Today - Lecture #9

1. Learn about the "class" construct — Topic 3
2. Apply classes to the "show" list problem

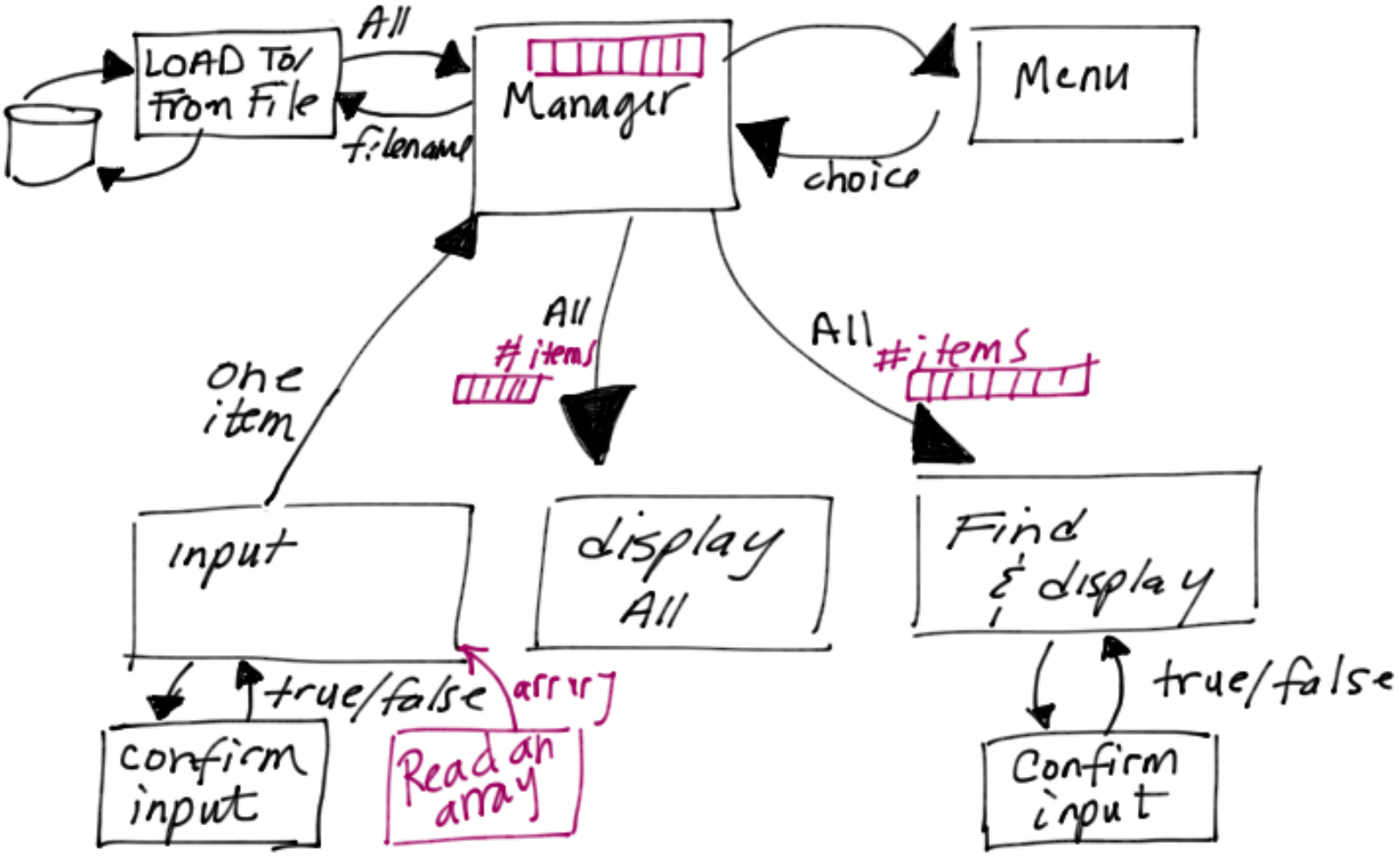
Announcements

* Make sure to watch class lectures and practice with small programs

During Lecture 8....

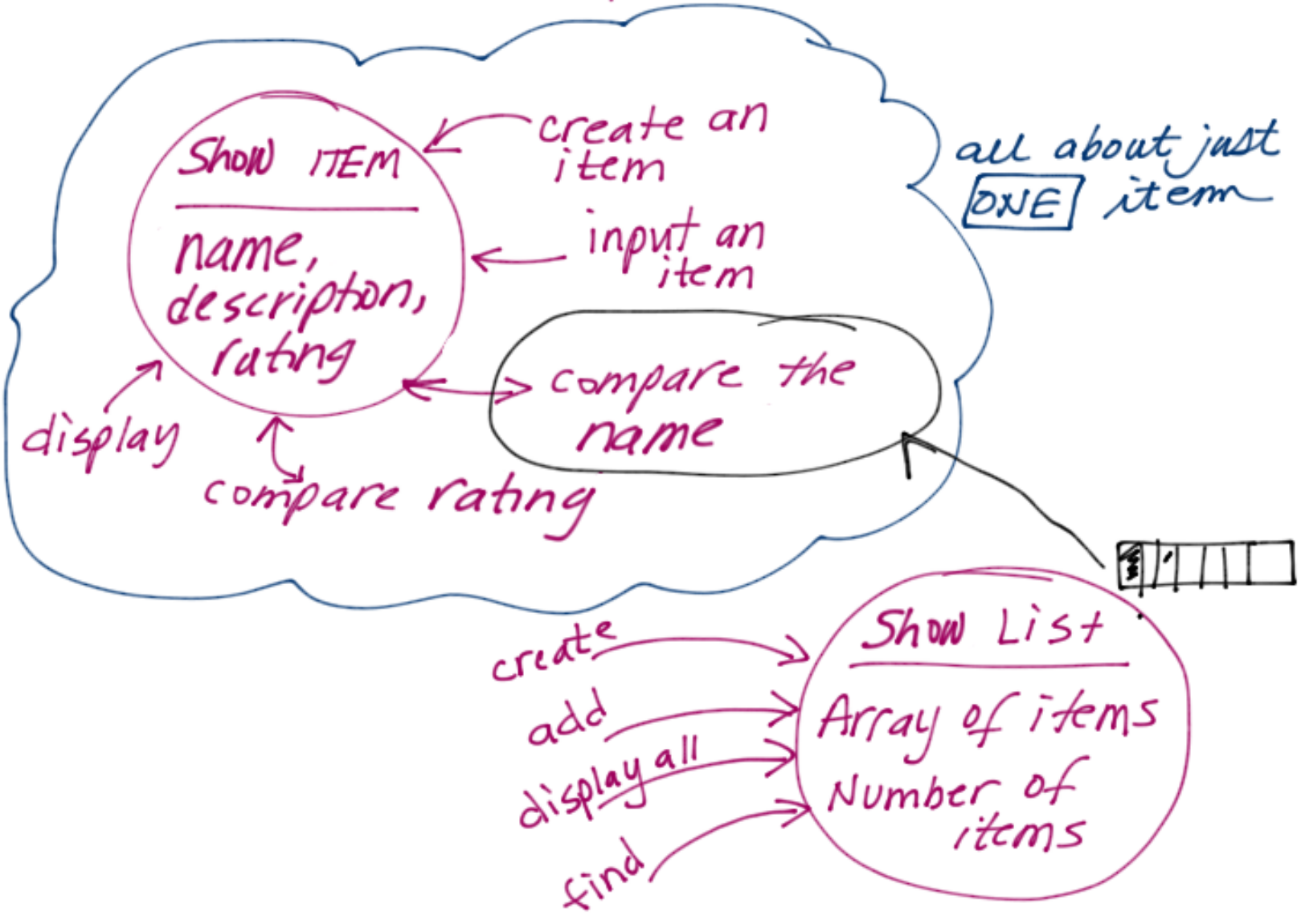
1) A "show" item has a title, description, etc.

2) Allow the user to enter these, find the highest rating, find a particular show based on name and display all. (and more!)



Now... using classes

1) Design the solution by thinking about the data and thinking about what operations make sense working on that data — GROUPING IT TOGETHER



Start small & incrementally Implement

```
class show_item   // one
{  public:        // interface
     show_item();          //constructor
     int add();
     int add( char name[], char des[], int rating);
     int display_all();
     bool find (char match[]);

   private:       // data
     char name[41];
     char description [131];
     int rating ;

};
```

initialize data members

data members

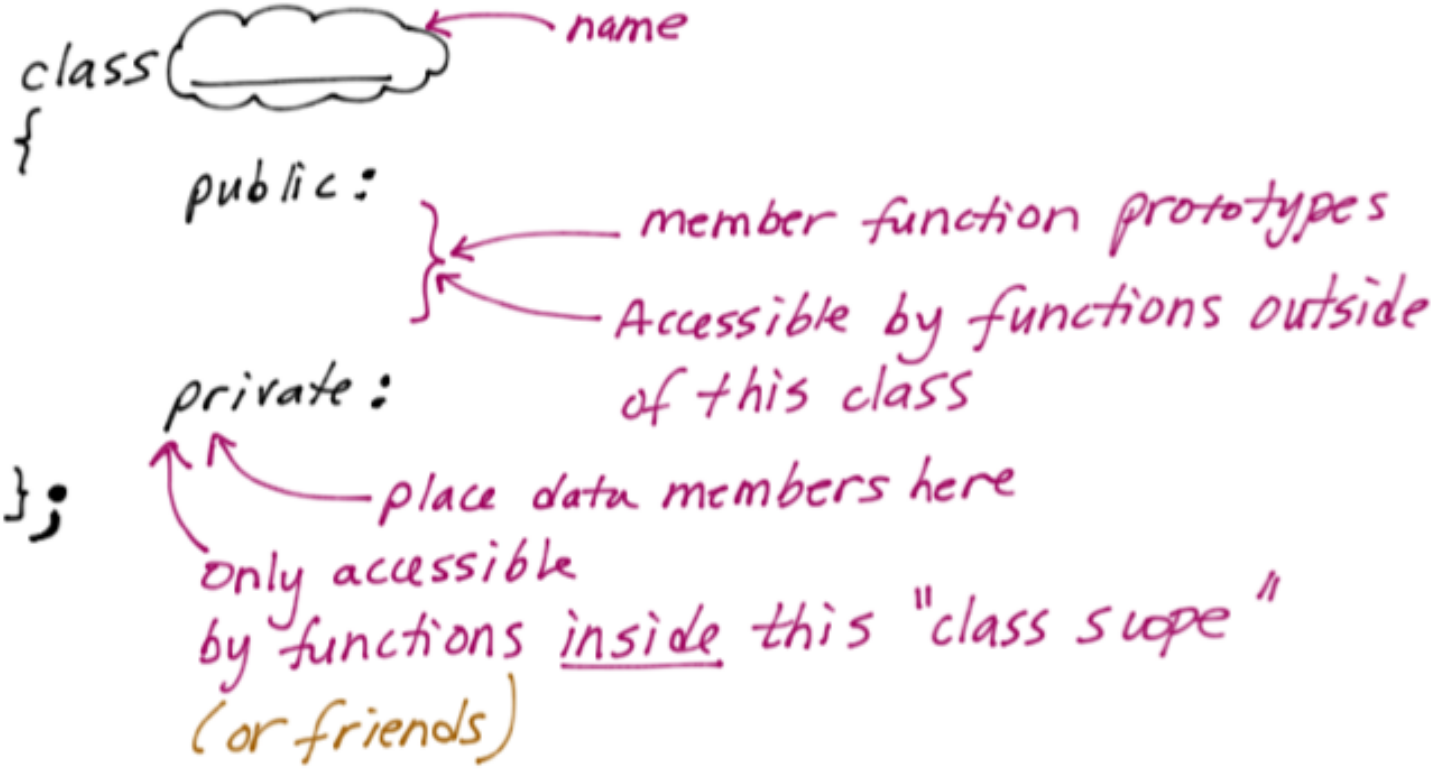
mandatory

Class Construct

class ⟶ data type

object ⟶ variable, instance of a data type

class interface ⟶ where we declare functions (function prototypes) and specify the data that will be available for all objects of this class.



Multiple Files

.h file (declarations)

1) #includes
2) constants
3) structs also prototypes
4) class interfaces
5) DO NOT implement the "body" of functions in the .h file
6) DO NOT #include any .cpp file

.cpp files (implementation file)

1) #include "~.h" ← goes to your current working directory
2) Function bodies ← function definitions
3) There can be only 1 main function in all of the .cpp files put together

On unix, compile via:

```
g++ main.cpp video.cpp
```

or

```
g++ *.cpp
```

this works if all of the functions in your directory are part of this "project"

To use the gdb or ddd debuggers, compile with the -g option

```
g++ -g *.cpp
```

When implementing member functions

1) In the .cpp file ALL prototypes listed in the class interface (.h) MUST be implemented

2) Preceed function name with the class name and the scope resolution operator (::)

```
video::video()
{
    // body of the function
}

void video::display()
{
    // body of the function
}
```