CLASSES II

CONSTRUCTORS

Purpose:	**to initialize data members of newly created objects**
Syntax:	**ClassName(** *parameter list* **):** *initialization list* **{** *code* **}**
Default:	**automatically created constructor with no parameters**
User Defined:	**constructors with one or more parameters**
Delegation:	**constructors can call each other to eliminate repetitive code**

CONST MEMBER FUNCTIONS

const: anything that is const (a constant) cannot be modified

Syntax: returnType functionName( ) const { }

Purpose: the calling object is set to const within the function

STATIC MEMBERS

Purpose: maintain class data separate from object data

Syntax:

```
static int value;
static int getValue( ) { return value; }
```

Example: A car dealer has an inventory program with a Car class. Should each Car object contain a data member that tracks the total number of Car objects?

Note: const static members can be defined in the class (inline)
non-const static members must be defined externally

INLINE FUNCTIONS

Purpose: replace function calls with local code to increase performance

Syntax: `inline int getValue( ) { return value; }`

Classes: member functions defined internally are automatically inline
member functions defined externally are out-of-line

Note: recommended for small commonly used functions
increases size of program executable

CLASS COMPOSITION

Purpose: classes can contain data of other class types
composition is a has-a object relationship

Examples: a course has-a professor

a car has-an engine

a family has-one-or-more persons

Requirement: if class A contains a member of class B,
class B must be declared before class A

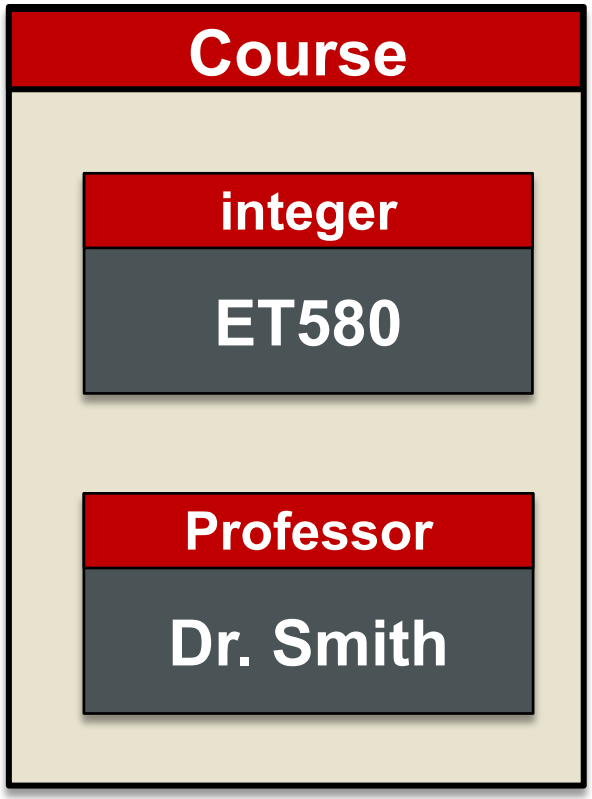
COMPOSITION BY VALUE

Purpose: used when object lifetimes are dependent

Concept: if class A contains a member of class B stored by value,
the lifetime of A and B objects are linked,
B does not exist without A and vice versa

Examples: a human has-a heart

CLASS COMPOSITION BY VALUE



a **Course** object contains an object of type **Professor**

when a Course object is deleted, its Professor object is also deleted

thus the lifetime of the Course object and its data members are **dependent**

e.g. when a **Human** dies, so does its **Heart**

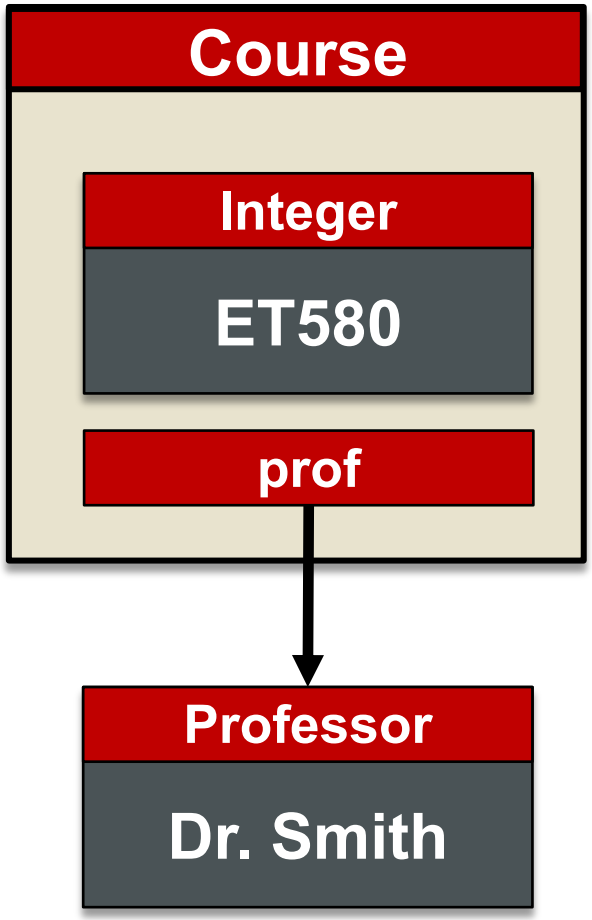
COMPOSITION BY REFERENCE

Purpose: used when object lifetimes are independent

Concept: if class A contains a member of class B,
the lifetime of A objects and B objects are distinct,
objects of A or B or both can exist without the other

Examples: a student has-one-or-more courses,
but a student can drop a course,
and a course can drop a student,
yet both the student and course will still exist

CLASS COMPOSITION BY REFERENCE



a **Course** object contains an **Integer**
and a reference to a distinct **Professor** object

when a Course object is deleted,
its reference **prof** is deleted,
but the Professor object lives on

thus the lifetime of the Course object
and Professor object are **independent**

e.g. when a **Patient** dies, their **Doctor** remains

NESTED CLASSES

Purpose: a form of composition where the one class is hidden within another class for abstraction

typically the nested class is not meant to be instantiated on its own

Requirements: class A contains a class declaration for class B

class B declaration precede B object declarations

typically class B declaration is in class A private area

typically class B members are all public

TEMPORARY OBJECTS

Concept: an object that without a reachable memory address
an object without a name

Syntax: ClassName( )

Purpose: temporary objects store data for pass by value operations

Example: a function returns an object by value,
this creates a temporary object to transfer
data back to the calling function

L-VALUE VS R-VALUE

L-VALUE:	any expression that resolves to a memory address	
R-VALUE:	any expression that is not an l-value	
L-VALUES:	int a;	a variable
	string s;	a named object
	const double d = 5;	a const l-value
R-VALUES:	5;	a literal
	string( );	a temporary object