W6.1

- · Destructors
- Data Members and Member Functions
- Returning a Reference to a Private Data Member
- Default Memberwise Copy
- Software Reusability

© 2000 Prentice Hall Inc. All rights reserved

• Destructors - Are member function of class - Perform termination housekeeping before the system reclaims the object's memory - Complement of the constructor - Name is tilde (~) followed by the class name (i.e., ~Time) • Recall that the constructor's name is the class name - Receives no parameters, returns no value - One destructor per class • No overloading allowed

6.13 When Constructors and Destructors Are Called

- · Constructors and destructors called automatically
 - Order depends on scope of objects
- Global scope objects
 - Constructors called before any other function (including main)
 - Destructors called when main terminates (or exit function called)
 - Destructors not called if program terminates with abort
- · Automatic local objects
 - Constructors called when objects are defined
 - Destructors called when objects leave scope
 - i.e., when the block in which they are defined is exited
 - Destructors not called if the program ends with ${\tt exit}$ or ${\tt abort}$

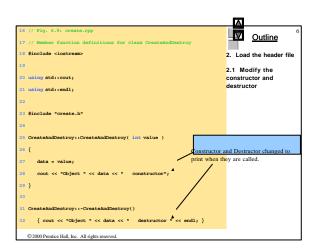
© 2000 Prentice Hall, Inc. All rights reserved.

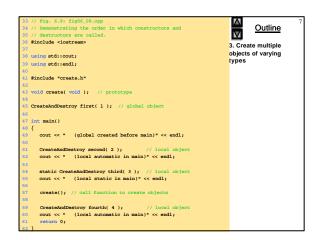
6.13 When Constructors and Destructors Are Called

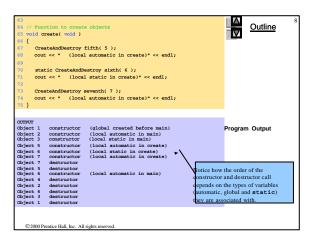
- · Static local objects
 - Constructors called when execution reaches the point where the objects are defined
 - Destructors called when main terminates or the exit function is called
 - Destructors not called if the program ends with abort

©2000 Prentice Hall, Inc. All rights reserved.

```
1 // Fig. 6.9: create.h
2 // Definition of class CreateAndDestroy.
3 // Member functions defined in create.cpp.
4 #ifndef CREATE,H
5 # Medine CREATE,H
6 # Lass CreateAndDestroy {
8 public|
9 CreateAndDestroy(int); // constructor
10 -CreateAndDestroy(); // destructor
11 privates
12 int data;
13 };
14 # Lass CreateAndDestroy(); // destructor
11 privates
12 int data;
13 };
14 # Lass CreateAndDestroy(); // destructor
13 privates
15 # endif
```







6.14 Using Data Members and Member Functions

- · Member functions
 - Allow clients of the class to set (i.e., write) or get (i.e., read) the values of private data members
 - Example:

Adjusting a customer's bank balance

- private data member balance of a class BankAccount could be modified through the use of member function computeInterest
- A member function that sets data member interestRate could be called setInterestRate, and a member function that returns the interestRate could be called getInterestRate
- Providing *set* and *get* functions does not make **private** variables **public**
- A set function should ensure that the new value is valid

© 2000 Prentice Hall, Inc. All rights reserved.

6.15 A Subtle Trap: Returning a Reference to a Private Data Member

- · Reference to an object
 - Alias for the name of the object,
 - May be used on the left side of an assignment statement, makes perfectly acceptable *lvalue*.
 - Reference can receive a value, which changes the original object as well
- · Returning references
 - public member functions can return non-const references to private data members
 - · Should be avoided, breaks encapsulation
- Please avoid using references in this way, very, very bad!!!

© 2000 Prentice Hall, Inc. All rights reserved.

