Conditionals. Boolean expressions.

Conditional statements are used to select a course of action based on the value of a specified condition. Java’s conditional statements are:

- if statement
- if-else statement
- switch statement

A condition often uses one of Java’s equality operators or relational operators, which all return boolean results (true or false):

- `==` equal to
- `!=` not equal to
- `<=` less than or equal to
- `>=` greater than or equal to
- `>` greater than
- `<` less than

Note the difference between the equality operator (==) and the assignment operator (=).

Also note the difference in comparing two strings with == versus the `equals()` method. The first compares the references, the second compares the actual content of the strings.

1. **if-statement syntax:**
   ```java
   if (condition) {
       if-block
   }
   
or
   if (condition)
   if-block;
   
in case the if-block only consists of one statement.

Examples:

```java
delta = 0;
if (sum > MAX) {
    delta = sum - MAX;
}
System.out.println ("The sum is " + sum);
```

- Suppose sum = 50, MAX = 30. What will be the value of delta?
- What if sum = 20, MAX = 30
2. if-else statement syntax

```java
if ( condition ) {
    if-block
}
else {
    else-block
}

{}'s can be dropped around an if- or an else-block when it consists of just one statement.
```

Examples:

```java
if(time < limit)
    System.out.println("You made it.");
else
    System.out.println("You missed the deadline.");
System.out.println("Bye!");
```

Nested conditionals:

```java
if(time < limit){
    System.out.println("You made it.");
    if (limit - time >= 10 )
        bonus = 100;
    else
        bonus = 50;
} else{
    System.out.println("You missed the deadline.");
    bonus = 0;
}
```

Multibranch selection

```java
if(score >= 90)
    grade = 'A';
else if (score >= 80)
    grade = 'B';
else if (score >= 70)
    grade = 'C';
else if (score >= 60)
    grade = 'D';
else
    grade = 'E';
```

3. Boolean (logical) operators: used to form complex conditions.

```
! Logical NOT
&& Logical AND
|| Logical OR
```

They all take boolean operands and produce boolean results
Logical NOT is a unary operator (it operates on one operand). Logical AND and logical OR are binary operators (each operates on two operands).

Rules:

The logical NOT expression – negates the argument.

if some boolean condition a is true, then !a is false; if a is false, then !a is true

The logical AND expression  a && b is true if both a and b are true, and false otherwise

The logical OR expression  a || b is true if a or b or both are true, and false otherwise

Precedence: NOT(!) is evaluated first, then AND(&&), then OR (||).

Examples:

- Find a value of n that will result in the execution of the if-block; else-block

```java
if (n % 2 == 0 && n > 5 || n <= 3) {
    x = 35;
} else {
    x = 20;
}
```

- Identify one set of values of the variables and constants below that will result in the execution of the printing statement.

```java
// note: type of found is Boolean
boolean found;
...
if (total < MAX+5 && !found) {
    System.out.println("Processing...");
}
```

Practice problems:

- Assume variable yearHired stores the year an employee was hired. Write a code segment that prints “found” if the yearHired is in 1990’s but is not one of 1992 or 1993.

- Assuming str1 and str2 are two variables of type String, write a code segment that would print “one” if str1 and str2 store the identical strings, “two” if they are not identical, but start with the same letter, and “three” in case they end with the same letter.
4. Another way to program multibranch selection

Switch statement: syntax

```
switch(Controlling_Expression)
{
    case Case_Label_1:
        statements
        ...
        break;
    case Case_Label_2:
        statements
        ...
        break;
    ...
    default:
        statements
        ...
        break;
}
```

Example:

```
// seatLocationCode is an int variable
switch(seatLocationCode)
{
    case 1:
        System.out.println("Orchestra");
        price = 40.00;
        break;
    case 2:
        System.out.println("Mezzanine");
        price = 30.00;
        break;
    case 3:
        System.out.println("Balcony");
        price = 15.00;
        break;
    default:
        System.out.println("Unknown seat code");
        break;
}
```

- Controlling_Expression must be char, int, short or byte
- Controlling Expression and Case_Label must have the same type
- When a break statement is encountered, control goes to the first statement after the switch.
**Practice Problem.** Write a program that will calculate and print bills for the city power company. The rates vary depending on whether the use is residential, commercial or industrial. A code of R means residential use, a code of C means commercial use and a code of I means industrial use. Any other code should be treated as an error.

The rates are as follows:
- Residential: $6.00 plus $0.052 per kwh used
- Commercial: $60.00 for the first 1000 kwh and $0.045 for each additional kwh
- Industrial: Rate varies depending on time of usage.
  - Peak hours: $76.00 for first 1000 kwh and $0.028 for each additional kwh
  - Off peak hours: $40.00 for first 1000 kwh and $0.028 for each additional kwh

Your program should prompt the user to enter an integer account number, the use code (type char), and the necessary consumption figures in kilowatt-hours. For the industrial account prompt the user to enter “yes” if the usage corresponds to peak hours, and “no” for off peak hours. Your program should print the amount due from the user.