrCandy->mMaker << ptrCandy -> mName << ptrCandy->mCost << endl;
/// length of the Candy's maker….
cout << ptrCandy->mMaker.length( );
```

Everything -9 or less and 4 and higher….

```
If (value <= -9 || value >= 4 ) {   …..   }   /// positively after the exact range    BLUE
```

```
If (!( value > -9 && value < 4 ))  { …. }   /// negatively    NOT in the range     NOT BLUE
```

-8 between 2 and -2 11 or more….

If (value == -8 || (value >= -2 && value <= 2) || value >= 11) { … } /// green range

Std::string you("You"), me("me");
If (you < me) { …. }
LEXICOGRAPHICALLY
Ascii

A = 65
B = 66
C = 67…

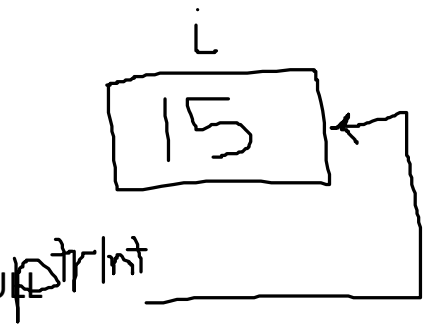
a = 97

10085 ---->>>>> 100 85
Guaranteed two digit percent 7% 10107
Guaranteed no one earned 100%

Guaranteed 100 or more

packedField % 100--------> 85 % only works with int on both sides…
Double d = 10085; d%100 /// won't build…

packedField / 100 ----> how many times does 100 go into 10085 -----> 100

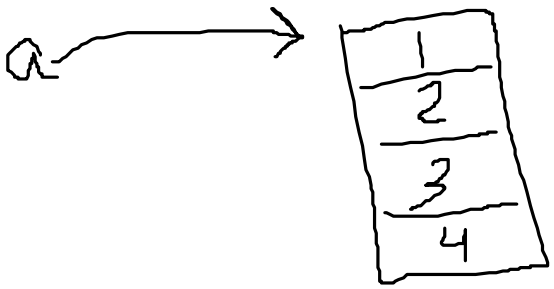


Int I = 15;
Int * ptrInt = &I;
Int * ptrOtherInt = nullptr ; // NULL

Int &

Void foo(int & referenceToAnInt); /////// pass-by-reference
/// that made an "alias" for the caller's version….
/// if I change this referenceToAnInt am changing the caller's universe…..
/// the difference between int & and int * ONLY int * can be nullptr

foo(I);



Int array[4] = { 1, 2, 3, 4 };
//// truly int * const array
Int * ptrVariable = nullptr;
ptrVariable = array; /// does…..
array = ptrVariable; /// doesn't work

const int * something; ///// immutable

Int const * something; /// two statements are identical…

```
const int Bar::c( const Foo & f ) const
{
  cout << f.getValue( );///  getValue          int getValue( ) const;
  f.setValue( v );      ///// not ever work…..
}
```

Just like public and private into parts….
About const
Into the read-only and the non-read-only portion….