Selection – `if`…`then`…`else` Structure

Boolean Expressions

Boolean expressions are expressions that evaluate to `true` or `false`.

Examples

Expression	Value
`6 > 9`	`false`
`'a' < 'b'`	`true (compare the ASCII code)`
`7 + 2 != 9`	`false`

Relational Operators

Rational operators are used to form Boolean expressions.

Operators	Meaning
`= =`	Equals (important to use **two**; one equal sign will be an assignment statement)
`!=`	not equal to
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to

Scenario

Consider the following scenario where a person is logging into a system (like school account or gmail). In this case, the program needs to react differently under different conditions: if the login information is correct, they should enter the system and if something is incorrect then they should not be permitted entry.

Selection structures, also called decision structures, are used in programs that require different execution paths in different situations. In selection, execution paths are dependent on the evaluation of certain values. In most programming languages, selection structures include `if`, `if-else`, `if-else if` & `switch-case` statements

The `if` Statement

Syntax

```
if (condition)
{
   //body of if clause
}
```

The condition in an `if` statement is a Boolean expression, which evaluates to either `true` or `false`.

Example

```
int x;

System.out.print("Enter a number");
x = sc.nextInt();

if (x > 100)
{
   System.out.println("Your number is greater than 100");
}
```

The `if-else` Statement

The `if` statement can include an optional `else` clause that is executed when the if condition evaluates to false.

Syntax

```
if (condition)
{
   //body of if clause
}
else
{
   //body of else clause
}
```

Example

```
final int FREEZE = 0;

double temp;

System.out.print("Enter the temperature");
temp = sc.nextDouble();

if (temp > FREEZE)
{
      System.out.println("the ice will melt");
}
else
{
      System.out.println("the water will freeze");
}
```

The `if-else if` Statement

The `if-else if` statement is used to decide among three or more possible path of execution. Remember: the **first** condition found true is the **only** body executed; only one body can be executed.

Syntax

```
if (condition)
```

```
{
   //body of if clause
}
else if (condition)
{
   //body of 1st else if clause
}
else if (condition)
{
   //body of 2nd else if clause
}
else
{
   //body of else clause
}
```

Example

```
float mark;
System.out.print("Enter your mark: ");
mark = sc.nextFloat();

if (mark >= 90)
{
     System.out.println("grade = A+");
}
else if (mark >= 80)
{
     System.out c.println("grade = A");
}
else if (mark >= 70)
{
     System.out.println("grade = B");
}
else if (mark >= 60)
{
     System.out.println("grade = C");
}
else if (mark >= 50)
{
     System.out.println("grade = D");
}
else
{
      System.out.println("grade = F (failure)");
}
```

Please note that there is no semicolon after the `if` statement, `else-if` statement and `else` statement. The { and } is used to specify the block of code that fall under the each condition.

Nested `if` Statements

A nested `if` structure is to have one `if..else` structure within another one. The following example determines the largest among three values `x`, `y`, `z` and assigns this value to `largest`.

```
if (x >= y) {
   // y eliminated – largest must be either x or z
   if (x >= z) {
      largest = x;
   } else {
      largest = z;
   }
} else {
   // x eliminated – largest must be either y or z
   if (y >= z) {
      largest = y;
   } else {
      largest = z;
   }
}
```

Compound Boolean Expressions

Boolean expressions can be acted upon by Boolean operators to produce new Boolean expressions.

There are three commonly used Boolean operators in Java: `!` (not), `&&` (and), and `||` (or).

Truth Tables

These tables illustrate the behavior of the Boolean (logical) Operators given the different values for expressions A and B.

Expression A	**Expression B**	**!(A)**	**A || B**	**A && B**
false	false	true	false	false
false	true	true	true	false
true	false	false	true	false
true	true	false	true	true

Compound Boolean Expressions are especially useful when specifying a range.

Example

```
if (num >= 0 && num <= 100) {
      System.out.println("Valid Test mark.");
}

if (input >= 'a' && input <='z') {
      System.out.println("Lower case letter.");
}
```

Tracing

Example

```
int i;
System.out.println(“Enter an integer”);
i = sc.nextInt();
if (i >= 3)
{
    System.out.println ("yep");
}
else if ((i == 0) || (i < 2))
{
    System.out.println ("jump");
}
else if (i != 2)
{
    System.out.println ("nobody");
}
else
{
    System.out.println ("dump");
}
```

	Memory	Output
User inputs `4`	`i(i): 4`	`yep`
User inputs `2`	`i(i): 2`	`dump`