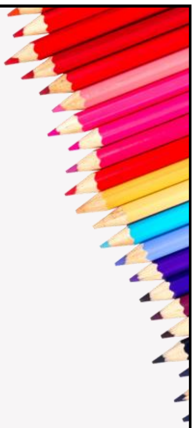
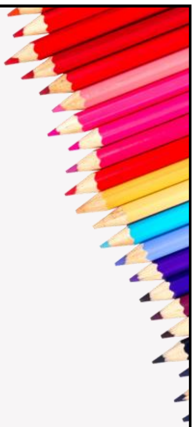
Chapter 3
Selections



Motivations

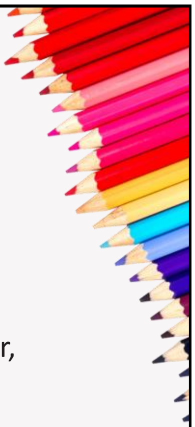
- If you assigned a negative value for radius in Listing 2.2, ComputeAreaWithConsoleInput.java, the program would print an invalid result.
- If the radius is negative, you don't want the program to compute the area. How can you deal with this situation?
- *The program can decide which statements to execute based on a condition.*
- Java provides ***selection statements****:* statements that let you choose actions with alternative courses.
- Selection statements use conditions that are ***Boolean expressions***.



Objectives (1)

- To declare boolean variables and write ***Boolean expressions*** using relational operators (§3.2).
- To implement selection control using ***one-way if*** statements (§3.3).
- To implement selection control using ***two-way if-else*** statements (§3.4).
- To implement selection control using ***nested if and multi-way if*** statements (§3.5).
- To avoid common errors and pitfalls in if statements (§3.6).
- To generate random numbers using the *Math.random()* method (§3.7).

3



Objectives (2)

- To program using selection statements for a variety of examples (SubtractionQuiz, BMI, ComputeTax) (§§3.7–3.9).
- To combine conditions using ***logical operators*** (&&, ||, and !) (§3.10).
- To program using selection statements with combined conditions (LeapYear, Lottery) (§§3.11–3.12).
- To implement selection control using ***switch*** statements (§3.13).
- To write expressions using the ***conditional operator*** (§3.14).
- To examine the rules governing operator ***precedence and associativity*** (§3.15).
- To apply common techniques to ***debug*** errors(§3.16).

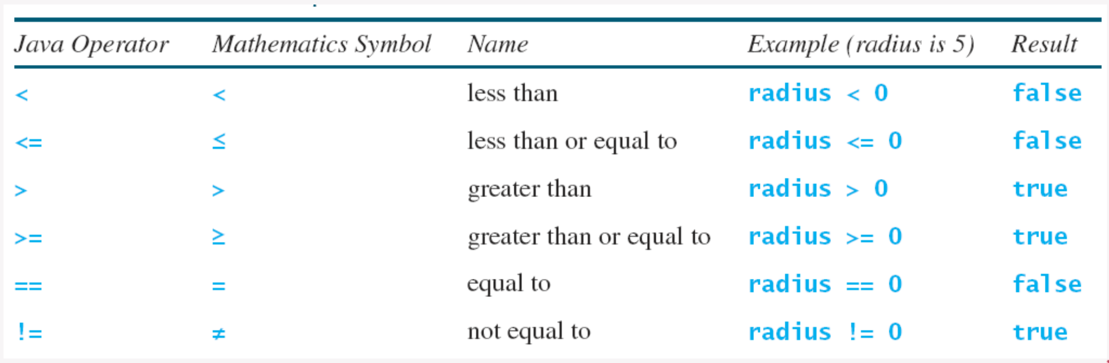
4

3.2 boolean Data Type, Values, and Expressions

- *The **boolean** data type declares a variable with the value either **true** or **false**.*
- Often in a program you need to compare two values, such as whether i is greater than j. Java provides six comparison operators (also known as relational operators) that can be used to compare two values.
- The result of the comparison is a Boolean value: ***true*** *or* ***false***.

  boolean b = (1 > 2);

Relational Operators

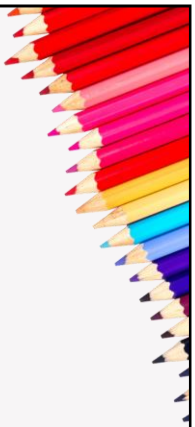


Java Operator	*Mathematics Symbol*	*Name*	*Example (radius is 5)*	*Result*
`<`	$<$	less than	`radius < 0`	`false`
`<=`	$\leq$	less than or equal to	`radius <= 0`	`false`
`>`	$>$	greater than	`radius > 0`	`true`
`>=`	$\geq$	greater than or equal to	`radius >= 0`	`true`
`==`	$=$	equal to	`radius == 0`	`false`
`!=`	$\neq$	not equal to	`radius != 0`	`true`

Problem: A Simple Math Learning Tool

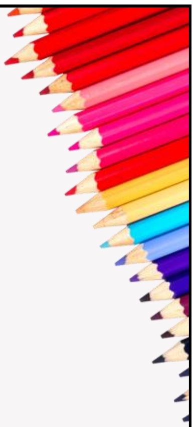
- This example creates a program to let a first grader practice additions.
  - The program randomly generates two single-digit integers number1 and number2 and displays a question such as “What is 7 + 9?” to the student.
  - After the student types the answer, the program displays a message to indicate whether the answer is true or false.

AdditionQuiz Run



LISTING 3.1 AdditionQuiz.java

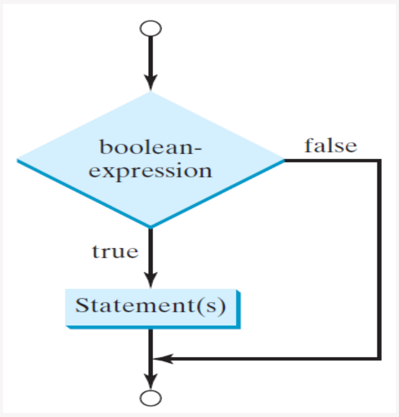
```java
import java.util.Scanner;

public class AdditionQuiz {
  public static void main(String[] args) {
    int number1 = (int)(System.currentTimeMillis() % 10);
    int number2 = (int)(System.currentTimeMillis() / 7 % 10);

    // Create a Scanner
    Scanner input = new Scanner(System.in);

    System.out.print(
      "What is " + number1 + " + " + number2 + "? ");

    int number = input.nextInt();

    System.out.println(
      number1 + " + " + number2 + " = " + answer + " is " +
      (number1 + number2 == answer));
  }
}
```



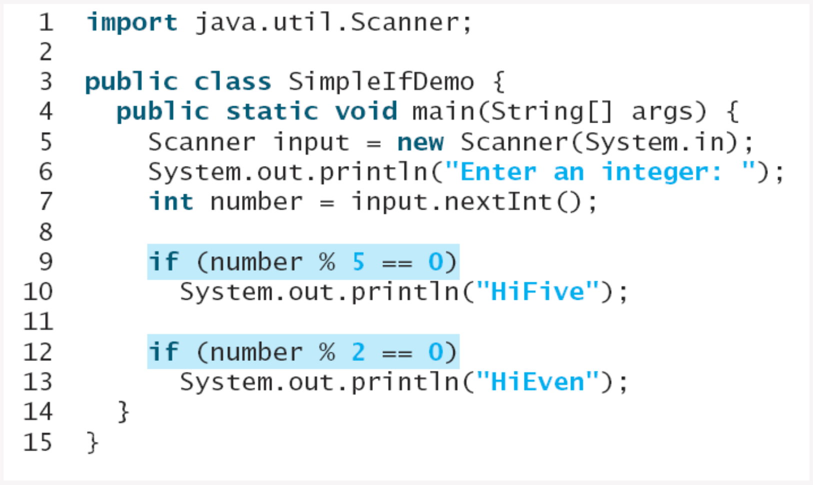
3.3 if Statements

- *An* ***if statement*** *is a construct that enables a program to specify alternative paths of execution.*
- One-way **`if`** Statements

```
if (boolean-expression) {
  statement(s);
}
```



Simple if Demo



```
import java.util.Scanner;

public class SimpleIfDemo {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("Enter an integer: ");
    int number = input.nextInt();

    if (number % 5 == 0)
      System.out.println("HiFive");

    if (number % 2 == 0)
      System.out.println("HiEven");
  }
}
```

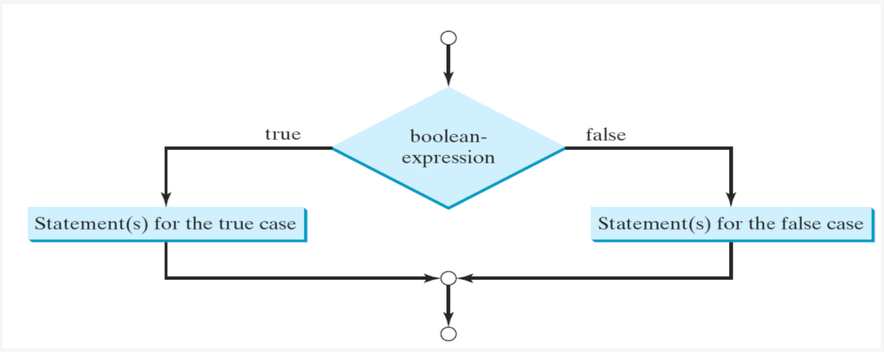
SimpleIfDemo

Run

3.4 Two-Way if-else Statements

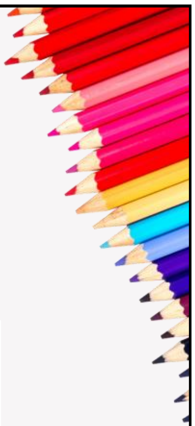
- *An* ***if-else*** *statement decides the execution path based on whether the condition is* ***true*** *or* ***false****.*

```
if (boolean-expression) {
  statement(s)-for-the-true-case;
}
else {
  statement(s)-for-the-false-case;
}
```


true
boolean-expression
false
Statement(s) for the true case
Statement(s) for the false case

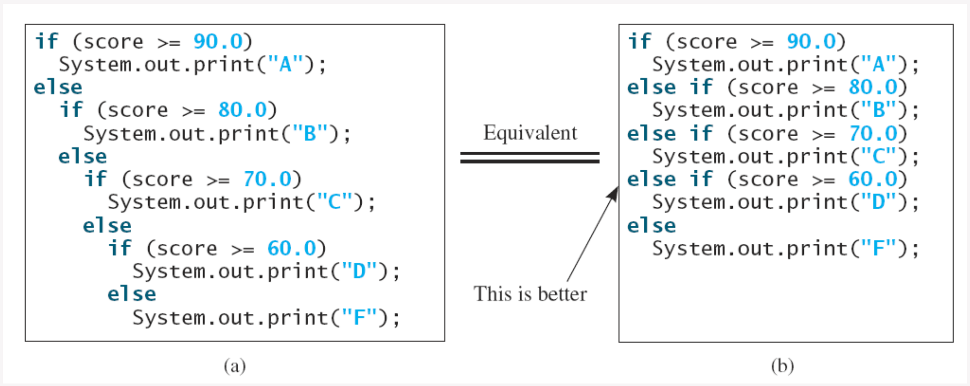
if-else Example

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("The area for the circle of radius " +
    radius + " is " + area);
}
else {
  System.out.println("Negative input");
}
```



3.5 Nested if and Multi-Way if-else Statements

- An if statement can be inside another if statement to form a nested if statement.
- The nested if statement can be used to implement multiple alternatives.



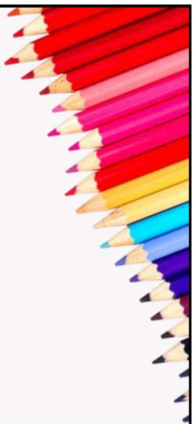
```
if (score >= 90.0)
  System.out.print("A");
else
  if (score >= 80.0)
    System.out.print("B");
  else
    if (score >= 70.0)
      System.out.print("C");
    else
      if (score >= 60.0)
        System.out.print("D");
      else
        System.out.print("F");
```

Equivalent

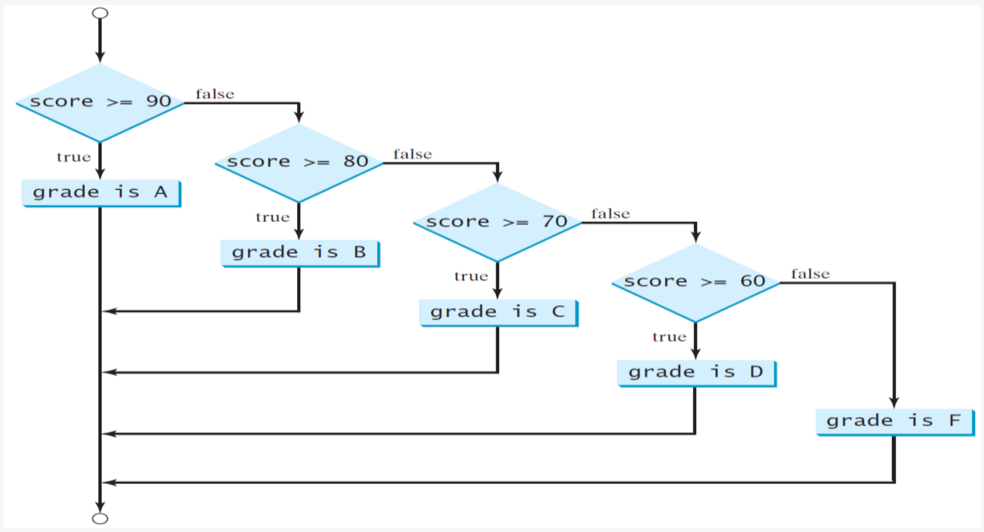
This is better

```
if (score >= 90.0)
  System.out.print("A");
else if (score >= 80.0)
  System.out.print("B");
else if (score >= 70.0)
  System.out.print("C");
else if (score >= 60.0)
  System.out.print("D");
else
  System.out.print("F");
```

(a) (b)



Multi-Way if-else Statements



14

3.6 Common Errors and Pitfalls

- **Common Error 1: Forgetting Necessary Braces**
- **Common Error 2: Wrong Semicolon at the if Line**
- **Common Error 3: Redundant Testing of Boolean Values**
- **Common Error 4: Dangling else Ambiguity**
- **Common Error 5: Equality Test of Two Floating-Point Values**
- **Common Pitfall 1: Simplifying Boolean Variable Assignment**
- **Common Pitfall 2: Avoiding Duplicate Code in Different Cases**

3.7 Generating Random Numbers

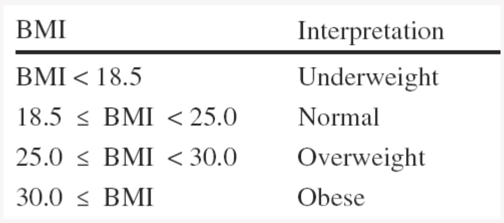
- *You can use **Math.random()** to obtain a random double value between 0.0 and 1.0, excluding 1.0.*
- This example creates a program to teach a first grade child how to learn subtractions.
  - The program randomly generates two single-digit integers number1 and number2 with number1 >= number2 and displays a question such as “What is 9 – 2?” to the student.
  - After the student types the answer, the program displays whether the answer is correct.

SubtractionQuiz   Run

16

Case Study: Body Mass Index

- Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters.
- The interpretation of BMI for people 16 years or older is as follows:



BMI	Interpretation
BMI < 18.5	Underweight
18.5 ≤ BMI < 25.0	Normal
25.0 ≤ BMI < 30.0	Overweight
30.0 ≤ BMI	Obese

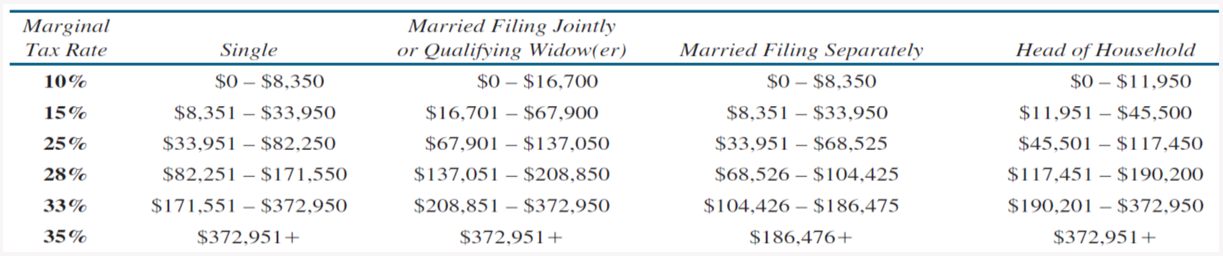
ComputeAndInterpretBMI

Run

17

Case Study: Computing Taxes

- The US federal personal income tax is calculated based on the filing status and taxable income. There are four filing statuses: single filers, married filing jointly, married filing separately, and head of household. The tax rates for 2009 are shown below.



Marginal Tax Rate	*Single*	*Married Filing Jointly or Qualifying Widow(er)*	*Married Filing Separately*	*Head of Household*
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351 – $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 – $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

18

Case Study: Computing Taxes, cont.

```
if (status == 0) {
  // Compute tax for single filers
}
else if (status == 1) {
  // Compute tax for married filing jointly or qualifying widow(er)
}
else if (status == 2) {
  // Compute tax for married filing separately
}
else if (status == 3) {
  // Compute tax for head of household
}
else {
  // Display wrong status
}
```

ComputeTax

Run

3.10 Logical Operators

- *The logical operators* ***!, &&, ||,*** *and* ***^*** *can be used to create a compound Boolean expression.*

Operator	*Name*	*Description*
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction
^	exclusive or	logical exclusion

Examples

- Here is a program that checks whether a number is divisible by 2 and 3, whether a number is divisible by 2 or 3, and whether a number is divisible by 2 or 3 but not both:

```
if (number % 2 == 0 && number % 3 == 0)
  System.out.println(number + " is divisible by 2 and 3.");

if (number % 2 == 0 || number % 3 == 0)
  System.out.println(number + " is divisible by 2 or 3.");

if (number % 2 == 0 ^ number % 3 == 0)
  System.out.println(number +
    " is divisible by 2 or 3, but not both.");
```

TestBooleanOperators

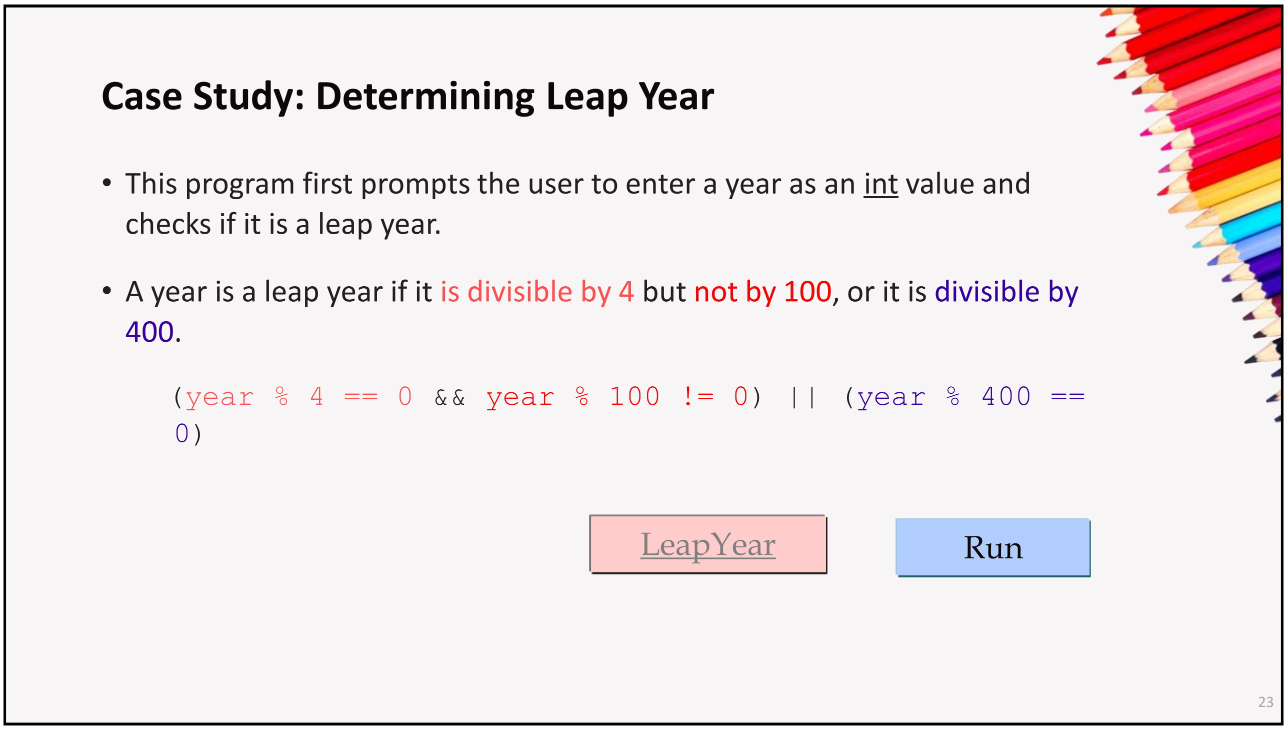
Run

The & and | Operators

Companion Website

- Supplement III.B, “The & and | Operators”

```
If x is 1, what is x after this expression?
(x > 1) & (x++ < 10)

If x is 1, what is x after this expression?
(1 > x) && ( 1 > x++)

How about (1 == x) | (10 > x++)?
(1 == x) || (10 > x++)?
```

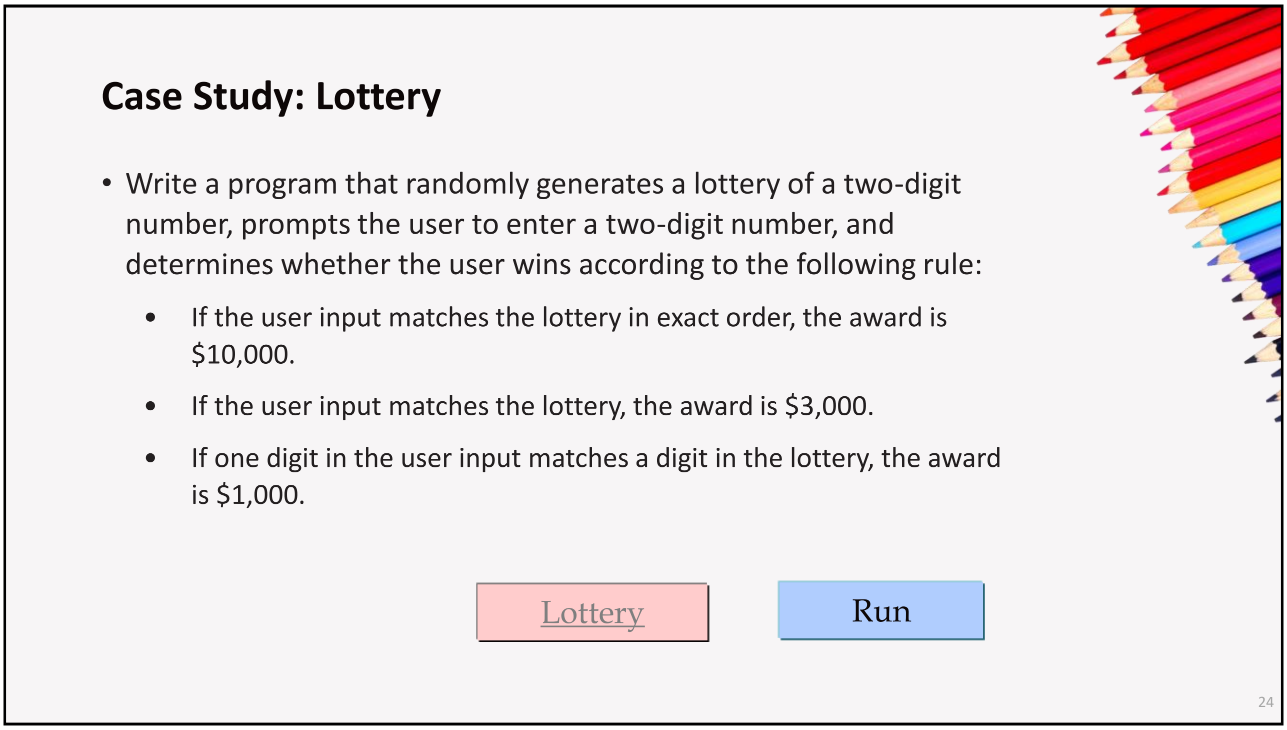


Case Study: Determining Leap Year

- This program first prompts the user to enter a year as an int value and checks if it is a leap year.
- A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.

```
(year % 4 == 0 && year % 100 != 0) || (year % 400 ==
0)
```

LeapYear Run



Case Study: Lottery

- Write a program that randomly generates a lottery of a two-digit number, prompts the user to enter a two-digit number, and determines whether the user wins according to the following rule:
  - If the user input matches the lottery in exact order, the award is $10,000.
  - If the user input matches the lottery, the award is $3,000.
  - If one digit in the user input matches a digit in the lottery, the award is $1,000.

Lottery Run

3.13 switch Statements

- *A **switch statement** executes statements based on the value of a variable or an expression.*

```
switch (switch-expression) {
  case value1: statement(s)1;
               break;
  case value2: statement(s)2;
               break;
  ...
  case valueN: statement(s)N;
               break;
  default:     statement(s)-for-default;
}
```

25

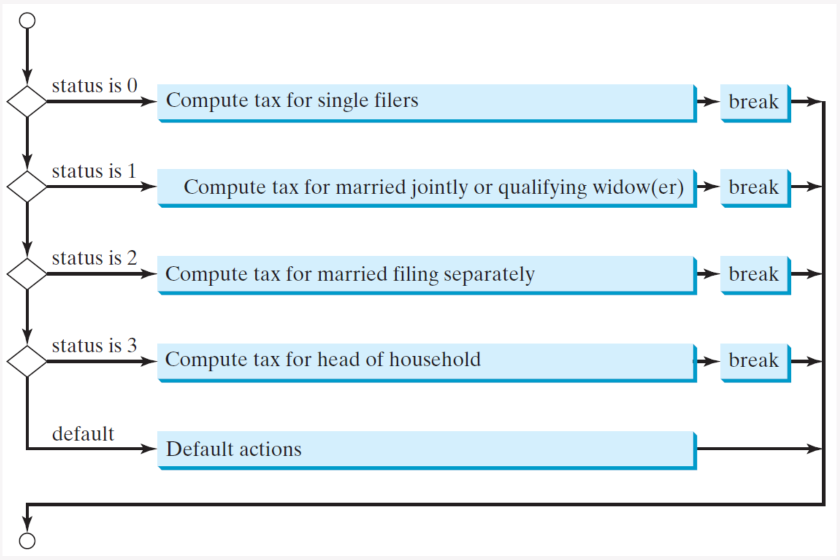
switch Statements

- You can write the following switch statement to replace the nested if statement in **Listing 3.5**:

```
switch (status) {
  case 0:  compute tax for single filers;
           break;
  case 1:  compute tax for married jointly or qualifying widow(er);
           break;
  case 2:  compute tax for married filing separately;
           break;
  case 3:  compute tax for head of household;
           break;
  default: System.out.println("Error: invalid status");
           System.exit(1);
}
```

26

switch Statement Flow Chart


status is 0
Compute tax for single filers
break
status is 1
Compute tax for married jointly or qualifying widow(er)
break
status is 2
Compute tax for married filing separately
break
status is 3
Compute tax for head of household
break
default
Default actions

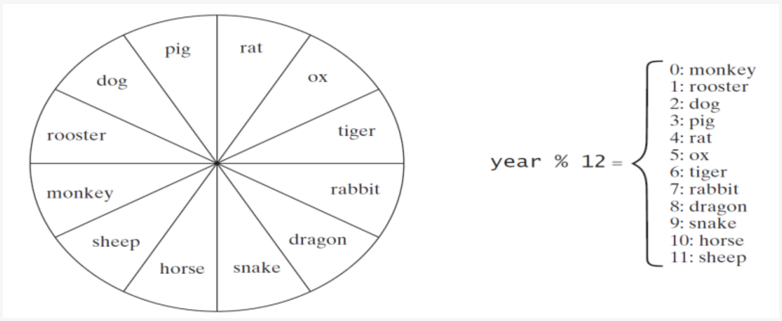
27

switch Statement Rules

- The switch-expression must yield a value of **char, byte, short, int**, or **String** type and must always be enclosed in parentheses.
- The value1, . . ., and valueN are **constant expressions**, they must have the same data type as the value of the switch expression.
- When the value in a **case** statement matches the value of the switch-expression, the statements starting from this case are executed until either a break statement or the end of the switch statement is reached.
- The **default** case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.
- The keyword **break** is optional. The break statement immediately ends the switch statement.

Problem: Chinese Zodiac

- Write a program that prompts the user to enter a year and displays the animal for the year.



ChineseZodiac   Run

3.14 Conditional Expressions

- *A conditional expression evaluates an expression based on a condition.*
- Conditional Operator

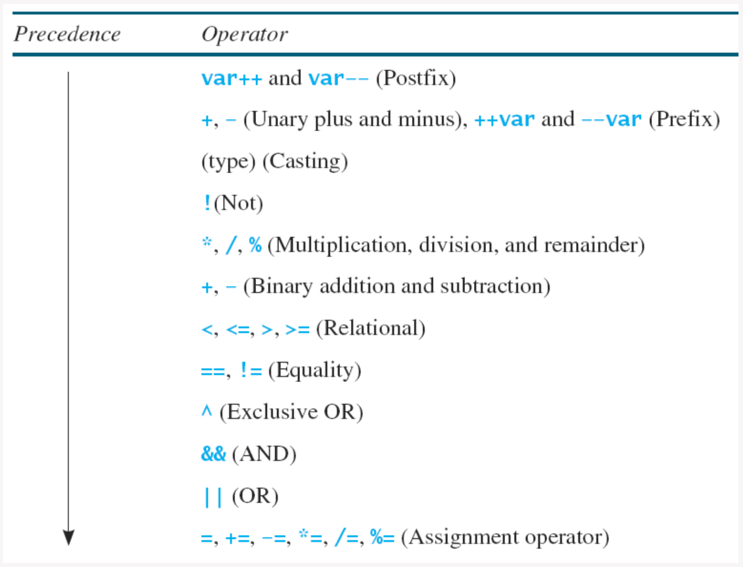
```
(boolean-expression) ? expression1 : expression2
```

```
if (x > 0)
  y = 1;
else
  y = -1;


y = (x > 0) ? 1 : -1;
```

```
if (num % 2 == 0)
  System.out.println(num + "is even");
else
  System.out.println(num + "is odd");


System.out.println(
  (num % 2 == 0)? num + "is even" :
  num + "is odd");
```

3.15 Operator Precedence and Associativity

- Operator precedence and associativity determine the order in which operators are evaluated.



Precedence	*Operator*
↓	var++ and var-- (Postfix)
	+, - (Unary plus and minus), ++var and --var (Prefix)
	(type) (Casting)
	!(Not)
	*, /, % (Multiplication, division, and remainder)
	+, - (Binary addition and subtraction)
	<, <=, >, >= (Relational)
	==, != (Equality)
	^ (Exclusive OR)
	&& (AND)
	|| (OR)
	=, +=, -=, *=, /=, %= (Assignment operator)

Operator Precedence and Associativity

- The expression in the parentheses is evaluated first.
  - Parentheses can be nested, in which case the expression in the inner parentheses is executed first.
- When evaluating an expression without parentheses, the operators are applied according to the precedence rule and the associativity rule.
- If operators with the same precedence are next to each other, their associativity determines the order of evaluation.

Operator Associativity

- When two operators with the same precedence are evaluated, the associativity of the operators determines the order of evaluation.
- All binary operators except assignment operators are left-associative.

  a – b + c – d is equivalent to ((a – b) + c) – d
- Assignment operators are right-associative. Therefore, the expression

  a = b += c = 5 is equivalent to a = (b += (c = 5))

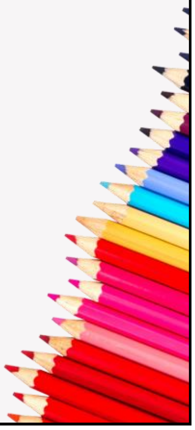
33

3.16 Debugging

- *Debugging is the process of finding and fixing errors in a program.*
- Logic errors are called *bugs*. The process of finding and correcting errors is called *debugging*.
- A common approach to debugging is to use a combination of methods to help pinpoint the part of the program where the bug is located.
  - *hand-trace* the program (i.e., catch errors by reading the program)
  - insert print statements in order to show the values of the variables or the execution flow of the program.
  - use a debugger utility
- JDK includes a command-line debugger, jdb, which is invoked with a class name.

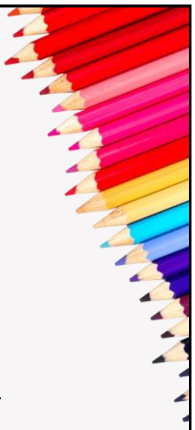
Debuggers

- All the Java IDE tools, such as Eclipse and NetBeans, include integrated debuggers.
- The features of debugger utilities:
  - Executing a single statement at a time
  - Tracing into or stepping over a method
  - Setting breakpoints
  - Displaying variables
  - Displaying call stacks
  - Modifying variables



Chapter Summary

Chapter Summary



- A boolean type variable can store a true or false value.
- The ***relational operators*** (<, <=, ==, !=, >, >=) yield a Boolean value.
- The Boolean operators **&&, ||, !, and ^** operate with Boolean values and variables.
- Selection statements are used for programming with alternative courses of actions. There are several types of selection statements:
  - one-way if statements, two-way if-else statements, nested if statements, multi-way if-else statements, switch statements, and conditional expressions.
- The operators in expressions are evaluated in the order determined by the rules of parentheses, *operator precedence, and operator associativity.*

Programming Exercises

1, 4, 7, 9, 11,

12, 15, 16, 17, 19,

21, 22, 24, 30, 32

38