8.11 static Class Members

- In certain cases, only one copy of a particular variable should be shared by all objects of a class.
  - A static field—called a class variable—is used in such cases.
- A static variable represents classwide information—all objects of the class share the same piece of data.
  - The declaration of a static variable begins with the keyword static.
8.11 static Class Members (Cont.)

• A static method cannot access non-static class members, because a static method can be called even when no objects of the class have been instantiated.
  ▪ For the same reason, the this reference cannot be used in a static method.
  ▪ The this reference must refer to a specific object of the class, and when a static method is called, there might not be any objects of its class in memory.
• If a static variable is not initialized, the compiler assigns it a default value.
Common Programming Error 8.7

A compilation error occurs if a static method calls an instance (non-static) method in the same class by using only the method name. Similarly, a compilation error occurs if a static method attempts to access an instance variable in the same class by using only the variable name.

Fig. 8.12  |  static variable used to maintain a count of the number of Employee objects in memory. (Part 1 of 2)
Fig. 8.12 | static variable used to maintain a count of the number of Employee objects in memory. (Part 2 of 2.)

Good Programming Practice 8.1
Invoke every static method by using the class name and a dot (.) to emphasize that the method being called is a static method.
public class EmployeeTest {
    public static void main(String[] args) {
        // show that count is 0 before creating Employees
        System.out.printf("Employees before instantiation: %d\n", Employee.getCount());

        // create two employees; count should be 2
        Employee e1 = new Employee("Susan", "Baker");
        Employee e2 = new Employee("Bob", "Blue");

        // show that count is 2 after creating two Employees
        System.out.printf("Employees after instantiations: %d\n", Employee.getCount());
        System.out.printf("via e1.getCount(): %d\n", e1.getCount());
        System.out.printf("via e2.getCount(): %d\n", e2.getCount());
        System.out.printf("via Employee.getCount(): %d\n", Employee.getCount());
    }
}

Fig. 8.13 | static member demonstration (Part 1 of 2)

// get names of Employees
System.out.printf("Employee 1: %s %s, Employee 2: %s %s\n", e1.getFirstName(), e1.getLastName(), e2.getFirstName(), e2.getLastName());

// in this example, there is only one reference to each Employee, so the following two statements indicate that these objects are eligible for garbage collection
\n\n e1 = null;
 e2 = null;
\n} // end main

1 // end Class EmployeeTest

Employees before instantiation: 0
Employee constructor: Susan Baker; count = 1
Employee constructor: Bob Blue; count = 2

Employees after instantiation:
via e1.getCount(): 2
via e2.getCount(): 2
via Employee.getCount(): 2

Employee 1: Susan Baker
Employee 2: Bob Blue

Fig. 8.13 | static member demonstration (Part 2 of 2)