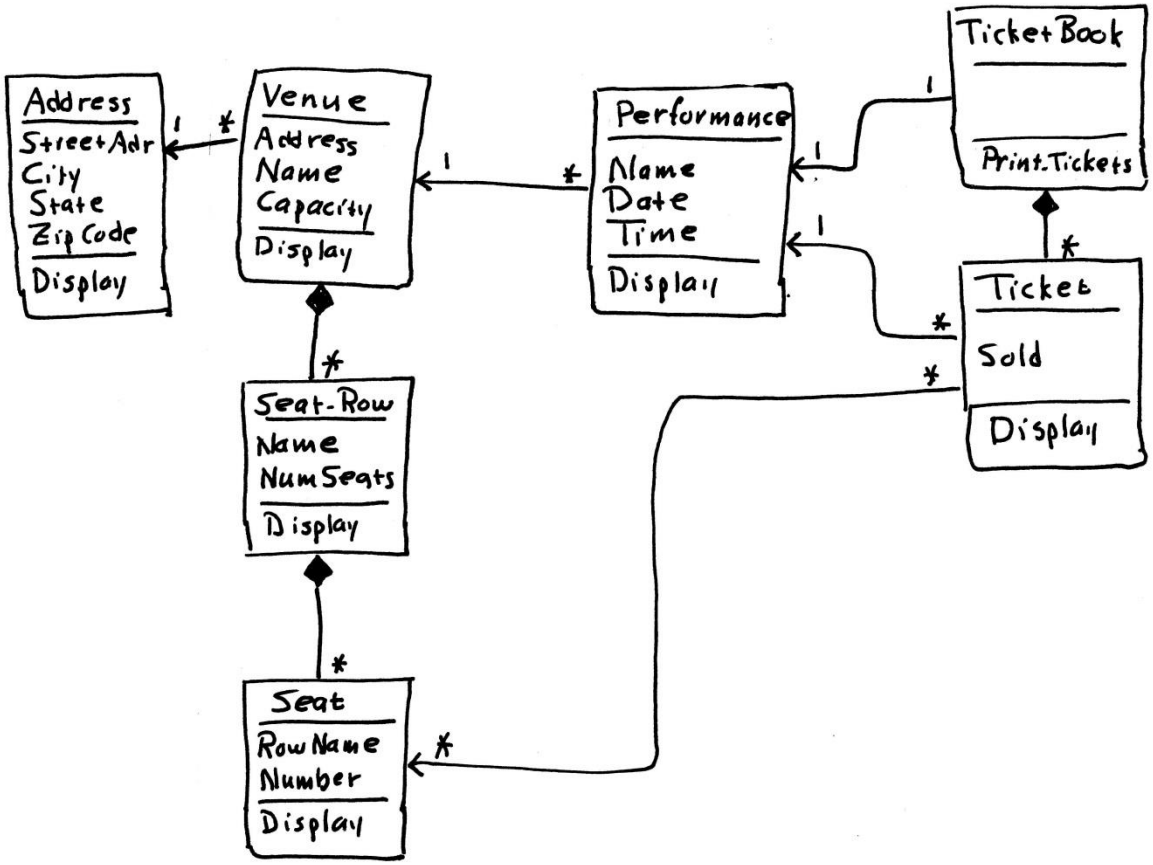
Implementing Ticket Printer

Class Diagram



Dependencies

- A class that *contains* objects of another class, or *has a reference* to another class, *depends on* that class.
  - Example: Class Venue *depends on* class Address and class Seat_Row.
- Before we can implement a class, we have to implement the classes on which it depends.
  - Those classes may also depend on other classes, forming *dependency chains*.

Implementing the Design

- Keep the class diagram in front of you.
  - This is a blueprint for the structure you are building.
- Start with a stub
  - Just the main.cpp
  - Hello world
- Add classes one at a time.
  - Get each one to compile and work before adding another.
  - Start at the ends of dependency chains and work up the chains.

Classes

- Design your classes as reusable components
- No knowledge of the specific venue and performance described for this program.
  - Pass that information in from main.
  - Future programs might get that information from user input or a file.

Create a New Project

- Visual C++
  - Win32
    - Win32 Console Application
      - Empty project

Add main.cpp

- Project > Add New Item

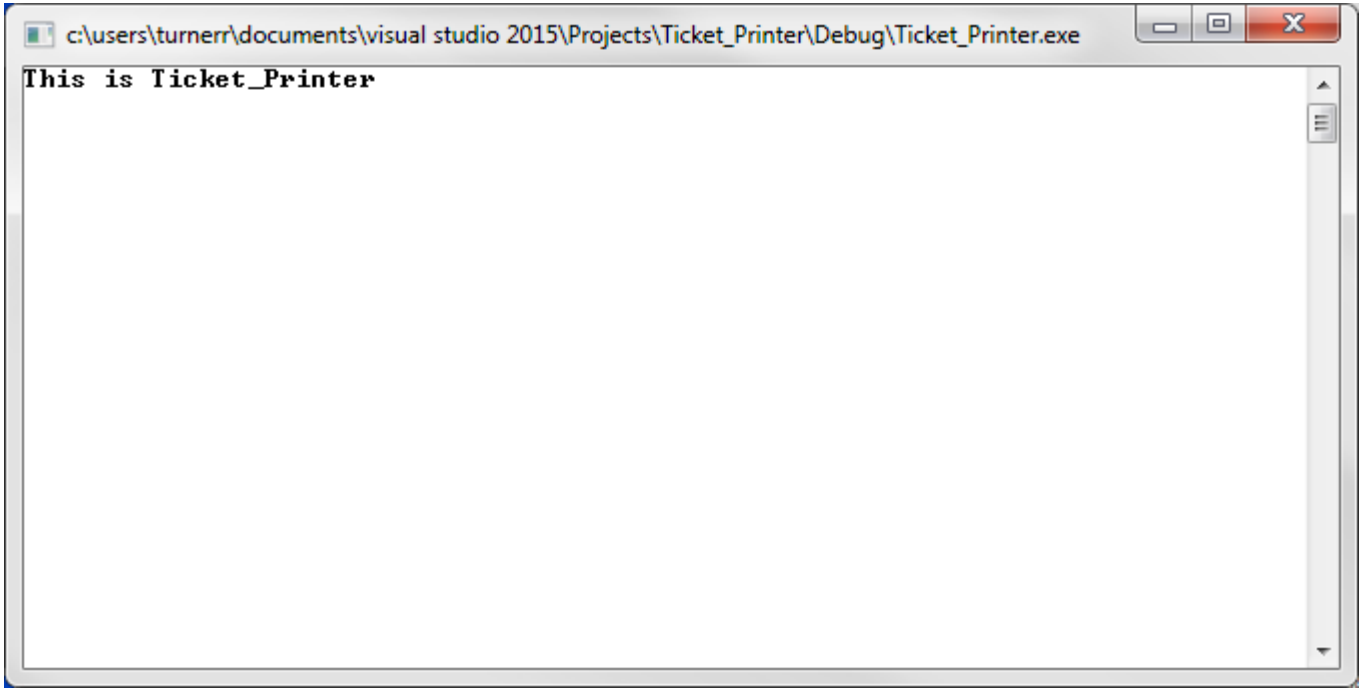
- Visual C++
  - C++ File
    - Name: main.cpp

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    cout << "This is Ticket_Printer\n\n\n";

    cin.get();
    return 0;
}
```

Build and run

Ticket_Printer in Action



From here on, ALWAYS HAVE A WORKING PROGRAM

What Next?

- Pick a class that has no dependencies.
- How about class Address?
- Project > Add Class
- Start with the class declaration
  - Address.h

Address.h

```
#pragma once
#include <string>

using namespace std;

class Address
{
private:
    string street_address;
    string city;
    string state;
    int zip_code;

public:
    Address(string Street_Address,
        string City,
        string State,
        int Zip_Code);

    void Display() const;
};
```

Implement the Class

Address.cpp

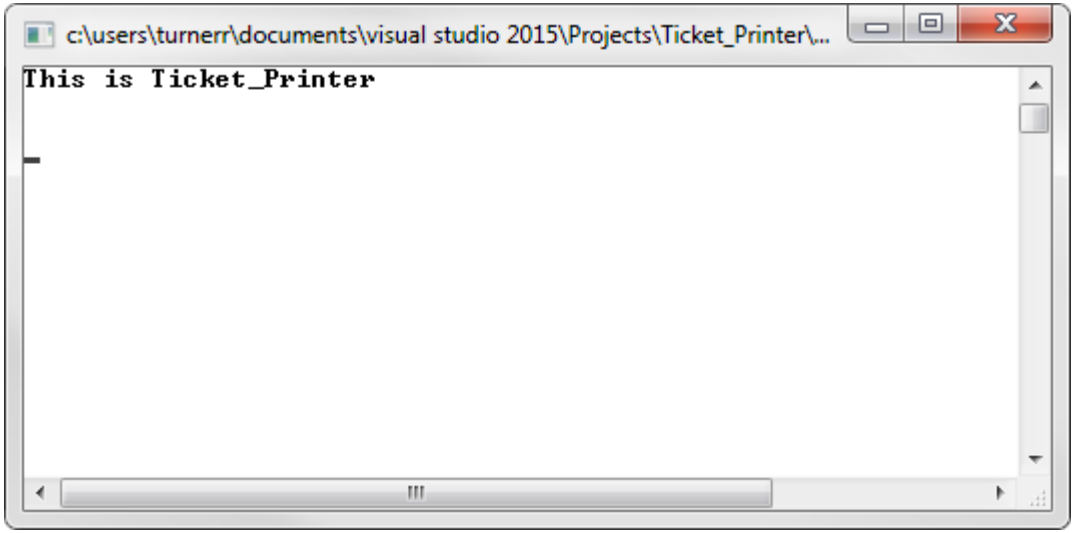
```
#include <iostream>
#include "Address.h"

Address::Address(string Street_Address,
                 string City,
                 string State,
                 int Zip_Code) :
        street_address(Street_Address),
        city(City),
        state(State),
        zip_code(Zip_Code)
{}
```

Implement the Class

```
void Address::Display() const
{
    cout.fill('0');
    cout << street_address << endl;
    cout << city << ", " << state << " ";
    cout.width(5);
    cout << zip_code << endl;
}
```

Build and run

Program Running



It's still at the "Hello, World" level, because we have not added any code to main.cpp

Add Test Driver for the Class

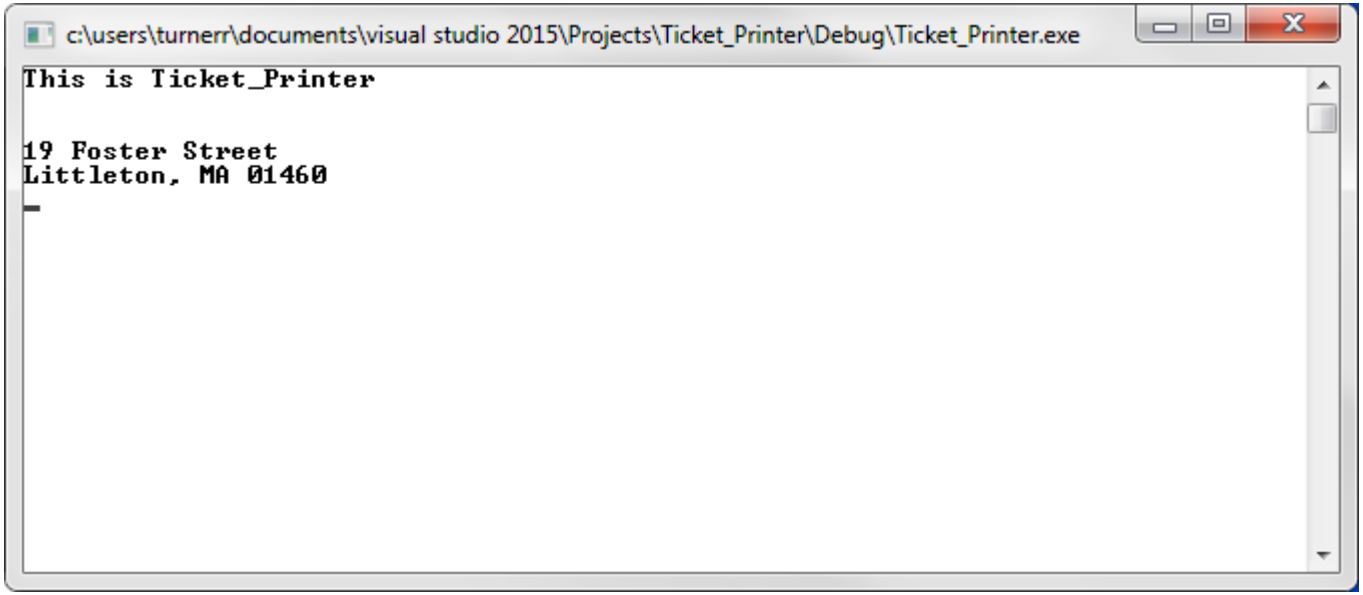
```
#include <iostream>
#include "Address.h"

using namespace std;

// Create an Address object with the address of The Little Theater
Address* Create_Address()
{
    Address* address = new Address("19 Foster Street",
        "Littleton",
        "MA",
        1460);
    return address;
}
```

Add Test Driver for the Class

```
int main()
{
    cout << "This is Ticket_Printer\n\n\n";

    Address* adr = Create_Address();
    adr->Display();
    cout << endl;

    cin.get();          // Hold the window open
    return 0;
}
```

Build and run

Program Running



Often you will find errors when you compile or when you run a test.

Keep working on that class until it compiles and runs correctly.

Add Another Class

- The Seat class has no dependencies.
  - Only remaining class with no dependencies.
  - A good choice to add next.

- Add class Seat to the project

Seat.h

```
#pragma once
#include <string>
using namespace std;

class Seat
{
private:
    string seat_row_name;
    int seat_number;

public:
    Seat(string Row_Name,
         int Seat_Number);

    void Display() const;
};
```

Seat.cpp

```
#include <iostream>
#include "Seat.h"

using namespace std;

Seat::Seat(string Row_Name,
           int Seat_Number) :
        seat_row_name(Row_Name),
        seat_number(Seat_Number)
{}

void Seat::Display() const
{
    cout << "Row " << seat_row_name << " Seat " << seat_number << endl;
}
```

Build and run

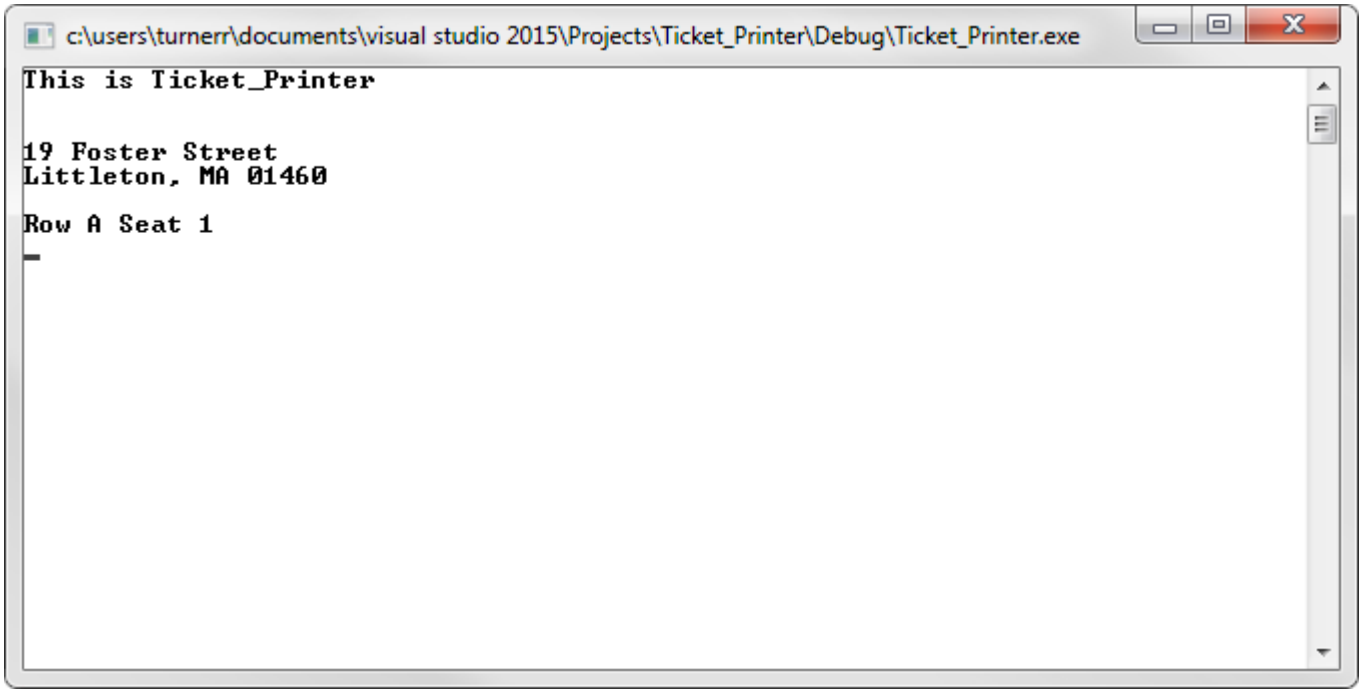
Add Test Code for Class Seat

main.cpp

```
#include "Seat.h"

...

    adr->Display();
    cout << endl;

    Seat* seat = new Seat("A", 1);
    seat->Display();
```

Build and run

Program Running



Test successful!

Add Another Class

- Work up a dependency chain
- Now that class Seat exists we can add class Seat_Row.
- Add class Seat_Row to the project.

Seat_Row.h

```
#pragma once
#include "Seat.h"

class Seat_Row
{
public:
    static const int MAX_SEATS_PER_ROW = 1000;

private:
    string row_name;
    const Seat* seats[MAX_SEATS_PER_ROW];
    int  number_of_seats;

public:
    Seat_Row(const string& Row_Name);
    void Add_Seat(const Seat* new_seat);
    void Display() const;
};
```

Seat_Row.cpp

```
#include <iostream>
#include <string>
#include <cassert>
#include "Seat_Row.h"
using namespace std;

Seat_Row::Seat_Row(const string& Row_Name) :
    row_name(Row_Name),
    number_of_seats(0)
{}

void Seat_Row::Add_Seat(const Seat* new_seat)
{
    assert(number_of_seats < MAX_SEATS_PER_ROW);
    seats[number_of_seats++] = new_seat;
}
```

Seat_Row.cpp

```
void Seat_Row::Display() const
{
    cout << "Row " << row_name << " has "
        << number_of_seats << " seats" << endl;
    for (int i = 0; i < number_of_seats; ++i)
    {
        cout << "\t";          // Indent the Seat description
        seats[i]->Display();
    }
}
```

Build and run

Add Test Code for Seat Row

main.cpp

```
#include "Seat_Row.h"

...
```

- In main()

```
    cout << endl;
    Seat_Row* row = new Seat_Row("Test");
    row->Display();

    cin.get();
    return 0;
}
```

Program Running

c:\users\turnerr\documents\visual studio 2015\Projects\Ticket_Printer\Debug\Ticket_Printer.exe

```
This is Ticket_Printer


19 Foster Street
Littleton, MA 01460

Row A Seat 1


Row Test has 0 seats
```

Add some seats to row Test.

In main.cpp

```
// Create a Seat_Row with the specified name and
// specified number of seats,
Seat_Row* Create_Seat_Row(const string seat_row_name,
                          int number_of_seats)
{
    Seat_Row* row = new Seat_Row(seat_row_name);
    for (int i = 1; i <= number_of_seats; ++i)
    {
        Seat* new_seat = new Seat(seat_row_name, i);
        row->Add_Seat(new_seat);
    }
    return row;
}
```

In main.cpp

```
    //Seat_Row* row = new Seat_Row("Test");
    Seat_Row* row = Create_Seat_Row("Test", 4);
    row->Display();
    cout << endl;

    cin.get();
    return 0;
}
```

Build and run

Program Running

c:\users\turnerr\documents\visual studio 2015\Projects\Ticket_Printer\Debug\Ticket_Printer.exe

```
This is Ticket_Printer


19 Foster Street
Littleton, MA 01460

Row A Seat 1


Row Test has 4 seats
        Row Test Seat 1
        Row Test Seat 2
        Row Test Seat 3
        Row Test Seat 4
```

Add Class Venue

- We now have the classes that class Venue depends on.
- Add class Venue to the project.

Venue.h

```
#pragma once
#include <string>
#include "Address.h"
#include "Seat.h"
#include "Seat_Row.h"

class Venue
{
public:
    static const int MAX_SEAT_ROWS = 1000;

private:
    string name;
    const Address* address;
    const Seat_Row* seat_rows[MAX_SEAT_ROWS];
    int number_of_seat_rows;
```

Venue.h

```
public:
    Venue(const string& name_,
          const Address& address_);

    void Add_Seat_Row(const Seat_Row* seat_row);

    int Capacity() const;     // Number of seats

    void Display() const;
};
```

Venue.cpp

```
#include <string.h>
#include <iostream>
#include <cassert>
#include "Venue.h"
#include "Seat_Row.h"
using namespace std;

Venue::Venue(const string& name_,
             const Address& address_) :
    name(name_), address(&address_), number_of_seat_rows(0)
{}

void Venue::Add_Seat_Row(const Seat_Row* seat_row)
{
    assert(number_of_seat_rows < MAX_SEAT_ROWS - 1);
    seat_rows[number_of_seat_rows++] = seat_row;
}
```

Venue.cpp

```
void Venue::Display() const
{
    cout << name << endl;
    address->Display();
    for (int i = 0; i < number_of_seat_rows; ++i)
    {
        const Seat_Row* row = seat_rows[i];
        row->Display();
    }
}
```

Build and run

Program Running

c:\users\turnerr\documents\visual studio 2015\Projects\Ticket_Printer\Debug\Ticket_Printer.exe

```
This is Ticket_Printer


19 Foster Street
Littleton, MA 01460

Row A Seat 1


Row Test has 4 seats
        Row Test Seat 1
        Row Test Seat 2
        Row Test Seat 3
        Row Test Seat 4
```

No output for Venue yet.

Add Test Code for Class Venue

main.cpp

```
#include "Venue.h"
...
// Create a Venue object corresponding to The Little Theater.
Venue* Create_Venue()
{
    Address* adr = Create_Address();
    Venue* venue = new Venue("The Little Theater", *adr);

    Seat_Row* sr = Create_Seat_Row("A", 4);
    venue->Add_Seat_Row(sr);

    sr = Create_Seat_Row("B", 4);
    venue->Add_Seat_Row(sr);

    sr = Create_Seat_Row("C", 4);
    venue->Add_Seat_Row(sr);

    return venue;
}
```

Add Test Code for Class Venue

In main()

```
    Venue* venue = Create_Venue();
    venue->Display();

    cin.get();    // Hold the window open
    return 0;
}
```

Build and run

Program Running

c:\users\turnerr\documents\visual studio 2015\Projects\Ticket_Printer\Debug\Ticket_Printer.exe

```
This is Ticket_Printer


19 Foster Street
Littleton, MA 01460

Row A Seat 1


Row Test has 4 seats
        Row Test Seat 1
        Row Test Seat 2
        Row Test Seat 3
        Row Test Seat 4

The Little Theater
19 Foster Street
Littleton, MA 01460

Row A has 4 seats
        Row A Seat 1
        Row A Seat 2
        Row A Seat 3
        Row A Seat 4

Row B has 4 seats
        Row B Seat 1
        Row B Seat 2
        Row B Seat 3
        Row B Seat 4

Row C has 4 seats
        Row C Seat 1
        Row C Seat 2
        Row C Seat 3
        Row C Seat 4
```

Summary

- Work in tiny steps
- Add one class at a time
  - Working up dependency chains.
    - Before adding a class, add classes on which that class depends.
  - Add test code to main function.
  - Build and test.
  - Be sure the program compiles and works before adding more code.