Al-Mustaqbal University
College of Sciences
Intelligent Medical System Department

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الانظمة الطبية الذكية

Lecture: (8)

Branching and Selection Part I

Subject: Computer Programming (I)
Level: First
Lecturer: Dr. Maytham N. Meqdad

Relational Operators in Java

Relational or Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to find answers and make decisions.

The return value of a comparison is either `true` or `false`. These values are known as *Boolean values*, and you will learn more about them in the Booleans and If..Else chapter.

In the following example, we use the **greater than** operator (>) to find out if 5 is greater than 3:

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 3;
    System.out.println(x > y); // returns true, because 5 is higher than 3
  }
}
```

Output: true

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 3;
    System.out.println(x == y); // returns false because 5 is not equal to 3
  }
}
```

Output: false

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 3;
    System.out.println(x != y); // returns true because 5 is not equal to 3
  }
}
```

Output: true

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 3;
    System.out.println(x > y); // returns true because 5 is greater than 3
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 3;
    System.out.println(x < y); // returns false because 5 is not less than 3
```

```
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 3;
    System.out.println(x >= y); // returns true because 5 is greater, or equal, to 3
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 3;
    System.out.println(x <= y); // returns false because 5 is neither less than or equal to 3
  }
}
```

Java Logical Operators

You can also test for `true` or `false` values with logical operators. Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    System.out.println(x > 3 && x < 10); // returns true because 5 is greater than 3 AND 5 is less than 10
  }
}
```

Output: true

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    System.out.println(x > 3 || x < 4); // returns true because one of the conditions are true (5 is greater than 3, but 5 is not less than 4)
```

```
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    System.out.println(!(x > 3 && x < 10)); // returns false because ! (not) is used to reverse the result
  }
}
```

• The if Statement

Java Conditions and If Statements

You already know that Java supports the usual logical conditions from mathematics:

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

You can use these conditions to perform different actions for different decisions.

Java has the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

The if Statement

Use the `if` statement to specify a block of Java code to be executed if a condition is `true`.

Syntax

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

Note that `if` is in lowercase letters. Uppercase letters (If or IF) will generate an error.

In the example below, we test two values to find out if 20 is greater than 18. If the condition is `true`, print some text:

```
public class Main {
  public static void main(String[] args) {
    if (20 > 18) {
      System.out.println("20 is greater than 18"); // obviously
    }
  }
}
```

Output: 20 is greater than 18

We can also test variables:

```
public class Main {
  public static void main(String[] args) {
    int x = 20;
    int y = 18;
    if (x > y) {
      System.out.println("x is greater than y");
    }
  }
}
```

The if-else Statement

Use the `else` statement to specify a block of code to be executed if the condition is `false`.

Syntax

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

```
public class Main {
  public static void main(String[] args) {
    int time = 20;
    if (time < 18) {
      System.out.println("Good day.");
    } else {
      System.out.println("Good evening.");
    }
  }
}
```

Output: Good evening.