Contents

1. Introduction
2. Types and Variables
3. Statements and Control Flow
4. Reading Input
5. Classes and Objects
6. Arrays
7. Methods
8. Scope and Lifetime
9. Utility classes
10. Introduction to Object-Oriented Analysis and Design
Chapter 3: Statements and Control Flow

After this chapter you will be able to:

- Use shorthands for increment/decrement
- Understand and override operator precedence
- Use if statements
- Incorporate for and while loops
Pre and Post-Increment

- `++x` // Pre-increment x
- `x++` // Post-increment x

Examples:

\[
\begin{align*}
\text{a} &= ++x \times b; \\
&\quad \quad \quad \quad \downarrow \\
&\quad \quad \quad \quad \text{a} = x++ \times b; \\
\text{x} &= x + 1; \\
\text{a} &= x \times b; & \quad \quad \quad \quad \downarrow \\
\text{a} &= x \times b; & \quad \quad \quad \quad \text{x} = x + 1;
\end{align*}
\]
Pre and Post-Decrement

• \(--x\)  // Pre-decrement x
• \(x--\)  // Post-decrement x

• Examples:
  \[a = --x \times b;\]  
  \[a = x-- \times b;\]  
  \[x = x - 1;\]  
  \[a = x \times b;\]  
  \[x = x - 1\]
Assignment Shorthands

\[\begin{align*}
x & \;+=\; 5; \quad \longrightarrow \quad x & \;=\; x + 5; \\
x & \;-=\; 10; \quad \longrightarrow \quad x & \;=\; x - 10; \\
x & \;*=\; 4; \quad \longrightarrow \quad x & \;=\; x * 4; \\
x & \;/=\; 3; \quad \longrightarrow \quad x & \;=\; x / 3; \\
x & \;%=\; 2; \quad \longrightarrow \quad x & \;=\; x \% 2; \\
\end{align*}\]

- The shorthands on the left allow the compiler to optimize the assignment
Questions

\[
\begin{align*}
\text{int } a=2, b=3, c; \\
c &= ++a \times b++; \\
a, b, c \ ?
\end{align*}
\]

\[
\begin{align*}
\text{int } a=2, b=3, c=0; \\
c &= --a \times b++; \\
a, b, c \ ?
\end{align*}
\]

\[
\begin{align*}
\text{int } a=4, b=3, c=20; \\
c &= a++; \\
a, b, c \ ?
\end{align*}
\]

\[
\begin{align*}
\text{int } a=2, b=3, c=4; \\
c &= ++a \times b++; \\
a, b, c \ ?
\end{align*}
\]
Unary Operators

- A unary operator operates on a single value

- x
  - x
  - ++x
  - --y
  - +5
Binary Operators

- Binary operators operate on two values

\[
a \leq 5 \\
a + 10 \\
c \lor d
\]
Operator Precedence

• Evaluation of expressions is not always left to right

\[ a = 4 + 5 \times 6; \]

• What value is in \( a \)?

• Multiplication operator has higher precedence than the addition operator
Operator Precedence cont,

• Operators in order of **decreasing** precedence

1. unary +,-
2. *, /, %
3. +,-
4. ==, !=
5. <, <=, >, >=
6. !
7. &&
8. ||
Overriding Precedence

- Can control evaluation order using parenthesis

\[ a = (4 + 5) \times 6; \]

- What value is in \( a \) ?

- Use parentheses to control evaluation order or if unsure of operator precedence
Control Flow Statements

- **if** statement
- **for** loop
- **while** loop
- **switch** statement
If Statement

• Syntax:  \textit{if (bool\_expression) statement}

• The statement is executed if the boolean expression is true

\begin{verbatim}
if (x > 0)
  y = z / x;
\end{verbatim}
If With Else

• Syntax:
  
  \[
  \text{if (bool\_expr) statement1 else statement2}
  \]

• \textit{statement1} is executed if the boolean expression is true. Otherwise \textit{statement2} is executed

```java
if (a < 4)
    System.out.println("a is < 4");
else
    System.out.println("a is >= 4");
```
Grouping Statements

• Use braces to group statements together

```java
if (a < 4) {
    System.out.println("a is < 4");
    a++;
}
```

• What is wrong with this?

```java
if (a < 4)
    System.out.println("a is < 4");
    a++;
```
Dangling Else Problem

• Which if does the else go with?
• In Java an else will be associated with the closest if

```java
if (a < 4)
    if (a > 2)
        System.out.println("a is 3");
else
    System.out.println("a is >= 4");
```

```java
if (a < 4)
    if (a > 2)
        System.out.println("a is 3");
else
    System.out.println("a is <= 2");
```
Dangling Else with Braces

• Use braces to associate an else with an outer if

```java
if (a < 4){
    if (a > 2)
        System.out.println("a is 3");
} else
    System.out.println("a is >= 4");
```
For Loop

- Syntax: for (init; bool_expr; update) statement
- init is executed once
- as long the boolean expression is true the statement is executed
- update is executed after each iteration

```java
for (int i = 1; i <= 4; i++)
    System.out.println(i);
```

Output

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Declaring Loop Control Variable

- Loop control variable may be declared before or at for loop

```java
for (int i = 1; i <= 4; i++)
    System.out.println(i);
```

OR

```java
int i;

for (i = 1; i <= 4; i++)
    System.out.println(i);
```
Loop Control Variable Visibility

• Variable \( i \) is **not** visible outside the scope of the \texttt{for} loop

```
for (int i = 1; i <= 4; i++){
    System.out.println(i);
}
System.out.println(i);
```
Loop Control Variable Visibility

• Variable \texttt{i} is visible outside the scope of the \texttt{for} loop

```java
int i;

for (i = 1; i <= 4; i++)
    System.out.println(i);
System.out.println(i);
```
for (int i = 1; i <= 4; i++)
    if ((i % 2) == 0 ){
        System.out.print(i);
        System.out.println(" is even");
    }
    else {
        System.out.print(i);
        System.out.println(" is odd");
    }

Output

1 is odd
2 is even
3 is odd
4 is even
While Loop

• Syntax: `while (bool_expression) statement`
• The statement is executed while the boolean expression remains true

```java
int i = 1;
while (i < 5)
    System.out.println(i++);
```

Output

```
1 2 3 4
```
Grouping with Braces

• Use braces to create a loop body with more than one statement

```java
int i = 1;
while (i < 5){
    System.out.print("i = ");
    System.out.println(i++);
}
```

Output

```
i = 1
i = 2
i = 3
i = 4
```
Summing

- Calculate the sum of all numbers from 1 to 10

```java
int sum = 0;
int i = 1;
while (i <= 10)
    sum += i++;

System.out.print("the sum from 1 to 10 is: ");
System.out.println(sum);
```

Output: the sum from 1 to 10 is: 55
Switch Statement

- The `switch` statement dispatches control flow based on an integer or character value

```java
int itemNumber = 2;

switch (itemNumber) {
    case 1:  price = 10; break;
    case 2:  price = 20; break;
    case 3:  price = 30; break;
    default: System.out.println("invalid price");
}
System.out.print("Item price: ");
System.out.println(price);
```

Output: Item price: $20
Default Case

- Default branch is executed if no case matches

```java
int itemNumber = 10;
boolean valid = true;

switch (itemNumber) {
    case 1:  price = 10; break;
    case 2:  price = 20; break;
    case 3:  price = 30; break;
    default:  valid = false;
        System.out.println("invalid price");
}
if (valid) {
    System.out.print("Item price: $ ");
    System.out.println(price);
}
```

**Output**

```
invalid price
```
Switch with Characters

cchar charCode = 'a';
boolean valid = true;

switch (charCode) {
    case 'a':  price = 10; break;
    case 'b':  price = 20; break;
    case 'c':  price = 30; break;
    default:   valid = false;
        System.out.println("invalid price");
    }
if (valid) {
System.out.print("Item price: ");
System.out.println(price);
}
Multiple Case Labels

• All three cases correspond to the same branch

```java
char charCode = 'a';
boolean valid = true;

switch (charCode) {
    case 'a':
    case 'b':
    case 'c':
        price = 30; break;
    default:
        valid = false;
        System.out.println("invalid price");
}
if (valid) {
    System.out.print("Item price: $ ");
    System.out.println(price);
}
```
It’s Exercise Time