Outline

• Class Interface Declaration
• Access Control
• Constructors and Destructors
• Instance Declaration
class Ex {

public:

    // interface visible to the user goes here

private:

    // hidden declarations go here

};
Access Control

• Public section
  – Declare interface (usually only methods)
  – Usable anywhere outside of class

• Private
  – Prevent access outside of class
  – Primarily attributes (data), some methods
Constructors

- Responsible for initializing new objects
- Default: `Ex();`
- Copy: `Ex(const Ex& f);`
- User:
  - `Ex(char* name);`
  - `Ex(char* name, int level);`
Instance Declaration

• Use class name like built in type
• Declaration uses different constructors
  – Ex fst; // uses default
  – Ex scd (fst); //copy
  – Ex thd (“Third”); //user
  – Ex fth = “Third”; //user
• When decl. comes in scope, object created
Constructors and Initialization

- Sequence of object creation:
  1. Create storage for object
  2. Initialize storage
  3. Perform body of constructor

- Initializer list
  - \texttt{Ex(char* name): nm(name), lvl(0)}
    
  - List attributes in same order as declared in class
Destructors

• Responsible for properly destroying object
• Prototype: \texttt{~Ex()} 
• Declare one even if you don’t need it:
  \texttt{~Ex()} \{ \}
• Important when
  – have pointers as field
  – get around to learning inheritance
(Other) Methods

• Declare prototypes in class declaration
  
  ```cpp
  bool hasName(char *nme);
  ```

• Note: no return type for constructors and destructors

• Using methods
  – Inside class: hasName(“Default”) //unqualified
  – Outside class: fst.hasName(“Map”);
Parameters

• Pass-by-value
  
  \[
  \text{bool hasName(string nme);}\
  \]

• Pass-by-reference
  
  \[
  \text{bool hasName(string& nme);}\
  \]

• What happens when pass complex object by value?
Const and Methods

- Proper use of const allows compiler to help find errors
- Method that doesn’t change object
  ```cpp
  bool hasName(string nm) const;
  ```
- Method that doesn’t change ref parameter
  ```cpp
  void setName(const string& nm);
  ```
- Method that gives reference to internal object
  ```cpp
  const string& name() const;
  ```
Assignment Method

- Overloaded operator
  Frame& operator=(const Frame&);
- Usually similar to copy constructor
- If have pointers as fields
  - Check not doing something like a = a;
  - Delete pointer values
  - Copy values
- End with return *this;
Matters of Necessity

If you want to use your class like a built-in type, you must include

- A default constructor
- A copy constructor
- An assignment operator
- A destructor
Matters of Style

• One class to one pair of files
• Use class name as file name
• Public first, private second (for class user)
• Only prototypes in class declaration
• Function definitions in implementation file
⇒ More advanced language features force us to break some of these rules