

Chapter 7 Objects and Classes

1. See the section "Declaring and Creating Objects."
2. Constructors are special kinds of methods that are called when creating an object using the new operator. Constructors do not have a return type—not even void.
3. An array is an object. The default value for the elements of an array is 0 for numeric, false for boolean, '\u0000' for char, null for object element type.
4. (a) There is such constructor ShowErrors(int) in the ShowErrors class.

The ShowErrors class in the book has a default constructor. It is actually same as

```
public class ShowErrors {  
    public static void main(String[] args) {  
        ShowErrors t = new ShowErrors(5);  
    }  
    public ShowErrors () {  
    }  
}
```

On Line 3, new ShowErrors(5) attempts to create an instance using a constructor ShowErrors(int), but the ShowErrors class does not have such a constructor. That is an error.

(b) x() is not a method in the ShowErrors class.

The ShowErrors class in the book has a default constructor. It is actually same as

```
public class ShowErrors {  
    public static void main(String[] args) {  
        ShowErrors t = new ShowErrors();  
        t.x();  
    }  
    public ShowErrors () {  
    }  
}
```

On Line 4, `t.x()` is invoked, but the `ShowErrors` class does not have the method named `x()`. That is an error.

(c) The program compiles fine, but it has a runtime error because variable `c` is null when the println statement is executed.

(d) `new C(5.0)` does not match any constructors in class `C`. The program has a compilation error because class `C` does not have a constructor with a double argument.

5. The program does not compile because new A() is used in class `Test`, but class `A` does not have a default constructor. See the second NOTE in the Section, "Constructors."
6. false
7. Use the `Date`'s no-arg constructor to create a `Date` for the current time. Use the `Date`'s `toString()` method to display a string representation for the `Date`.
8. Use the `JFrame`'s no-arg constructor to create `JFrame`. Use the `setTitle(String)` method to set a title and use the `setVisible(true)` method to display the frame.
9. `Date` is in `java.util`. `JFrame` and `JOptionPane` are in `javax.swing`. `System` and `Math` are in `java.lang`.
10. `System.out.println(f.i);`
Answer: Correct

`System.out.println(f.s);`
Answer: Correct

`f.imethod();`
Answer: Correct

`f.smethod();`
Answer: Correct

`System.out.println(Foo.i);`
Answer: Incorrect

`System.out.println(Foo.s);`
Answer: Correct

`Foo.imethod();`
Answer: Incorrect

```
Foo.smethod();
```

Answer: Correct

11. Add static in the main method and in the factorial method because these two methods don't need reference any instance objects or invoke any instance methods in the Test class.

12. You cannot invoke an instance method or reference an instance variable from a static method. You can invoke a static method or reference a static variable from an instance method? c is an instance variable, which cannot be accessed from the static context in method2.

13. Accessor method is for retrieving private data value and mutator method is for changing private data value. The naming convention for accessor method is getDataFieldName() for non-boolean values and isDataFieldName() for boolean values. The naming convention for mutator method is setDataFieldName(value).

14. Two benefits: (1) for protecting data and (2) for easy to maintain the class.

15. Yes. Though radius is private, myCircle.radius is used inside the Circle class. Thus, it is fine.

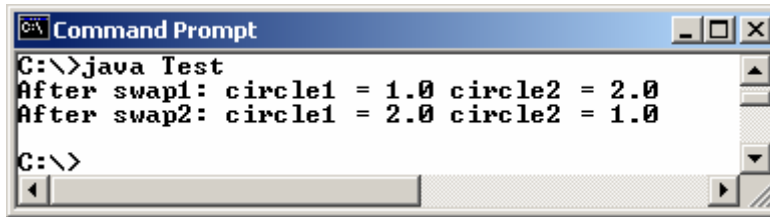
16. No. It must also contain no get methods that would return a reference to a mutable data field object.

17. Yes.

18. Java uses "pass by value" to pass parameters to a method. When passing a variable of a primitive type to a method, the variable remains unchanged after the method finishes. However, when passing a variable of a reference type to a method, any changes to the object referenced by the variable inside the method are permanent changes to the object referenced by the variable outside of the method. Both the actual parameter and the formal parameter variables reference to the same object.

The output of the program is as follows:
count 101
times 0

19.



```
Command Prompt
C:\>java Test
After swap1: circle1 = 1.0 circle2 = 2.0
After swap2: circle1 = 2.0 circle2 = 1.0
C:\>
```

Remark: The reference value of circle1 is passed to x and the reference value of circle2 is passed to y. The contents of the objects are not swapped in the swap1 method. circle1 and circle2 are not swapped. To actually swap the contents of these objects, replace the following three lines

```
Circle temp = x;
x =y;
y =temp;
```

by

```
double temp = x.radius;
x.radius = y.radius;
y.radius = temp;
```

as in swap2.

- 20. a. a[0] = 1 a[1] = 2
- b. a[0] = 2 a[1] = 1
- c. e1 = 2 e2 = 1
- d. t1's i = 2 t1's j = 1
- t2's i = 2 t2's j = 1

- 21. i + j is 23 (because i (value of 2) is concatenated with string j + j is, then j (value of 3) is concatenated with string i + j is 2.)
- k is 2
- j is 0

- 22. See the section on the this keyword. The value of parameter p is assigned to parameter p in Line 5. It should be this.p = p;

- 23. (Line 4 prints null since dates[0] is null. Line 5 causes a NullPointerException since it invokes toString() method from the null reference.)

- 24. No. The Loan class has the getLoanDate() method that returns loanDate. loanDate is an object of the Date class. Since Date is mutable, the contents of loanDate can be changed. So, the Loan class is not immutable.

25. The StackOfInteger class is immutable, because the contents of a stack can be changed through the pop and push methods.