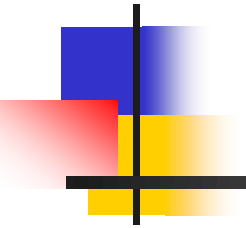
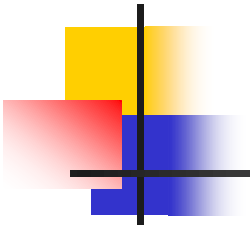


Project 1: Tickets



Class Ticket

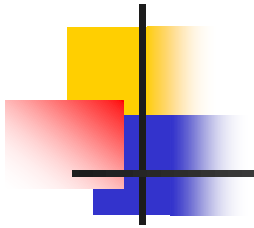
- Write a definition for class Ticket.
- A Ticket object corresponds to a physical ticket for a stage show.
- Attributes of a Ticket are:
  - Name of Show
  - Name of Venue
  - Date (3 integers, Month, Day, Year)
  - Time (2 integers, hour (0..23), minute (0..59)
  - Seat
    - Row Name
    - Seat Number (A positive integer)

Class Ticket

- Define structs to represent date/time and seat.
  - Put the struct definitions in Ticket.h prior to the class definition.
- Let the constructor take four parameters:
  - Show_Name
  - Venue_Name
  - When (Date and Time)
  - Seat (Row Name and Seat Number)

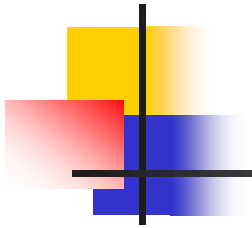
A Changeable Attribute

- Let each ticket have a Boolean attribute called "sold"
  - Initially false.
- Member function Sell() sets sold to true.
- Note: bool is a *type* in C++.



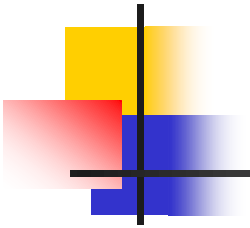
Display()

- Public member function Display() outputs all attribute values to the console.
  - See sample run for format.
  - Make your output match the sample run.



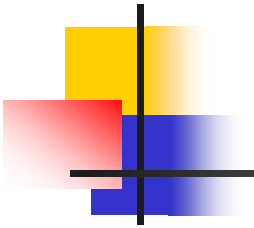
Accessor Methods

- Provide an accessor method for each attribute:
  - Venue_Name
  - Show_Name
  - When
  - Seat
  - Sold



Main Program

- Write a program to test the class definition.
- The main program:
  - Gets attributes for a ticket from the user.
  - Dynamically creates a Ticket object with the specified attributes.
  - Calls the ticket's Display() method.
  - Gets the attributes of the Ticket object with the accessor methods and shows them in the same format as the Display method.
  - Sells the ticket.
  - Calls the ticket's Display() method again.
  - Deletes the ticket.

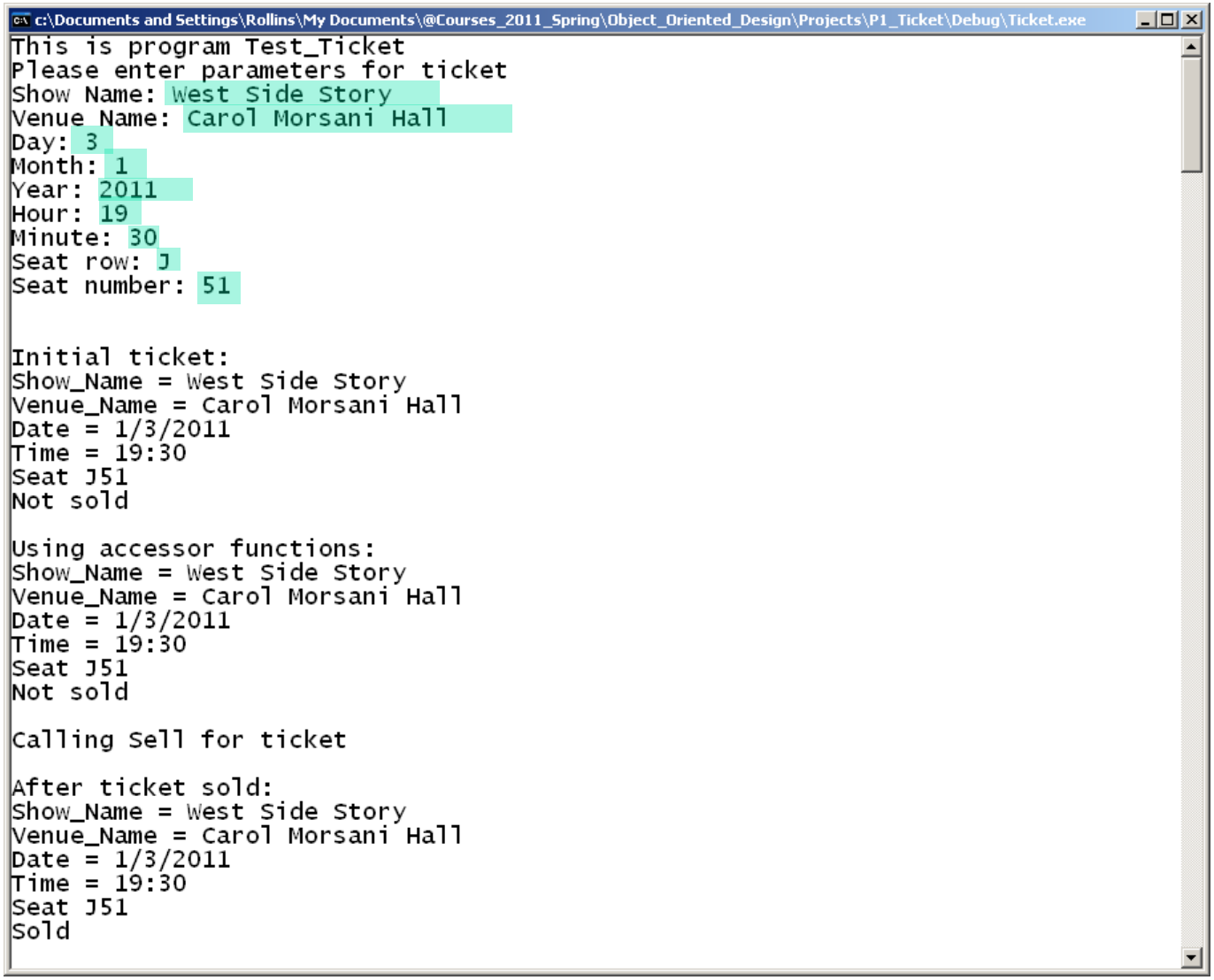


Main Program

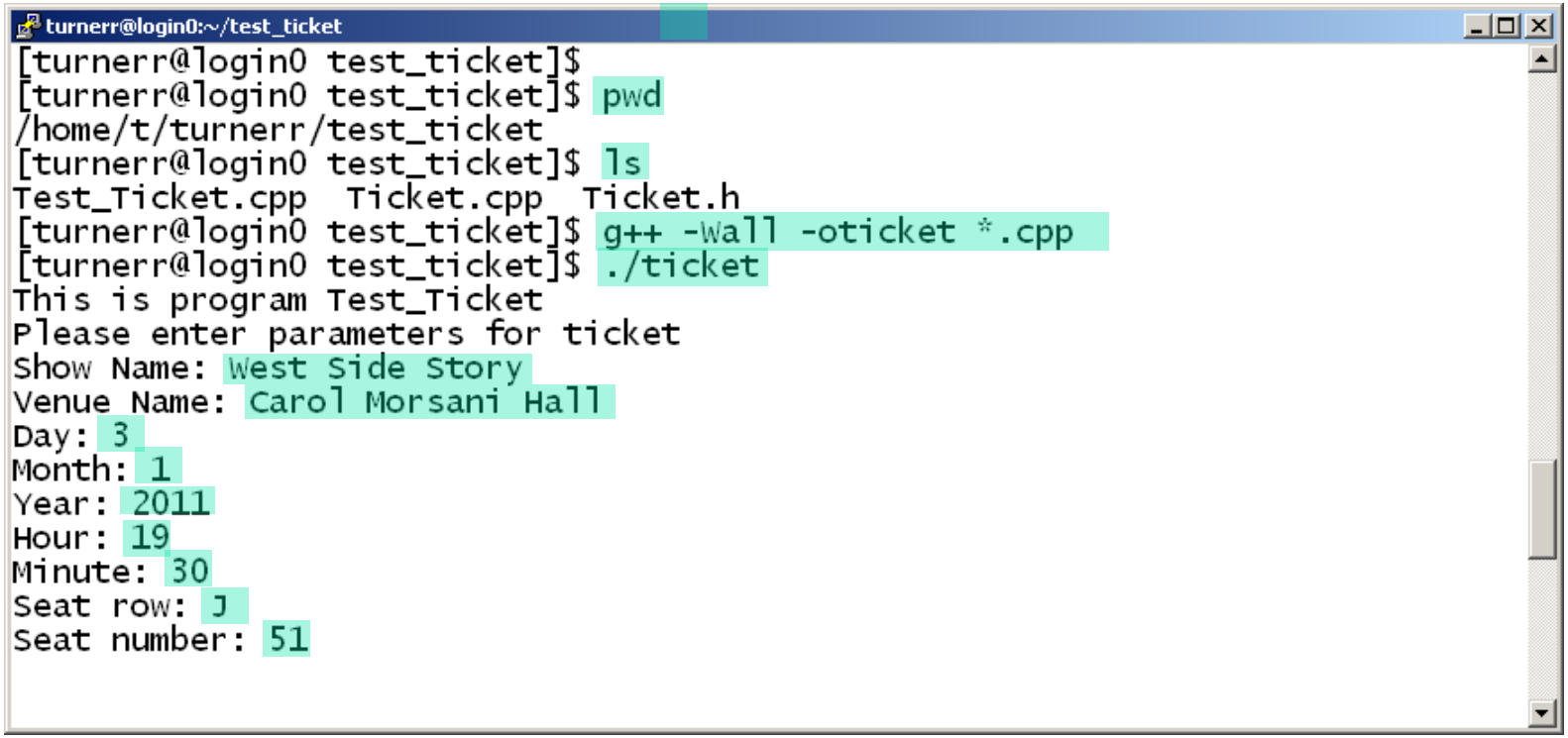
- Exit after creating and displaying one Ticket.

- Input validity checking is not required.
  - Assume user inputs valid values for all parameters.
  - OK to fail on invalid input.

Sample Run -- Windows

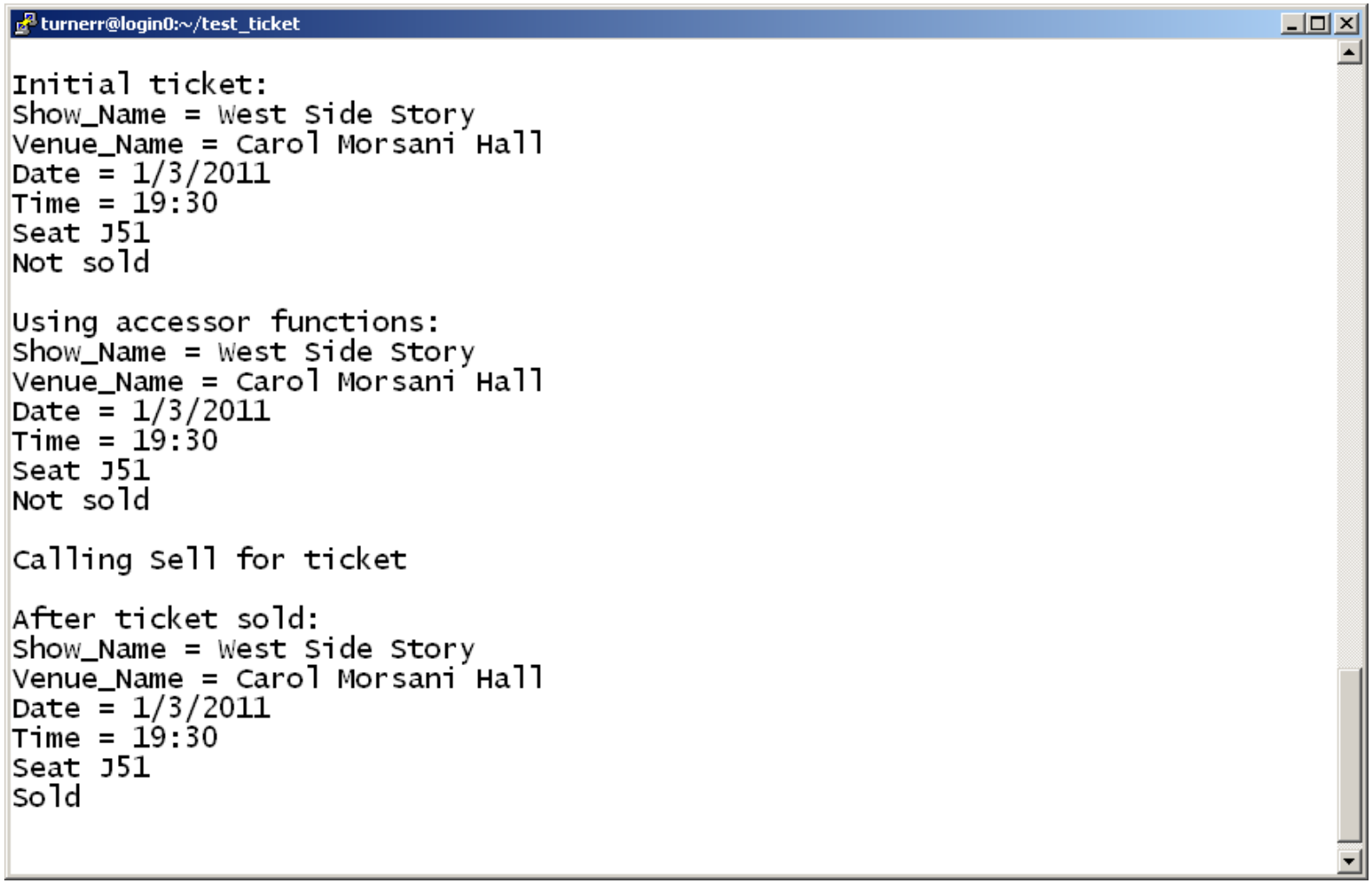


Sample Run -- Linux



Continued on next slide

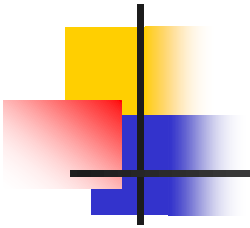
Sample Run -- Linux (continued)



Implementation Tips

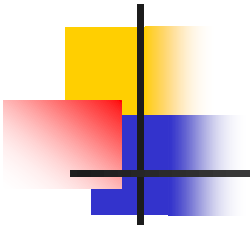
- Read about I/O manipulators.
  - Textbook page 1091
  - Header file iomanip

- Specific manipulators:
  - setfill( )
  - setw( )



Development Environment

- You may develop your program on any system you like.
  - But you should test the finished program on Circe.
- The same source file should compile and run on *either Windows or Linux*.



Ground Rules

- You may work with one other person.
  - OK to work alone if you prefer.
- If you do work as a pair
  - Work together!
  - Both members are expected to contribute.
  - Submit a single program.
  - Both members should understand work in detail.
- Do not share your code with other students.
  - Before or after submitting the project.
  - It is OK to *discuss* the project.
- Do not copy any other student's work.
  - Don't *look at* anyone else's code
  - Don't let anyone look at your code.

Ground Rules

Except for code posted on the class web site

- Do not copy code from the Internet
  - or any other source.
- Do not ask for help on an Internet forum.
- Write your own code.

Submission

- Project is due by 11:59 PM, Thursday, Jan 28.
- Deliverables:
  - Source code only.
  - Three files: Test_Ticket.cpp (Main Program),
    Ticket.h, Ticket.cpp
  - Zip the files and submit the .zip file
- Submit the files using the Canvas Assignment for this class.
- If you work with another student, include both names in the assignment comments.
  - One student submits the files
  - Other student should submit just a Canvas submission comment including both names.

End of Presentation