Language elements

Overview

- Statements and expressions
- Variables and primitive data types
- Constants
- Comments
- Literals
- Arithmetic
- Comparisons
- Logical operators

Statements and expressions

- A **statement** is a command that causes something to happen.

  Example:

  ```
  int weight = 225;
  System.out.println(“My weight is ” + weight);
  ```

- An **expression** is a statement that produces a value. The value produced by the statement is called a **return value**.

  Example:

  ```
  int sum = add(x, y);
  ```

Variables

- Used to store information (data) while the program is running.
- Types of variables:
  - **Class variables** - used to define the attributes for an entire class of objects and apply to all instances of it.
  - **Instance variables** - used to defined the object's attributes.
  - **Local variables** - used inside method definitions or even smaller blocks of statements within a method.

Declaring variables

- Variable declaration:
  - `int loanLength;`
  - `String message;`
  - `boolean gameOver;`

- Assigning initial values:
  - `int zipCode = 1000;`
  - `String city = “Skopje”;`

Naming variables

- The name must start with a letter, an underscore character ("_"), or a dollar sign ("$"). In practice, prefer to use letters only.
- Names are case sensitive.
- CamelCase notation:
  - The first letter of the variable name is lowercase.
  - Each successive world in the variable name begins with a capital letter.
  - All other letters are lowercase.
  - Examples:
    - `Button loadFile;`
    - `int localAreaCode;`
    - `boolean quitGame;`

Variable types

- A variable in Java can be declared as one of these three types:
  - One of the primitive data types
  - The name of a class or interface
  - An array

Primitive data types

- Eight primitive data types are part of the Java language:
  - Four to store integers:
    - **byte** (8 bits) - 128 to 127
    - **short** (16 bits) - 32,768 to 32,767
    - **int** (32 bits) - 2,147,483,648 to 2,147,483,647
    - **long** (64 bits) - 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
  - Two floating-point numbers:
    - **float** - 1.4E-45 to 3.4E+38
    - **double** - 4.9E-324 to 1.7E+308
  - The **char** type is used for individual characters such as letters, numbers, punctuation, and other symbols.
  - The **boolean** type can hold either a true or a false value.

Class types

- A variable can be of Java's built-in classes, a third-party class or a class explicitly defined in the project.

  Examples:

  - `String lastName = “Hopper”;`
  - `Color hair;`
  - `VolcanoRobot vr;`

Assigning values to variables

- Assigning a value is done using the assignment operator which is the equal sign ("=")

  Examples:
  - `idCode = 8675309;`
  - `accountOverdrawn = false;`

Constants

- A **constant** is a variable which is not allowed to change its value as the program runs.
- In Java, constants are defined using the **final** keyword.
- Examples:
  - `final float PI = 3.141592;`
  - `final boolean DEBUG = false;`
  - `final int PENALTY = 25;`

Comments

- Used to improve the readability of the program.
  - **Single line comments** - denoted by two slash characters "//“.

    Example:

    ```
    int creditHours = 3; // set up credit hours for course
    ```

  - **Multiple line comments** - everything that's in between "/*" and "*/“.

    Example:

    ```
    /* This program occasionally deletes all files on
    your hard drive and renders it completely unusable
    when you press the Save button. */
    ```

Comments

- **Javadoc comments** - everything that's in between "/**" and "*/". Considered to be official documentation for the code an can be extracted by tools to create the source code documentation.

  Example:

```
/**
* Returns <tt>true</tt> if this list contains no elements.
*
* @return <tt>true</tt> if this list contains no elements.
*/
boolean isEmpty();
```

Literals

- Number literals: `10, -0x101, 0777, 0xFF`
- Boolean literals: `true, false`
- Character literals: `'a', '#', '3'`
  - Special characters:
    - `\n` - New line, `\t` – Tab, `\b` – Backspace, `\r` - Carriage return, `\f` – Formfeed, `\\` - Backslash, `\’` - Single quotation mark, `\”` - Double quotation mark, `\d` – Octal, `\xd` – Hexadecimal, `\ud` - Unicode character
- String literals: `"Socrates asked, \"Hemlock is poison?\""`

Arithmetic operations

Operator	Meaning	Example
+	Addition	3 + 4
-	Subtraction	5 - 7
*	Multiplication	5 * 5
/	Division	14 / 7
%	Modulus	20 % 7

Example: Weather

Shorthand assignment operators

Expression	Meaning
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y

Increment and decrement operators

- Incrementing and decrementing are used very often so Java provides special operators to make this less verbose.

  Examples:
  - `int x = 7;`
  - `x++; // The new value is 8.`
- When using in an expression, it's important whether the increment or decrement operator is used in a **prefix** (e.g. ++x) or a **postfix** (e.g. x++) notation.

  Examples:
  - `int x, y, z; // x, y, and z are all declared.`
  - `x = 42; // x is given the value of 42.`
  - `y = x++; // y is given x’s value (42) before it is incremented and x is then incremented to 43.`
  - `z = ++x; // x is incremented to 44, and z is given x’s value.`

Comparison operators

Operator	Meaning	Example
==	Equal	x == 3
!=	Not equal	x != 3
<	Less than	x < 3
>	Greater than	x > 3
<=	Less than or equal to	x <= 3
>=	Greater than or equal to	x >= 3

Logical operators

Operator	Meaning	Example
&&	AND	(score > 75000) && (playerLives < 10)
||	OR	(score > 75000) || (playerLives == 0)
^	XOR	(score > 75000) ^ (playerLives == 0)
!	NOT	!(age < 30)

String arithmetic

- The "+" operator can be used outside of mathematics to concatenate (combine) two or more strings.

  Example:

  ```
  String firstName = “Raymond”;
  System.out.println(“Everybody loves “ + firstName);
  ```

  prints

  ```
  "Everybody loves Raymond"
  ```

Exercises

Exercise: InvestmentCalculator

- Create a program that calculates how much a $14,000 investment would be worth if it increased in value by 40% during the first year, lost $1,500 in value the second year, and increased 12% in the third year.

Exercise: QuotientAndRemainder

- Write a program that displays two numbers and uses the "/" and "%" operators to display the result and remainder after they are divided. Use the "\t" character escape code to separate the result and remainder in your output.