



Java Programming

Chapter 3

Conditions & Switch

Condition

- Conditions and if statements let you control the flow of your program
 - deciding which code runs, and which code is skipped.
 - *If it rains, take an umbrella. Otherwise, do nothing.*
 - Every if statement needs a condition that results in true or false.

```
1 public class Main {  
2     public static void main(String[] args) {  
3         boolean isRaining = true;  
4  
5         if (isRaining) {  
6             System.out.println("Bring an umbrella!");  
7         }  
8     }  
9 }  
10
```

If-statement

- Most often, conditions are created using comparison operators, like the ones below:
 - Less than: $a < b$
 - Less than or equal to: $a \leq b$
 - Greater than: $a > b$
 - Greater than or equal to: $a \geq b$
 - Equal to: $a == b$
 - Not equal to: $a != b$

If-statement

- The if statement specifies a block of code to be executed if a condition is true:

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

```
boolean isLightOn = true;  
  
if (isLightOn) {  
    System.out.println("The light is on.");  
}
```

If-statement

```
boolean isLightOn = true;  
  
if (isLightOn) {  
    System.out.println("The light is on.");  
}
```

If-statement

- Potential Problem
 - Without braces, only the first line after the if belongs to it. Any other lines will run no matter what, which can lead to unexpected results:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int x = 20;  
4         int y = 18;  
5  
6         if (x > y)  
7             System.out.println("x is greater than y");  
8             System.out.println("This line runs no matter what (not part of the if statement)");  
9     }  
10 }
```

If-statement

- The Safe Way
 - To avoid mistakes, always use curly braces { }. This makes it clear which lines belong to the if statement:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int x = 20;  
4         int y = 18;  
5  
6         if (x > y) {  
7             System.out.println("x is greater than y");  
8             System.out.println("Both lines are part of the if");  
9         }  
10  
11         System.out.println("I am outside if, not part of if!");  
12     }  
13 }  
14
```

If-statement

- The else Statement
 - The else statement lets you run a block of code when the condition in the if statement is false.

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int time = 20;  
4         if (time < 18) {  
5             System.out.println("Good day.");  
6         } else {  
7             System.out.println("Good evening.");  
8         }  
9     }  
10 }  
11 // Outputs "Good evening."
```


If-statement

Exercise

- Select the correct text that will be printed when time is 20.

```
int time = 20;  
if (time < 18) {  
    System.out.println("Good day");  
} else {  
    System.out.println(" ");  
}
```

Good morning

Good day

Good evening

Hello

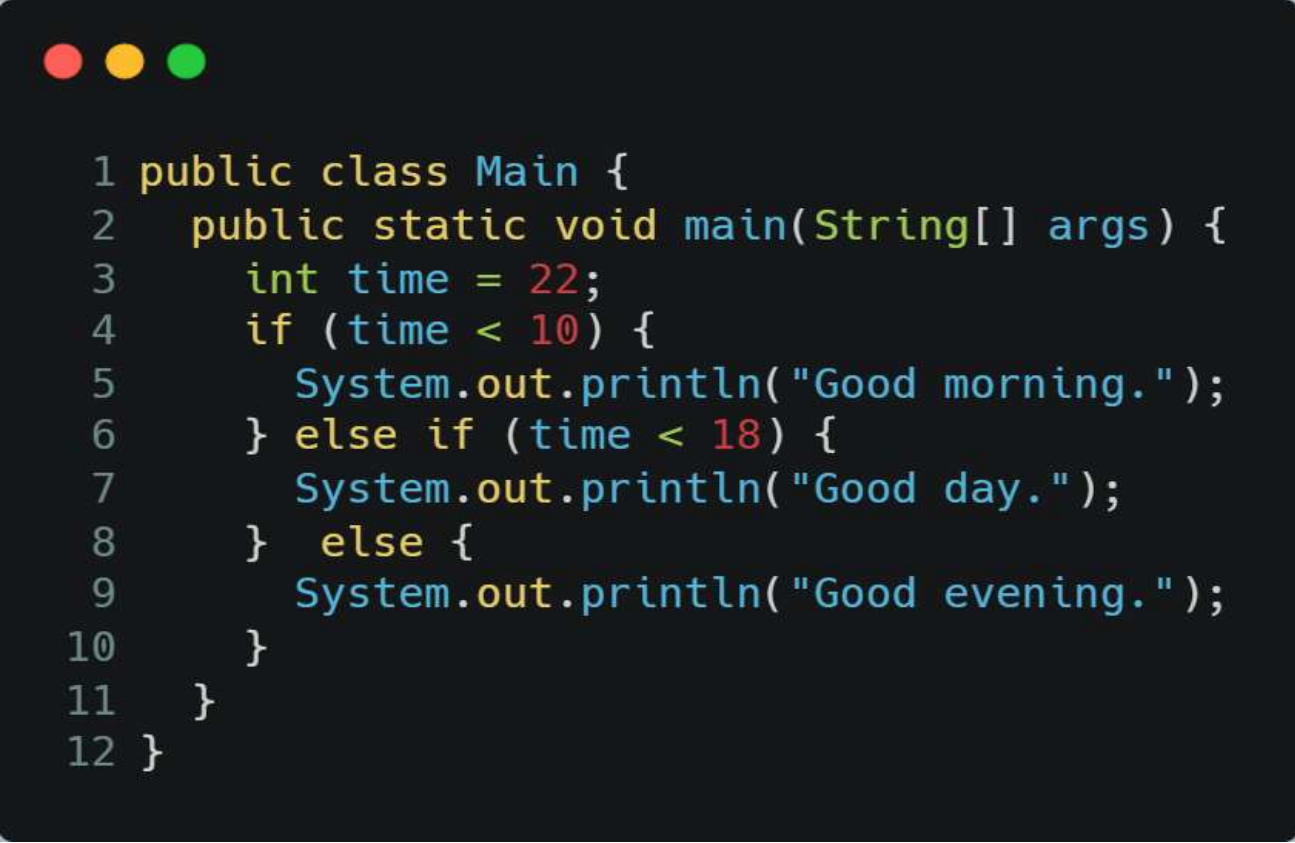
else if Statement

- The else if Statement
 - Use the else if statement to specify a new condition if the first condition is false.


```
1 public class Main {  
2     public static void main(String[] args) {  
3         int weather = 2; // 1 = raining, 2 = sunny, 3 = cloudy  
4  
5         if (weather == 1) {  
6             System.out.println("Bring an umbrella.");  
7         } else if (weather == 2) {  
8             System.out.println("Wear sunglasses.");  
9         } else {  
10            System.out.println("Just go outside normally.");  
11        }  
12    }  
13 }  
14  
15
```

If-statement


In the example above, time (22) is greater than 10, so the first condition is false. The next condition, in the else if statement, is also false, so we move on to the else condition since condition1 and condition2 is both false - and print to the screen "Good evening".



```
1 public class Main {  
2     public static void main(String[] args) {  
3         int time = 22;  
4         if (time < 10) {  
5             System.out.println("Good morning.");  
6         } else if (time < 18) {  
7             System.out.println("Good day.");  
8         } else {  
9             System.out.println("Good evening.");  
10        }  
11    }  
12 }
```



However, if the time was 14, our program would print "Good day.":



```
1 int time = 14;
2 if (time < 10) {
3     System.out.println("Good morning.");
4 } else if (time < 18) {
5     System.out.println("Good day.");
6 } else {
7     System.out.println("Good evening.");
8 }
```


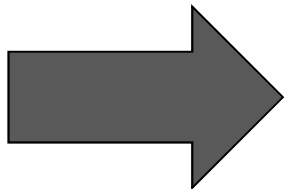
- Short Hand if...else

- Syntax

- `variable = (condition) ? expressionTrue : expressionFalse;`



```
1 int time = 20;
2 if (time < 18) {
3     System.out.println("Good day.");
4 } else {
5     System.out.println("Good evening.");
6 }
```



```
1 int time = 20;
2 String result = (time < 18) ? "Good day." :
    "Good evening.";
3 System.out.println(result);
```

Nested If

- You can also place an if statement inside another if. This is called a nested if statement.
- A nested if lets you check for a condition only if another condition is already true.
 - Ex) we first check if x is greater than 10. If it is, we then check if y is greater than 20:

```
1 int x = 15;
2 int y = 25;
3
4 if (x > 10) {
5     System.out.println("x is greater than
6     10");
7     // Nested if
8     if (y > 20) {
9         System.out.println("y is also greater
10        than 20");
11    }
```

Nested If

- Nested if statements are useful when you need to test multiple conditions that depend on each other.
- For example, checking if a person is old enough to vote, and if they are a citizen:

```
1 int age = 20;
2 boolean isCitizen = true;
3
4 if (age >= 18) {
5     System.out.println("Old enough to vote.");
6
7     if (isCitizen) {
8         System.out.println("And you are a citizen, so you can
9         vote!");
10    } else {
11        System.out.println("But you must be a citizen to vote.");
12    }
13 } else {
14     System.out.println("Not old enough to vote.");
15 }
```

```
Old enough to vote.
And you are a citizen, so you can vote!
```

Nested If

- Real-Life Style Example
 - In real programs
 - logical operators are often used for access control.
 - For example, to get access to a system, there are specific requirements:
 - You must be logged in, and then you either need to be an admin, or have a high *security clearance* (level 1 or 2):
 - Access Control



```
1 boolean isLoggedIn = true;
2 boolean isAdmin = false;
3 int securityLevel = 3; // 1 = highest
4
5 if (isLoggedIn && (isAdmin || securityLevel <= 2)) {
6     System.out.println("Access granted");
7 } else {
8     System.out.println("Access denied");
9 }
10
11 // Try changing securityLevel to test different outcomes:
12 //
13 // securityLevel 1 = Access granted
14 // securityLevel 2 = Access granted
15 // securityLevel 3 = Access denied
16 // securityLevel 4 = Access denied
17 //
18 // If isAdmin = true, access is granted.m.out.println("Not old
    enough to vote.");
19 }
```

Switch

- Instead of writing many if..else statements, you can use the switch statement.
- ordering food in a restaurant:
 - If you choose number 1, you get Pizza.
 - If you choose 2, you get a Burger.
 - If you choose 3, you get Pasta.
 - Otherwise, you get nothing.
- The switch statement selects one of many code blocks to be executed:

Switch

- This is how it works:
 1. The switch expression is evaluated once.
 2. The result is compared with each case value.
 3. If there is a match, the matching block of code runs.
 4. The break statement stops the switch after the matching case has run.
 5. The default statement runs if there is no match.

```
1 switch(expression) {  
2     case x:  
3         // code block  
4         break;  
5     case y:  
6         // code block  
7         break;  
8     default:  
9         // code block  
10 }
```

Switch

- ordering food in a restaurant:
 - If you choose number 1, you get Pizza.
 - If you choose 2, you get a Burger.
 - If you choose 3, you get Pasta.
 - Otherwise, you get nothing


```
1 int day = 2
2 switch (day) {
3     case 1:
4         System.out.println("Pizza");
5     case 2:
6         System.out.println("Burger");
7     case 3:
8         System.out.println("Pasta");
9     default:
10        System.out.println("Nothing");
11 }
12 // Outputs "Thursday" (day 4)
```

break

- When Java reaches a break keyword, it breaks out of the switch block.
- This will stop the execution of more code and case testing inside the block.
- When a match is found, and the job is done, it's time for a break. There is no need for more testing.

break

- To correct the code , we insert break statement



```
1 int day = 2
2 switch (day) {
3     case 1:
4         System.out.println("Pizza");
5         break;
6     case 2:
7         System.out.println("Burger");
8         break;
9     case 3:
10        System.out.println("Pasta");
11        break;
12    default:
13        System.out.println("Nothing");
14 }
15 // Outputs "Thursday" (day 4)
```

Exercise

- Fill in the blank

```
int day = 2;  
[ ] (day) {  
    [ ] 1:  
        System.out.println("Monday");  
        [ ] ;  
    case 2:  
        System.out.println("Tuesday");  
        break;  
}
```

break

case

switch

default