More Conditionals and Loops

• Now we can fill in some additional details regarding Java conditional and repetition statements

• Chapter 6 focuses on:
  – the switch statement
  – the conditional operator
  – the do loop
  – the for loop
  – drawing with the aid of conditionals and loops
  – dialog boxes
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
The switch Statement

• The *switch* statement provides another way to *decide* which *statement* to *execute* next

• The *switch* statement *evaluates* an expression, then attempts to *match* the result to one of several possible *cases*

• Each case contains a value and a list of statements

• The flow of control *transfers* to statement associated with the *first case value* that matches
The general syntax of a `switch` statement is:

```java
switch (expression) {
    case value1:
        statement-list1
        break;
    case value2:
        statement-list2
        break;
    case value3:
        statement-list3
        break;
    default:
        statement-list
        break;
}
```

- `switch` and `case` are reserved words.
- If `expression` matches `value2`, control jumps to here.

Copyright © 2012 Pearson Education, Inc.
The switch Statement

• An example of a switch statement:

```java
switch (option)
{
    case 'A':
        aCount++;
        break;
    case 'B':
        bCount++;
        break;
    case 'C':
        cCount++;
        break;
}
```
The switch Statement

• Often a break statement is used as the last statement in each case's statement list

• A break statement causes control to transfer to the end of the switch statement

• If a break statement is not used, the flow of control will continue into the next case

• Sometimes this may be appropriate, but (more) often we want to execute only the statements associated with one case
The switch Statement

• A switch statement can have an optional default case

• The default case has no associated value and simply uses the reserved word default

• If the default case is present, control will transfer to it if no other case value matches

• If there is no default case, and no other value matches, control falls through to the statement after the switch
The switch Statement

- The type of a switch expression must be **integers**, **characters**, or **enumerated types**

- As of **Java 7**, a switch can also be used with **strings**

- You cannot use a switch with floating point values

- The implicit boolean condition in a **switch** statement is equality

- You **cannot** perform **relational checks** with a **switch** statement

- See **GradeReport.java**
import java.util.Scanner;

public class GradeReport
{
    //-----------------------------------------------------------------
    // Reads a grade from the user and prints comments accordingly.
    //-----------------------------------------------------------------
    public static void main (String[] args)
    {
        int grade, category;

        Scanner scan = new Scanner (System.in);

        System.out.print ("Enter a numeric grade (0 to 100): ");
        grade = scan.nextInt();

        category = grade / 10;

        System.out.print ("That grade is ");
continue

    switch (category)
    {
        case 10:
            System.out.println("a perfect score. Well done.");
            break;
        case 9:
            System.out.println("well above average. Excellent.");
            break;
        case 8:
            System.out.println("above average. Nice job.");
            break;
        case 7:
            System.out.println("average.");
            break;
        case 6:
            System.out.println("below average. You should see the");
            System.out.println("instructor to clarify the material "+"presented in class.");
            break;
        default:
            System.out.println("not passing.");
    }
}
Sample Run

Enter a numeric grade (0 to 100): 91
That grade is well above average. Excellent.
Switch and enum

public class SwitchExample {

    private enum Color {red, green, blue}

    public static void main(String[] args) {
        Color col = Color.blue;
        switch (col) {
            // note that you do not write "case Color.blue:"
            case blue:
                System.out.println("It is blue");
                break;
            case red:
                System.out.println("It is red");
                break;
            case green:
                System.out.println("It is green");
                break;
            default:
                System.out.println("No color");
        }
    }
}
col = Color.green;

switch(col.name()) {

    case "blue":
        System.out.println("It is blue");
        break;
    case "red":
        System.out.println("It is red");
        break;
    case "green":
        System.out.println("It is green");
        break;
}
}
Quiz

You might choose to use a switch statement instead of nested if-else statements if

A) the variable being tested might equal one of several hundred int values
B) the variable being tested might equal one of only a few int values
C) there are two or more int variables being tested, each of which could be one of several hundred values
D) there are two or more int variables being tested, each of which could be one of only a few values
E) none of the above, you would never choose to use a switch statement in place of nested if-else statements under any circumstance
Quiz

You might choose to use a switch statement instead of nested if-else statements if

A) the variable being tested might equal one of several hundred int values
B) the variable being tested might equal one of only a few int values
C) there are two or more int variables being tested, each of which could be one of several hundred values
D) there are two or more int variables being tested, each of which could be one of only a few values
E) none of the above, you would never choose to use a switch statement in place of nested if-else statements under any circumstance

The switch statement can only be used if there is a single variable being tested and it is an integral type (an int or a char in Java). Further, because you have to enumerate each possible value being tested, the switch statement only makes sense if the number of values being tested is a small number.
Exercise

• Write a new toString() method of the class Die that returns a string with the English name of the faceValue of the die – use switch
public String toString() {
    String result = "";
    switch (faceValue) {
    case 1:
        result = "One";
        break;
    case 2:
        result = "Two";
        break;
    case 3:
        result = "Three";
        break;
    case 4:
        result = "Four";
        break;
    case 5:
        result = "Five";
        break;
    case 6:
        result = "Six";
        break;
    }
    return result;
}
Quiz

If x is currently equal to 5, what will the value of x be after the switch statement executes?

```
switch (x) {
    case 3 : x += 1;
    case 4 : x += 2;
    case 5 : x += 3;
    case 6 : x++;  // Error: Case 6 is not handled
    case 7 : x += 2;
    case 8 : x--;  // Error: Case 8 is not handled
    case 9 : x++;  // Error: Case 9 is not handled
}
```

A) 8  B) 6  C) 11  D) 5  E) 10
Quiz

If x is currently equal to 5, what will the value of x be after the switch statement executes?

```java
switch (x) {
    case 3 : x += 1;
    case 4 : x += 2;
    case 5 : x += 3;
    case 6 : x++;  //case 8 : x--;   
    case 7 : x += 2;
    case 9 : x++;  //case 8 : x--;   
}
```

A) 8   B) 6   C) 11   D) 5   E) 10
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
The Conditional Operator

• The *conditional operator* evaluates to one of two expressions based on a boolean condition.

• Its syntax is:

  \[ \text{condition} \ ? \ \text{expression1} : \ \text{expression2} \]

• If the *condition* is true, *expression1* is evaluated; if it is false, *expression2* is evaluated.

• The *value* of the entire conditional operator is the value of the selected expression.

• It is an operator not a statement.
The Conditional Operator

• The conditional operator is similar to an if-else statement, except that it is an expression that returns a value

• For example:

\[
\text{larger} = ((\text{num1} > \text{num2}) \ ? \ \text{num1} \ : \ \text{num2});
\]

• If \text{num1} is greater than \text{num2}, then \text{num1} is assigned to \text{larger}; otherwise, \text{num2} is assigned to \text{larger}

• The conditional operator is ternary because it requires three operands
The Conditional Operator

• Another example:

    System.out.println("Your change is " + count +
    ((count == 1) ? "Dime" : "Dimes");

• If count equals 1, then "Dime" is the value of the conditional expression

• If count is anything other than 1, then the conditional operator evaluates to "Dimes"
Quick Check

Express the following logic in a succinct manner using the conditional operator.

```java
if (val <= 10)
    System.out.println("It is not greater than 10.");
else
    System.out.println("It is greater than 10.");
```
Express the following logic in a succinct manner using the conditional operator.

```java
if (val <= 10)
    System.out.println("It is not greater than 10.");
else
    System.out.println("It is greater than 10.");

System.out.println("It is" +
    ((val <= 10) ? " not" : ") +
    " greater than 10.");
```
Quiz

• The statement:

\[
\text{if } (x < 0) \ y = x; \text{ else } y = 0;
\]

can be rewritten using a conditional operator as:

A) \( y = (x < 0) \ ? x : 0; \)
B) \( x = (x < 0) \ ? y : 0; \)
C) \( (x < 0) \ ? y = x : y = 0; \)
D) \( y = (x < 0); \)
E) \( y = \text{if } (x < 0) \ x : 0; \)
Quiz

• The statement \( \text{if} \ (x < 0) \ y = x; \ \text{else} \ y = 0; \) can be rewritten using a conditional operator as:

A) \( y = (x < 0) \ ? \ x : 0; \)
B) \( x = (x < 0) \ ? \ y : 0; \)
C) \( (x < 0) \ ? \ y = x : y = 0; \) ← Remember the conditional operator is not a **statement**
D) \( y = (x < 0); \)
E) \( y = \text{if} \ (x < 0) \ x : 0; \)
Quiz

• Is this code syntactically correct? If yes, what is printing?

```java
int y = 1, x = 2, z;
z = (x < 0) ? (y = x) : (y = 0);
System.out.println("y = " + y);
```
Quiz

• What is the difference between a conditional operator and a conditional statement?
Quiz

• What is the difference between a conditional operator and a conditional statement?

• A conditional operator is a ternary operator that evaluates a condition and produces one of two possible values.

• A conditional statement (e.g., if) is a category of statements that allow conditions to be evaluated and the appropriate statements executed as a result.
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
The do Statement

• A *do statement* has the following syntax:

```c
do
{
    statement-list;
}
while (condition);
```

• The *statement-list* is executed once initially, and then the *condition* is evaluated.

• The statement is executed repeatedly until the condition becomes false.
Logic of a do Loop

Statement

Condition evaluated

true

false
The do Statement

• An example of a `do` loop:

```java
int count = 0;
do {
    count++;
    System.out.println(count);
} while (count < 5);
```

• The body of a `do` loop executes at least once

• See `ReverseNumber.java`
import java.util.Scanner;

public class ReverseNumber {
    // Reverses the digits of an integer mathematically.
    public static void main (String[] args) {
        int number, lastDigit, reverse = 0;
        Scanner scan = new Scanner (System.in);
        continue
continue

System.out.print("Enter a positive integer: ");
number = scan.nextInt();

do
{
    lastDigit = number % 10;
    reverse = (reverse * 10) + lastDigit;
    number = number / 10;
}
while (number > 0);

System.out.println("That number reversed is "+ reverse);
System.out.print("Enter a positive integer: ");
number = scan.nextInt();
do{
  lastDigit = number % 10;
  reverse = (reverse * 10) + lastDigit;
  number = number / 10;
}while (number > 0);
System.out.println("That number reversed is "+ reverse);
Comparing while and do

The **while** Loop

- condition evaluated
- true
- statement
- true
- false
- condition evaluated

The **do** Loop

- statement
- true
- condition evaluated
- false
- false
Quiz

How many times will the following loop iterate?

```java
int x = 10;
do {
    System.out.println(x);
    x--;
}while (x > 0);
```
Quiz

How many times will the following loop iterate?

```java
int x = 10;
do {
    System.out.println(x);
    x--;
} while (x > 0);
```

10 times
Exercise

• Write a code fragment (using do loop) that prints the characters stored in a String object called str backwards. Hint: use the method of String charAt(int).
Solution

String s = "arbatax";
int index = s.length() - 1;
do {
    System.out.print(s.charAt(index));
    index--;
} while (index >= 0);

• But if the string is "" , this code produces a run time error
• We must do a test before entering the loop
• Better use a while loop here
Alternative solution

String s = "arbatax";
int index = s.length() - 1;
while (index >=0) {
    System.out.print(s.charAt(index));
    index--;
}

Outline

- The `switch` Statement
- The Conditional Operator
- The `do` Statement
- The `for` Statement
- Drawing with Loops and Conditionals
- Dialog Boxes
The for Statement

• A *for statement* has the following syntax:

```java
for (initialization; condition; increment) statement;
```

The *initialization* is executed once before the loop begins.

The *statement* is executed until the *condition* becomes false.

The *increment* portion is executed at the end of each iteration.
Logic of a for loop

- **initialization**
- **condition evaluated**
  - **true**
  - **statement**
  - **increment**
  - **false**
Example of a for loop

• i is initialized to 0
• i is compared to 5
  – if this is true x is incremented with i
• i is incremented by 1
• i is compared again with 5
  – ...

```java
int x = 0;
for (int i = 0; i < 5; i++)
    x += i;
```
The for Statement

• A for loop is functionally equivalent to the following while loop structure:

```
initialization;
while ( condition )
{
    statement;
    increment;
}
```
The for Statement

• An example of a for loop:

   for (int count=1; count <= 5; count++)
       System.out.println (count);

• The initialization section can be used to declare a variable

• Like a while loop, the condition of a for loop is tested prior to executing the loop body

• Therefore, the body of a for loop will execute zero or more times
The for Statement

• The increment section can perform any calculation:

```java
for (int num=100; num > 0; num -= 5)
    System.out.println (num);
```

• A for loop is well suited for executing statements a specific number of times that can be calculated or determined in advance

• See Multiples.java

• See Stars.java
import java.util.Scanner;

public class Multiples {
    // Prints multiples of a user-specified number up to a user-specified limit.
    public static void main (String[] args) {
        final int PER_LINE = 5;
        int value, limit, mult, count = 0;

        Scanner scan = new Scanner (System.in);

        System.out.print ("Enter a positive value: ");
        value = scan.nextInt();

        continue
System.out.print("Enter an upper limit: ");
limit = scan.nextInt();

System.out.println();
System.out.println("The multiples of " + value + " between " + value + " and " + limit + " (inclusive) are:");

for (mult = value; mult <= limit; mult += value)
{
    System.out.print(mult + "\t");

    // Print a specific number of values per line of output
    count++;
    if (count % PER_LINE == 0)
        System.out.println();
}
}
```java
continues

```Sample Run
```markdown
Enter a positive value: 7
Enter an upper limit: 400

The multiples of 7 between 7 and 400 (inclusive) are:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
</tr>
<tr>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
<td>70</td>
</tr>
<tr>
<td>77</td>
<td>84</td>
<td>91</td>
<td>98</td>
<td>105</td>
</tr>
<tr>
<td>112</td>
<td>119</td>
<td>126</td>
<td>133</td>
<td>140</td>
</tr>
<tr>
<td>147</td>
<td>154</td>
<td>161</td>
<td>168</td>
<td>175</td>
</tr>
<tr>
<td>182</td>
<td>189</td>
<td>196</td>
<td>203</td>
<td>210</td>
</tr>
<tr>
<td>217</td>
<td>224</td>
<td>231</td>
<td>238</td>
<td>245</td>
</tr>
<tr>
<td>252</td>
<td>259</td>
<td>266</td>
<td>273</td>
<td>280</td>
</tr>
<tr>
<td>287</td>
<td>294</td>
<td>301</td>
<td>308</td>
<td>315</td>
</tr>
<tr>
<td>322</td>
<td>329</td>
<td>336</td>
<td>343</td>
<td>350</td>
</tr>
<tr>
<td>357</td>
<td>364</td>
<td>371</td>
<td>378</td>
<td>385</td>
</tr>
<tr>
<td>392</td>
<td>399</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
public class Stars {
    // Prints a triangle shape using asterisk (star) characters.
    public static void main (String[] args) {
        final int MAX_ROWS = 10;
        for (int row = 1; row <= MAX_ROWS; row++) {
            for (int star = 1; star <= row; star++)
                System.out.print ("*");
            System.out.println();
        }
    }
}
public class Stars {
    
    public static void main (String[] args) {
        final int MAX_ROWS = 10;
        for (int row = 1; row <= MAX_ROWS; row++) {
            for (int star = 1; star <= row; star++)
                System.out.print (“*”);
            System.out.println();
        }
    }
}
Exercise

How many iterations will the following for loops execute?

a for (int i = 0; i < 20; i++) {} 
b for (int i = 0; i <= 20; i++) {} 
c for (int i = 5; i < 20; i++) {} 
d for (int i = 20; i > 0; i--) {} 
e for (int i = 1; i < 20; i=i+2) {} 
f for (int i = 1; i < 20; i*=2) {}
Exercise

How many iterations will the following for loops execute?

a for (int i = 0; i < 20; i++) {} 20
b for (int i = 0; i <= 20; i++) {} 21
c for (int i = 5; i < 20; i++) {} 15
d for (int i = 20; i > 0; i--) {} 20
e for (int i = 1; i < 20; i=i+2) {} 10
f for (int i = 1; i < 20; i*=2) {} 5
Quiz

Given that s is a String, what does the following loop do?

```java
for (int j = s.length(); j > 0; j--)
    System.out.print(s.charAt(j-1));
```

A) it prints s out backwards
B) it prints s out forwards
C) it prints s out backwards after skipping the last character
D) it prints s out backwards but does not print the 0th character
E) it yields a run-time error because there is no character at s.charAt(j-1) for j = 0
Quiz

Given that s is a String, what does the following loop do?

```java
for (int j = s.length(); j > 0; j--)
    System.out.print(s.charAt(j-1));
```

A) it prints s out backwards
B) it prints s out forwards
C) it prints s out backwards after skipping the last character
D) it prints s out backwards but does not print the 0th character
E) it yields a run-time error because there is no character at s.charAt(j-1) for j = 0
Quick Check

Write a code fragment that rolls a die 100 times and counts the number of times a 3 comes up. Use the for loop!
Use the roll() methods of the Die class (it returns the new int faceValue of the Die object on which is called).
Quick Check

Write a code fragment that rolls a die 100 times and counts the number of times a 3 comes up. Use the \texttt{for} loop!

\begin{verbatim}
Die die = new Die();
int count = 0;
for (int num=1; num <= 100; num++)
    if (die.roll() == 3)
        count++;
System.out.println (count);
\end{verbatim}
The for Statement

• Each **expression** in the header of a **for** loop is **optional**

• If the **initialization** is left out, no initialization is performed

• If the **condition** is left out, it is always considered to be true, and therefore creates an infinite loop

• If the **increment** is left out, no increment operation is performed
For-each Loops

• A variant of the for loop simplifies the repetitive processing of items in an iterator

• For example, suppose bookList is an ArrayList<Book> object

• The following loop will print each book:

```java
for (Book myBook : bookList)
    System.out.println (myBook);
```

• This version of a for loop is often called a for-each loop
For-each Loops

• A for-each loop can be used on any object that implements the `Iterable` interface

• It eliminates the need to retrieve an `Iterator` (from the `Iterable`) and call the `hasNext` and `next` methods explicitly

• It also will be helpful when processing arrays, which are discussed in Chapter 8
Quick Check

Write a for-each loop that prints all of the `Student` objects in an `ArrayList<Student>` object called `roster`. 
Quick Check

Write a for-each loop that prints all of the Student objects in an ArrayList<Student> object called roster.

```java
for (Student student : roster)
    System.out.println (student);
```
Quiz

• Transform the following `while` loop in a program with the same semantic but using `for`

```java
while (x > 2) {
    system.out.println(x);
    x++;
}
```
Quiz

• Transform the following while loop in a program with the same semantic but using for

```java
while (x > 2) {
    system.out.println(x);
    x++;
}
```

```java
for(;x > 2; x++) {
    system.out.println(x);
}
```
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
Drawing Techniques

• Conditionals and loops enhance our ability to generate interesting graphics

• See Bullseye.java
• See BullseyePanel.java

• See Boxes.java
• See BoxesPanel.java
import javax.swing.JFrame;

public class Bullseye
{
    public static void main (String[] args)
    {
        JFrame frame = new JFrame("Bullseye");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        BullseyePanel panel = new BullseyePanel();

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JFrame;

public class Bullseye {
    //------
    // Creates
    //------
    public static void main (String[] args) {
        JFrame frame = new JFrame("Bullseye");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        BullseyePanel panel = new BullseyePanel();
        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JPanel;
import java.awt.*;

public class BullseyePanel extends JPanel
{
    private final int MAX_WIDTH = 300, NUM_RINGS = 5, RING_WIDTH = 25;

    // Sets up the bullseye panel.
    public BullseyePanel ()
    {
        setBackground (Color.cyan);
        setPreferredSize (new Dimension(300,300));
    }

    continue
// Paints a bullseye target.

public void paintComponent (Graphics page)
{
    super.paintComponent (page);
    int x = 0, y = 0, diameter = MAX_WIDTH;
    page.setColor (Color.white);

    for (int count = 0; count < NUM_RINGS; count++)
    {
        if (page.getColor() == Color.black)  // alternate colors
            page.setColor (Color.white);
        else
            page.setColor (Color.black);

        page.fillOval (x, y, diameter, diameter);
        diameter -= (2 * RING_WIDTH);
        x += RING_WIDTH;
        y += RING_WIDTH;
    }

    // Draw the red bullseye in the center
    page.setColor (Color.red);
    page.fillOval (x, y, diameter, diameter);
}
import javax.swing.JFrame;

public class Boxes {
    public static void main (String[] args) {
        JFrame frame = new JFrame ("Boxes");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        BoxesPanel panel = new BoxesPanel();
        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JFrame;

public class Boxes {

    // Creates the main frame of the program.
    public static void main(String[] args) {
        JFrame frame = new JFrame("Boxes");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        BoxesPanel panel = new BoxesPanel();
        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JPanel;
import java.awt.*;
import java.util.Random;

public class BoxesPanel extends JPanel {

    private final int NUM_BOXES = 50, THICKNESS = 5, MAX_SIDE = 50;
    private final int MAX_X = 350, MAX_Y = 250;
    private Random generator;

    //----------------------------------------------------------------------
    // Sets up the drawing panel.
    //----------------------------------------------------------------------
    public BoxesPanel () {
        generator = new Random();

        setBackground (Color.black);
        setPreferredSize (new Dimension(400, 300));
    }

    continue

// Paints boxes of random width and height in a random location.  
// Narrow or short boxes are highlighted with a fill color.  

public void paintComponent(Graphics page) {
    super.paintComponent(page);

    int x, y, width, height;

    for (int count = 0; count < NUM_BOXES; count++) {
        x = generator.nextInt(MAX_X) + 1;
        y = generator.nextInt(MAX_Y) + 1;

        width = generator.nextInt(MAX_SIDE) + 1;
        height = generator.nextInt(MAX_SIDE) + 1;

        continue
    }
}
continue

if (width <= THICKNESS)  // check for narrow box
{
    page.setColor (Color.yellow);
    page.fillRect (x, y, width, height);
}
else
    if (height <= THICKNESS)  // check for short box
    {
        page.setColor (Color.green);
        page.fillRect (x, y, width, height);
    }
else
    {
        page.setColor (Color.white);
        page.drawRect (x, y, width, height);
    }
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
Dialog Boxes

• A dialog box is a window that appears on top of any currently active window

• It may be used to:
  – convey information
  – confirm an action
  – allow the user to enter data
  – pick a color
  – choose a file

• A dialog box usually has a specific, solitary purpose, and the user interaction with it is brief
Dialog Boxes

• The `JOptionPane` class provides methods that simplify the creation of some types of dialog boxes

• See `EvenOdd.java`

• Specialized dialog boxes for choosing colors and files are covered in Chapter 9
/*********************************************************************/  
// EvenOdd.java Author: Lewis/Loftus
//
// Demonstrates the use of the JOptionPane class.
//*********************************************************************/

import javax.swing.JOptionPane;

public class EvenOdd
{
    //-----------------------------------------------------------------
    // Determines if the value input by the user is even or odd.
    // Uses multiple dialog boxes for user interaction.
    //-----------------------------------------------------------------
    public static void main (String[] args)
    {
        String numStr, result;
        int num, again;

        continue
continue

do
{
    numStr = JOptionPane.showInputDialog("Enter an integer: ");
    num = Integer.parseInt(numStr);

    result = "That number is " + ((num%2 == 0) ? "even" : "odd");

    JOptionPane.showMessageDialog(null, result);
    again = JOptionPane.showConfirmDialog(null, "Do Another?");
} while (again == JOptionPane.YES_OPTION);
}
continue

do{
    numStr = JOptionPane.showInputDialog("Enter an integer: ");
    num = Integer.parseInt(numStr);
    result = "That number is " + ((num%2 == 0) ? "even" : "odd");
    JOptionPane.showMessageDialog(null, result);
    again = JOptionPane.showConfirmDialog(null, "Do Another?");
}while(again == JOptionPane.YES_OPTION);
Summary

• Chapter 6 focused on:
  – the switch statement
  – the conditional operator
  – the do loop
  – the for loop
  – drawing with the aid of conditionals and loops
  – dialog boxes
Exercise

• What output is produced by the following code fragment?

```java
for (int val = 200; val >= 0; val -= 1)
    if (val % 4 != 0)
        System.out.println(val);
```
Solution

• What output is produced by the following code fragment?

```java
for (int val = 200; val >= 0; val -= 1)
    if (val % 4 != 0)
        System.out.println(val);
```

• The output produced is all values from 200 down to 1, except those that are evenly divisible by 4:

  • 199
  • 198
  • 197
  • 195
  • and so on until...
  • 5
  • 3
  • 2
  • 1
Exercise

• When the following loop stops?

```java
for (double x = 0.0; x != 10.0 && x < 11; x += 0.1)
    System.out.println(x);
```
Exercise

• In mathematics, the $n$-th harmonic number is the sum of the reciprocals of the first $n$ natural numbers:

$$ H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \ldots + \frac{1}{n} $$

• Implement a method (of the class Harmonic) that computes the $n$-th harmonic number ($n$ is the int parameter of the method)
public class Harmonic {

    public double harmonic(int N) {
        double sum = 0.0;
        for (int i = 1; i <= N; i++)
            sum += 1.0 / i;
        return sum;
    }

    public static void main(String[] args) {
        Harmonic h = new Harmonic();
        for (int i = 1; i <= 20; ++i)
            System.out.println(h.harmonic(i));
    }
}
Exercise

• $x_1 = y$
• $x_{n+1} = 1/2*(x_n + y/x_n)$
• $\lim_{n \to \infty} x_n = \sqrt{y}$

• Using the information above write a (static) method that computes the square root of a double

• Hint: implement a loop where a variable $x$ is updated with the formula above. The loop is terminated when $x*x$ is enough close to $y$
public static double sqrt(double y) {
    if (y < 0)
        return Double.NaN;
    double err = 1e-15;
    double x = y;
    while (Math.abs(x*x - y) > err)
        x = (y / x + x) / 2.0;
    return x;
}
Exercise

• Implement a simple primality test that, given an integer, returns true if this number is larger than 1 and is not divisible by any number smaller than itself (and false otherwise).
Exercise

```java
public static boolean isPrime(int N) {
    if (N < 2)
        return false;
    for (int i = 2; i * i <= N; i++)
        if (N % i == 0)
            return false;
    return true;
}
```
Exercise

• Write a method that convert a positive integer \( n \) into a string containing the binary representation of the integer \( n \)

• The method has an int parameter and return a String

Convert \( 14_{10} \) to Base 2

\[
\begin{array}{c|c}
2 & 14 \\
2 & 7 \quad \text{with remainder 0} \\
2 & 3 \quad \text{with remainder 1} \\
2 & 1 \quad \text{with remainder 1} \\
0 & \quad \text{with remainder 1} \\
\end{array}
\]

→ Least significant digit

Read remainders in reverse order to obtain

\[ 14_{10} = 1110_2 \]
Solution

class Integer2BinaryString {

    public static String convert(int n) {
        String s = "";
        for (int m = n; m > 0; m /= 2) {
            s = m%2 + s;
            return s;
        }
    }

    public static void main (String[] args) {
        System.out.println(convert(10));
    }
}