Programming Fundamentals I
Java Fundamentals

Nguyen Ngoc Minh

Saigon Institute of Technology
Readings

• Readings for these lecture notes
  • Textbook:
    • Tony Gaddis & Godfrey, *Starting Out with Java From Control Structures through Data Structures*
  • Reference:
    • Cay S. Horstmann and Gary Cornell, *Core Java™*

• These lecture notes contain material © Tony Gaddis, and Godfrey Muganda, 2007
Chapter Objectives

Upon completion of this chapter, you should be able to:

• Write boolean expressions

• Write conditional statements using: `if`, `if-else`, `if-else-if`, `nested if` statements

• Compare String objects.

• Use conditional operator.

• Format numbers using DecimalFormat class.

• Concept of algorithm decomposition.
Outline

• Boolean expression & Relational Operators
• The if Statement
• Comparing Characters
• The if-else Statement
• The if-else-if Statement
• Nested if Statements
• Logical Operators
• Comparing String Objects
• Variable Declaration and Scope
• The Conditional Operator
• The switch Statement
A boolean expression is any variable or calculation that results in a true or false condition.

In most cases, the boolean expression uses relational operators.

<table>
<thead>
<tr>
<th>Relational Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>is greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>is less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>is greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>is less than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>is equal to</td>
</tr>
<tr>
<td>!=</td>
<td>is not equal to</td>
</tr>
</tbody>
</table>
Boolean expression & Relational Operators

- Example:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 &gt; 3</td>
<td>true</td>
</tr>
<tr>
<td>5 &lt; 3</td>
<td>false</td>
</tr>
<tr>
<td>4 &gt;= 3</td>
<td>true</td>
</tr>
<tr>
<td>4 &lt;= 4</td>
<td>true</td>
</tr>
<tr>
<td>-1 == -1</td>
<td>true</td>
</tr>
<tr>
<td>5 != 5</td>
<td>false</td>
</tr>
</tbody>
</table>
Outline

• Boolean expression & Relational Operators
  • The if Statement
• The if-else Statement
• Comparing Characters
• The if-else-if Statement
• Nested if Statements
• Logical Operators
• Comparing String Objects
• Variable Declaration and Scope
• The Conditional Operator
• The switch Statement
The if Statement

- The code in methods executes *sequentially*.
- The if statement uses a boolean expression as an argument to decide if the next statement or block of statements executes.

\[
\text{if(boolean expression)} \\
\text{statement;}
\]

- Whether or not the statement is executed depends on the result of boolean expression
The if Statement
(Example)

What will be the output?

```java
public class IfStatement {
    public static void main(String[] args) {
        boolean condition;

        condition = (5 > 4);
        if (condition)
            System.out.println("5 is greater than 4");

        condition = (5 <= 4);
        if (condition)
            System.out.println("5 is less than or equal 4");
    }
}
```
The if Statement
(Con’t)

• A block if statement may be modeled as:

```
if(boolean expression){
    statement_1;
    statement_2;
    ........
    statement_n;
}
```
The if Statement

Example

- These programs print the same output?

```java
public class IfStatement {
    public static void main(String[] args) {
        boolean condition;
        condition = (5 > 4);
        if (condition) {
            System.out.println("5 is greater than 4");
            System.out.println(condition);
        }
    }
}
```

```java
public class IfStatement {
    public static void main(String[] args) {
        boolean condition;
        condition = (5 > 4);
        if (condition) {
            System.out.println("5 is greater than 4");
            System.out.println(condition);
        }
    }
}
```
The if Statement

Flowcharts

- If statements can be modeled as a flow chart.

```java
if (coldOutside) wearCoat();
```
The if Statement

Flowcharts

- Use a block to group many statements inside the if statement.

```java
if (coldOutside){
    wearCoat();
    wearHat();
    wearGloves();
}
```

Note the use of curly braces to block several statements together.
Exercise 1

- Write a program ask the user to enter two number and show a message to notice the relationship between them.
  Ex: X is greater than Y
  or X is smaller than Y
  or X equal Y

Hint: get file `Comparison.java` and upgrade.
Exercise 2

- Write a program to ask the user to enter an integer number. Then show a message to say if that number is odd or even.
The if Statement
(Con’t)

• If statements can span more than one line; however, they are still one statement.

    if(average > 95)
           System.out.println("That’s a great score!");

• Is functionally equivalent to

    if(average > 95) System.out.println("That’s a great score!");
The if Statement
(Con’t)

Rules of thumb:

• The conditionally executed statement should be on the line after the if condition.
• The conditionally executed statement should be indented one level from the if condition.
• If an if statement does not have the block curly braces, it is ended by the first semicolon encountered after the if condition.

```java
if(expression)  // No semicolon here.
    statement;   // Semicolon ends statement here.
```
Outline

• Boolean expression
• Relational Operators
• The if Statement
• **Comparing Characters**
• The if-else Statement
• The if-else-if Statement
• Nested if Statements
• Logical Operators
• Comparing String Objects
• Variable Declaration and Scope
• The Conditional Operator
• The switch Statement
Comparing Characters

- Characters can be tested using the relational operators.
- Characters are stored in the computer using the unicode character format.
- Unicode is stored as a sixteen (16) bit number.
- Characters are *ordinal*, meaning they have an order in the unicode character set.
- Since characters are ordinal, they can be compared to each other.

```java
char c = 'A';
if(c < 'Z')
    System.out.println("A is less than Z");
```
Outline

• Boolean expression
• Relational Operators
• The if Statement
• Comparing Characters
• The if-else Statement
• The if-else-if Statement
• Nested if Statements
• Logical Operators
• Comparing String Objects
• Variable Declaration and Scope
• The Conditional Operator
• The switch Statement
The if-else Statement

- The if-else statement adds the ability to conditionally execute code based if the expression of the if statement is false.

```java
if( boolean expression )
    statement Or Block of Statements;
else
    statement Or Block of Statements;
```
The if-else Statement

Example

```java
import javax.swing.JOptionPane;

public class EvenOdd {
    public static void main(String[] args) {
        int number; // the number which user inputed
        int remainder;
        String temp;
        temp = JOptionPane.showInputDialog("Please enter an integer number: ");
        number = Integer.parseInt(temp);
        remainder = number % 2;
        if (remainder == 0 )
            JOptionPane.showMessageDialog(null, "This is an even number");
        if (remainder != 0 )
            JOptionPane.showMessageDialog(null, "This is an odd number");
    }
}
```
The if-else Statement

Example

```java
import javax.swing.JOptionPane;

public class EvenOdd {
    public static void main(String[] args) {
        int number; // the number which user inputted
        int remainder;
        String temp;
        temp = JOptionPane.showInputDialog("Please enter an integer number: ");
        number = Integer.parseInt(temp);
        remainder = number % 2;
        if (remainder == 0)
            JOptionPane.showMessageDialog(null, "This is an even number");
        else
            JOptionPane.showMessageDialog(null, "This is an odd number");
    }
}
```
The if-else Statement

Flow Chart

Is it even number?

Yes

This is an even number

No

This is an odd number
Outline

• Boolean expression
• Relational Operators
• The if Statement
• Comparing Characters
• The if-else Statement
• The if-else-if Statement
• Nested if Statements
• Logical Operators
• Comparing String Objects
• Variable Declaration and Scope
• The Conditional Operator
• The switch Statement
The if-else-if Statement

```java
if ( boolean expression )
    statement Or Block of Statements;
else if ( boolean expression )
    statement Or Block of Statements;
// Put as many else ifs as needed here
else
    statement or block
```
The if-else-if Statement

Flow Chart
The if-else-if Statement

Example (Debug to demo)

```java
import javax.swing.JOptionPane;

public class TestResults {
    public static void main(String[] args) {
        int testScore; // Numeric test score
        String input;  // to hold the user's input

        input = JOptionPane.showInputDialog("Enter your numeric " +
            "test score and I will tell you the grade: ");

        testScore = Integer.parseInt(input);
    }
}
```
The if-else-if Statement

Example

```java
if (testScore < 60)
    JOptionPane.showMessageDialog(null, "Your grade is F.");
else if (testScore < 70)
    JOptionPane.showMessageDialog(null, "Your grade is D.");
else if (testScore < 80)
    JOptionPane.showMessageDialog(null, "Your grade is C.");
else if (testScore < 90)
    JOptionPane.showMessageDialog(null, "Your grade is B.");
else if (testScore <= 100)
    JOptionPane.showMessageDialog(null, "Your grade is A.");
else
    JOptionPane.showMessageDialog(null, "Invalid score.");
```
The if-else-if Statement

- if-else-if statements can become very complex.

- Care must be used since else statements match up with the immediately preceding unmatched if statement.

- Example: If_Else_If_False_Matching.java
The if-else-if Statement

Example (bad Programming Style)

1. What is the output when run this program?
2. Replace line 6 with if(false), what is the output?

```java
public class If_Else_If_False_Matching {
    public static void main(String[] args) {
        if(true) System.out.println("0");
        if(true) System.out.println("1");
        else if(true) System.out.println("2");
        else if(true) System.out.println("3");
        else if(true) System.out.println("4");
        else System.out.println("5");
    }
}
```
Outline

- Boolean expression
- Relational Operators
- The if Statement
- Comparing Characters
- The if-else Statement
- The if-else-if Statement
- Nested if Statements
- Logical Operators
- Comparing String Objects
- Variable Declaration and Scope
- The Conditional Operator
- The switch Statement
Nested if Statements

• If an if statement appears inside of another if statement (single or block) it is called a nested if statement.

• The nested if is only executed if the if statement it is in results in a true condition.

• Nested if statements can get very complex, very quickly.

• Example: LoanQualifier.java
Nested if Statements

Flow chart

Is it cold outside?

- Yes
  - Is it snowing?
    - Yes: Wear a parka.
    - No: Wear a jacket.
- No: Wear shorts.

AAS-IT Programming Fundamentals I 34/78
if-else Matching

- Curly brace use is not required if there is only one statement to be conditionally executed.
- However, sometimes curly braces can help make the program more readable.
- Additionally, proper indentation makes it much easier to match up else statements with their corresponding if statement.
if-else Matching

```java
if (employed == 'y')
{
    if (recentGrad == 'y')
    {
        System.out.println("You qualify for the special interest rate.");
    }
    else
    {
        System.out.println("You must be a recent college graduate to qualify.");
    }
}
else
{
    System.out.println("You must be employed to qualify.");
}
```
Outline

- Boolean expression
- Relational Operators
- The if Statement
- Comparing Characters
- The if-else Statement
- The if-else-if Statement
- Nested if Statements
- Logical Operators
- Comparing String Objects
- Variable Declaration and Scope
- The Conditional Operator
- The switch Statement
Logical Operators

- Two binary logical operators are used to connect relational expressions into one
  - And `&&`
  - Or `||`
- unary logical operator `!` to reverse the logic of a boolean expression.
## Logical Operators

### Con’t

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
<td>Connects two boolean expressions into one. Both expressions must be true for the overall expression to be true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>NOT</td>
<td>The ! operator reverses the truth of a boolean expression. If it is applied to an expression that is true, the operator returns false. If it is applied to an expression that is false, the operator returns true.</td>
</tr>
</tbody>
</table>
Logical Operators
The && Operator

• The logical AND operator (&&) takes two operands that must both be boolean expressions.
• The resulting combined expression is true iff both operands are true.

<table>
<thead>
<tr>
<th>Expression 1</th>
<th>Expression 2</th>
<th>Expression1 &amp;&amp; Expression2</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Logical Operators
The && Operator (Example)

- Problem:
  Write a program to determine whether a bank customer qualifies for a loan. To qualify, a customer must earn at least $30,000 per year, **And** must have been on his or her current job for at least two years.
Logical Operators

The && Operator (Example)

- Using nested if

```
No

Salary >= 30000

Don’t Qualify for loan

Yes

YearsOnJob >= 2

Don’t Qualify for loan

Qualify for loan
```
Logical Operators
The && Operator (Example)

- Using && operator to connect two boolean expressions

```
(YearsOnJob >= 2) &&
(YearsOnJob >= 2)
```

Flowchart:
- Yes: Qualify for loan
- No: Don’t Qualify for loan
The && Operator (Example with nested if)

```java
if (salary >= 30000){
    if (yearsOnJobs >=2){
        JOptionPane.showMessageDialog(null, "You qualify " +
        "for the loan");
    }
} else{
    JOptionPane.showMessageDialog(null, "You must have " +
        "been on your current job for at least " +
        "two years to qualify.");
}
else{
    JOptionPane.showMessageDialog(null, "You must earn " +
        "at least $30000 per year to qualify.");
}
```
Logical Operators
The && Operator (Example with && operator)

```java
if (salary >= 30000 && yearsOnJobs >= 2)
{
    JOptionPane.showMessageDialog(null, "You qualify " +
    "for the loan");
}
else{
    JOptionPane.showMessageDialog(null, "You don't qualify " +
    "for the loan");
}
```
Logical Operators

The || Operator

- The logical OR operator (||) takes two operands that must both be boolean expressions.
- The resulting combined expression is false iff (if and only if) both operands are false.

| Expression 1 | Expression 2 | Expression1 || Expression2 |
|--------------|--------------|-------------------|
| true         | false        | true              |
| false        | true         | true              |
| false        | false        | false             |
| true         | true         | true              |
Logical Operators
The || Operator (Example)

- Problem:
  Write a program to determine whether a bank customer qualifies for a loan. To qualify, a customer must earn at least $30,000 per year, Or must have been on his or her current job for at least two years.
Logical Operators
The || Operator (Example)

- Using nested if
Logical Operators
The || Operator (Example)

- Using || operator to connect two boolean expressions

```
(YearsOnJob >= 2) || (YearsOnJob >= 2)
```

- No
- Don't Qualify for loan
- Yes
- Qualify for loan
Logical Operators

The || Operator (Example)

```java
if (salary >= 30000) {
    JOptionPane.showMessageDialog(null, "You qualify " + 
    "for the loan");
} else {
    if (yearsOnJobs >= 2) {
        JOptionPane.showMessageDialog(null, "You qualify " + 
        "for the loan");
    } else
        JOptionPane.showMessageDialog(null, "You don't qualify " + 
        "for the loan");
}
```
Logical Operators

The || Operator (Example with || operator)

```java
if (salary >= 30000 || yearsOnJobs >= 2)
{
    JOptionPane.showMessageDialog(null, "You qualify " +
    "for the loan");
}
else{
    JOptionPane.showMessageDialog(null, "You don't qualify " +
    "for the loan");
}
```
**Logical Operators**

The ! Operator

- The ! operator performs a logical NOT operation.
- If an expression is true, !expression will be false and vice versa.
- Ex: !(5>4) → false

<table>
<thead>
<tr>
<th>Expression 1</th>
<th>!Expression1</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
### Logical Operators

#### Order of precedence

<table>
<thead>
<tr>
<th>Order of Precedence</th>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(unary negation) !</td>
<td>Unary negation, logical NOT</td>
</tr>
<tr>
<td>2</td>
<td>* / %</td>
<td>Multiplication, Division, Modulus</td>
</tr>
<tr>
<td>3</td>
<td>+ -</td>
<td>Addition, Subtraction</td>
</tr>
<tr>
<td>4</td>
<td>&lt; &gt; &lt;= &gt;=</td>
<td>Less-than, Greater-than, Less-than or equal to, Greater-than or equal to</td>
</tr>
<tr>
<td>5</td>
<td>== ! =</td>
<td>Is equal to, Is not equal to</td>
</tr>
<tr>
<td>6</td>
<td>&amp;&amp;</td>
<td>Logical AND</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>= + = -= *= /= % =</td>
<td>Assignment and combined assignment operators.</td>
</tr>
</tbody>
</table>
Logical Operators
Order of precedence

• If operators have the same order of precedence, they are evaluated from Left to Right
• Parenthesis can be used to force the precedence to be changed.

Example: a and b are int variables.

!a>b
!(a>b)
Logical Operators
Order of precedence (Example)

Example 1: \( a > b \land \land x \leq y \)
   step 1: evaluate \( > \)
   step 2: evaluate \( \leq \)
   step 3: evaluate \( \land \land \)

Example 2: \( a == b \land \land x \geq y \lor m \neq n \)
   step 1: evaluate \( \geq \)
   step 2: evaluate \( == \)
   step 3: evaluate \( \neq \)
   step 3: evaluate \( \land \land \)
Logical Operators
Order of precedence (example)

• What’s the output?

```java
public class temp {
    public static void main(String[] args) {
        if(false && false || true)
            System.out.println("true");
        else
            System.out.println("false");
    }
}
```
Logical Operators

Order of precedence (example)

```java
public class temp {
    public static void main(String[] args) {
        if (5%2*3>2 == 1 >= 0+1)
            System.out.println(1);
        else
            System.out.println(2);
    }
}
```
Logical Operators

Short Circuiting

- Logical && and logical || operations perform short-circuit evaluation of expressions.
- Logical && will evaluate to false as soon as it sees that one of its operands is a false expression.
- Logical || will evaluate to true as soon as it sees that one of its operands is a true expression.
Logical Operators
Short Circuiting (example)

1. What’s the output?
2. If change && by ||, what’s the output?

```java
public class ShortCircuiting {
    public static void main(String[] args) {
        if (5 > 4) && (5 / 0 == 0) )
            System.out.println("1");
            else
                System.out.println("2");
    }
}
```
Outline

- Boolean expression
- Relational Operators
- The if Statement
- Comparing Characters
- The if-else Statement
- The if-else-if Statement
- Nested if Statements
- Logical Operators
- Comparing String Objects
- Variable Declaration and Scope
- The Conditional Operator
- The switch Statement
Comparing String Objects

- In most cases, you cannot use the relational operators to compare two String objects.
- Reference variables contain the address of the object they represent.
- Unless the references point to the same object, the relational operators will not return true.
- We must use functions of String class to compare content of reference variables:
  - `equals()`, `compareTo()`, `equalsIgnoreCase`, `compareToIgnoreCase()`
Comparing String Objects

Example

```java
public class CompareString {
    public static void main(String[] args) {
        String name1 = new String("Mary");
        String name2 = new String("Mary");

        if (name1 == name2)
            System.out.println("Compare Using == operator: equal");
        else
            System.out.println("Compare Using == operator: not equal");

        if (name1.equals(name2))
            System.out.println("Compare Using function: equal");
        else
            System.out.println("Compare Using function: not equal");
    }
}```
Comparing String Objects

Example

```java
public class CompareString {

    public static void main(String[] args) {
        String name1 = new String("Mary");
        String name2 = new String("Marz");

        int i;

        i = name1.compareTo(name2);
        if (i<0)
            System.out.println(name1 + " is less than " + name2);
        else if (i==0)
            System.out.println(name1 + " equals " + name2);
        else
            System.out.println(name1 + " is greater than " + name2);
    }
}
```
Outline

• Boolean expression
• Relational Operators
• The if Statement
• Comparing Characters
• The if-else Statement
• The if-else-if Statement
• Nested if Statements
• Logical Operators
• Comparing String Objects
• **Variable Declaration and Scope**
• The Conditional Operator
• The switch Statement
Variable Declaration and Scope

- In Java, a local variable does not have to be declared at the beginning of the method.

- The scope of a local variable begins at the point it is declared and terminates at the end of the method.

- When a program enters a section of code where a variable has scope, that variable has *come into scope*, which means the variable is visible to the program.
Variable Declaration and Scope

Example

```java
public class ScopeVariable {

    /*
     * @param args
     */

    public static void main(String[] args) {
        int i = 1;

        if(true)
        {
            //int i = 1;
            i = i + 1;
        }

        System.out.println(i);
    }
}
```
Outline

- Boolean expression
- Relational Operators
- The if Statement
- Comparing Characters
- The if-else Statement
- The if-else-if Statement
- Nested if Statements
- Logical Operators
- Comparing String Objects
- Variable Declaration and Scope
- The Conditional Operator
- The switch Statement
The Conditional Operator

• The *conditional operator* is a ternary (three operand) operator.
• The conditional operator allows a programmer to write a simple if-else type statement.
• The format of the operators is:
  
  \[
  \text{expression1} \ ? \ \text{expression2} \ : \ \text{expression3}
  \]
• The conditional operator can also return a value.
The Conditional Operator
Con’t

• The conditional operator can be used as a shortened if-else statement:
  
  \[ x > y \ ? \ z = 10 \ : \ z = 5; \]

• This line is functionally equivalent to:
  
  ```
  if(x > )
      z = 10;
  else
      z = 5;
  ```
The Conditional Operator
Con’t

• Many times, the conditional operator is used to supply a value.
  
  ```
  number = x > y ? 10 : 5;
  ```

• This is functionally equivalent to:
  
  ```
  if(x > y)
      number = 10;
  else
      number = 5;
  ```
Outline

• Boolean expression
• Relational Operators
• The if Statement
• Comparing Characters
• The if-else Statement
• The if-else-if Statement
• Nested if Statements
• Logical Operators
• Comparing String Objects
• Variable Declaration and Scope
• The Conditional Operator
• The switch Statement
The switch Statement

• The if-else statements allow the programmer to evaluate a boolean expression and determine how a program will branch.
• The switch statement allows the programmer to evaluate an integer type or character type variable and determine how a program will branch.
The switch Statement

The switch statement takes the form:

```java
switch (SwitchExpression)
{
    case CaseExpression:
        // place one or more statements here
        break;
    case CaseExpression:
        // place one or more statements here
        break;
        // case statements may be repeated
        // as many times as necessary
    default:
        // place one or more statements here
}
```
The switch Statement
Con’t

• The **SwitchExpression** is an expression that must result in a value of **char**, **byte**, **short**, or **int**

• The **CaseExpression** is a literal or a final variable which must be of **char**, **byte**, **short**, **int** types.

• The **CaseExpression** must be unique.

• The switch statement will evaluate the expression.

• If there is an associated case statement that matches that value, program execution will be transferred to that case statement.
The switch Statement

Con’t

- The break() statement ends the case statement.
- The break() statement is optional.
- If a case does not contain a break, then program execution continues into the next case.
- The default case is optional and will be executed if no CaseExpression matches the SwitchExpression.
The switch Statement
Example (SwitchDemo class)

```java
import java.util.Scanner;

public class SwitchDemo {

    public static void main(String[] args) {
        int number; //a number entered by the user
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Enter 1, 2, or 3: ");
        number = keyboard.nextInt();
        switch (number) {
            case 1:
                System.out.println("I");
                break;
            case 2:
                System.out.println("II");
                break;
            case 3:
                System.out.println("III");
                break;
            default:
                System.out.println("That's not 1, 2, or 3!");
        }
    }
}
```
The switch Statement
Example (PetFood class)

```java
import java.util.Scanner;

public class PetFood {

    public static void main(String[] args) {
        String input;
        char foodGrade;

        Scanner keyboard = new Scanner(System.in);
        System.out.println("Our pet food is available in three grades:");
        System.out.print("A, B, and C. Which do you want ");
        input = keyboard.nextLine();
        foodGrade = input.charAt(0);
    }
}
```
The switch Statement

Example (PetFood class)

```java
switch(foodGrade){
    case 'a':
    case 'A':
        System.out.println("30 cents per 1b.");
        break;
    case 'b':
    case 'B':
        System.out.println("20 cents per 1b.");
        break;
    case 'c':
    case 'C':
        System.out.println("15 cents per 1b.");
        break;
    default:
        System.out.println("Invalid Choice!");
}
```