5

Control Statements (Cont.)

Last time

- overview of control statements in Java
- if and if-else statements
- while, do-while, and for loop
- compound and increment/decrement operators
- GUI programming: simple graphics drawing

Objectives

- developer conferences
- review control statements
- break and continue
- unary cast operator
- GUI: colors and Unicode character set

Upcoming Developer Conferences

- Facebook f8 (March 25-26)
  - http://fbf8.com
- Google I/O (May 28-29)
  - https://events.google.com/io2015/
- Apple WWDC (usually in June)
- JavaOne (Oct 25-29)
  - https://www.oracle.com/javaone/index.html

Developer Conferences (JavaOne)

Outline

Control Statements Review

- *Selection* Statements
  - `if – else if – else` statement
    - Standard selection statement
  - `switch` statement
    - Multiple-selection statement

Control Statements (Cont.)

- *Repetition (looping)* **statements**
  - Repeatedly performs an action while its loop-continuation condition remains true
  - `while` statement
    - Performs the actions in its body zero or more times
  - `do…while` statement
    - Performs the actions in its body one or more times
  - `for` statement
    - Performs the actions in its body zero or more times

switch Multiple-Selection Statement

- switch statement
  - Used for multiple selections

```
switch (key) {
        case value:
                // your code here
                break;

        default:
                break;
}
```

`switch` Multiple-Selection Statement

- Expression in each `case`
  - Constant integral expression
    - Combination of integer constants that evaluates to a constant integer value
  - Character constant
    - E.g., 'A', '7' or '$'
  - Constant variable
    - Declared with keyword `final`
  - `String` constant since Java 7+

break and continue Statements

- break/continue
  - Alter flow of control
- break statement
  - Causes immediate exit from control structure
    - Used in while, for, do…while or switch statements
- continue statement
  - Skips remaining statements in loop body
  - Proceeds to next iteration
    - Used in while, for or do…while statements

Outline

```
1  // Fig. 5.12: BreakTest.java
2  // break statement exiting a for statement.
3  public class BreakTest
4  {
5     public static void main( String args[] )
6     {
7        int count; // control variable also used
8
9        for ( count = 1; count <= 10; count++ ) // loop 10 times
10       {
11          if ( count == 5 ) // if count is 5,
12             break;         // terminate loop
13
14          System.out.printf( "%d ", count );
15       } // end for
16
17       System.out.printf( "\nBroke out of loop at count = %d\n", count );
18    } // end main
19 } // end class BreakTest
```

Loop 10 times

Exit **for** statement (break) when count equals 5

```
1 2 3 4
Broke out of loop at count = 5
```

BreakTest.java

Line 9

Lines 11-12

Program output

Outline

```
 1 // Fig. 5.13: ContinueTest.java
 2 // continue statement terminating an iteration of a for statement.
 3 public class ContinueTest
 4 {
 5    public static void main( String args[] )
 6    {
 7       for ( int count = 1; count <= 10; count++ )
 8       {
 9          if ( count == 5 ) // if count is 5,
10             continue;      // skip remaining code in loop
11
12          System.out.printf( "%d ", count );
13       } // end for
14
15       System.out.println( "\nUsed continue to skip printing 5" );
16    } // end main
17 } // end class ContinueTest
```

Loop 10 times

Skip line 12 and proceed to line 7 when `count` equals 5

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing 5
```

`ContinueTest.java`

`Line 7`

`Lines 9-10`

`Program output`

Unary cast operator

- Unary cast operator
  - Creates a temporary copy of its operand with a different data type
    - example: `(double)` will create a temporary floating-point copy of its operand
  - Explicit conversion
- Promotion
  - Converting a value (e.g. `int`) to another data type (e.g. `double`) to perform a calculation
  - Implicit conversion

GUI

GUI: JFrame class

- `JFrame` class from the `javax.swing` package
  - Allows the programmer to create a window
  - Also allows customization of the window:
    - `setDefaultCloseOperation` method
    - `setSize` method
    - Many other options
  - `add` method
    - Attaches a `JPanel` to the `JFrame`

GUI: JPanel class

- The `JPanel` class
  - Every `JPanel` has a `paintComponent` method
    - `paintComponent` is called whenever the system needs to display the `Jpanel`
    - Provides “double-buffering” for smoother graphics
  - `getWidth` and `getHeight` methods
    - Return the width and height of the `JPanel`, respectively

GUI: Drawing Rectangles and Ovals

- **Draw rectangles**
  - **Method** `drawRect` **of** `Graphics`
- **Draw ovals**
  - **Method** `drawOval` **of** `Graphics`

GUI: Colors

- `Color` class of package `java.awt`
  - Represented as RGB (red, green and blue) values
  - Each component has a value from 0 to 255
  - 13 predefined `static` `Color` objects:
    - `Color.Black`, `Color.BLUE`, `Color.CYAN`, `Color.DARK_GRAY`, `Color.GRAY`, `Color.GREEN`, `Color.LIGHT_GRAY`, `Color.MAGENTA`, `Color.ORANGE`, `Color.PINK`, `Color.RED`, `Color.WHITE` and `Color.YELLOW`

GUI: Colors and Filled Shapes

- `fillRect` and `fillOval` methods of `Graphics` class
  - Similar to `drawRect` and `drawOval` but draw rectangles and ovals filled with color
- `setColor` method of `Graphics` class
  - Set the current drawing color (for filling rectangles and ovals drawn by `fillRect` and `fillOval`)

GUI example: Gradient panel

The Unicode Standard

- **Computer industry standard**
- **Encoding, representation, and handling of text for most of world's writing systems**
- **110,000+ characters covering 100+ scripts**
- **Unicode code (*e.g.*, smiley U+263A, Java \u263A)**

GUI example: Unicode character set