SchoolNova

IT101

Anatomy of a Java Program

Java Comments

- Include comments in your programs in order to make them more readable/understandable.
- Block comment syntax:

```
/*
  Multi-line
  Comment
 */
```

- One line comment syntax: // ...
- Commented text is ignored by the compiler.
- Style: Include a prologue section at the top of every class/program. The prologue section consists of:
  - filename
  - programmer's name
  - program description
  - version

```
/**
 * Graphics is the abstract base class for all graphics contexts
 * which allow an application to draw onto components realized on
 * various devices or onto off-screen images.
 * A Graphics object encapsulates the state information needed
 * for the various rendering operations that Java supports.  This
 * state information includes:
 * <ul>
 * <li>The Component to draw on
 * <li>A translation origin for rendering and clipping coordinates
 * <li>The current clip
 * <li>The current color
 * <li>The current font
 * <li>The current logical pixel operation function (XOR or Paint)
 * <li>The current XOR alternation color
 *     (see <a href="#setXORMode">setXORMode</a>)
 * </ul>
 * <p>
 * Coordinates are infinitely thin and lie between the pixels of the
 * output device.
 * Operations which draw the outline of a figure operate by traversing
 * along the infinitely thin path with a pixel-sized pen that hangs
 * down and to the right of the anchor point on the path.
 * Operations which fill a figure operate by filling the interior
 * of the infinitely thin path.
 * Operations which render horizontal text render the ascending
 * portion of the characters entirely above the baseline coordinate.
 * <p>
 * Some important points to consider are that drawing a figure that
 * covers a given rectangle will occupy one extra row of pixels on
 * the right and bottom edges compared to filling a figure that is
 * bounded by that same rectangle.
 * Also, drawing a horizontal line along the same y coordinate as
 * the baseline of a line of text will draw the line entirely below
 * the text except for any descenders.
 * Both of these properties are due to the pen hanging down and to
 * the right from the path that it traverses.
 * <p>
 * All coordinates which appear as arguments to the methods of this
 * Graphics object are considered relative to the translation origin
 * of this Graphics object prior to the invocation of the method.
 * All rendering operations modify only pixels which lie within the
 * area bounded by both the current clip of the graphics context
 * and the extents of the Component used to create the Graphics object.
 *
 * @author      Sami Shaio
 * @author      Arthur van Hoff
 * @version     %I%, %G%
 * @since       1.0
 */
public abstract class Graphics {
```

Class and Package

- All Java programs must be enclosed in a *class*. Think of a class as the name of the program.
- The name of the Java program's file must match the name of the Java program's class (except that the filename has a .java extension added to it).
- Proper style dictates that class names start with an uppercase first letter.
- Since Java is *case-sensitive*, that means the filename should also start with an uppercase first letter. Case-sensitive means that the Java compiler does distinguish between lowercase and uppercase letters.
- All classes in Java belong to some package.
- To create a package, put a “package” command at the top of a Java source file:
  - package com.schoolnova.it101;
- Java uses the file system to manage packages, with each package stored in its own directory.
- Package names are case-sensitive.
- You can create a hierarchy of packages by separating each package name with a period (see example above).

Main method

- Use “public class” access modifier prior to your class name. For example:
  - public class Fibonacci
- Inside your class, you must include one or more *methods*.
- **A method is a group of instructions that solves one task**. Large programs require multiple methods because they solve multiple tasks. A Java method is equivalent to a JavaScript function.
- Use this main method signature for the method that starts your program:
  - public static void main(String[] args)
- When a program starts, the JVM looks for the **main** method and begins execution with it.

System.out.println

- To generate output, use System.out.println().
- For example, to print the hello message, do this:
- System.out.println("Hello, world!");
- Note:
  - Put the printed item inside the parentheses.
  - Surround strings with quotes.
  - Put a semicolon at the end of a statement.
- What's the significance of the ln in println?

Identifiers

- Identifier is the technical term for a name in a programming language
- Identifier examples:
  - class name identifier: Fibonacci
  - method name identifier: main
  - variable name identifier: height
- Identifier naming rules:
  - Must consist entirely of letters, digits, dollar signs ($), and/or underscore (_) characters.
  - The first character must not be a digit.
  - If these rules are broken, your program won't compile.
- Identifier naming conventions:
  - All letters must be lowercase except the first letter in the second, third, etc. words. For example: firstName, x, daysInMonth
  - For class names, the first letter in every word (even the first word) must be uppercase. For example: StudentRecord, WorkShiftSchedule
  - Names must be descriptive.

Variables

- A variable can hold only one type of data. For example, an integer variable can hold only integers, a string variable can hold only strings, etc.
- How does the computer know which type of data a particular variable can hold?
  - Before a variable is used, its *type* must be *declared* in a *declaration* statement.
- Declaration statement syntax:
  - *<type> <one or more variables separated by commas>*;
- Example declarations:
  - String firstName = Ethan; // student's first name, initialized variable
  - String lastName; // student's last name
  - int studentId;
  - int gpa, age;
-

Trace this code:

```
int salary;
String bonusMessage;
salary = 50000;
bonusMessage = "Bonus = $" + (.02 * salary);
System.out.println(bonusMessage);
```

Reserved Words and Modifiers

- Java uses modifiers that specify the properties of the data, methods, and classes and how they can be used.
  - public
  - private
  - protected
  - static
  - final
  - abstract
- A **public** variable, method or class can be accessed by other programs.
- A **private** variable or method can be accessed by within the container class, but cannot be accessed by other classes/programs.
- Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program.
- For example, when the compiler sees the word class, it understands that the word after class is the name for the class.
- Other examples of reserved words are public, static, and void

Numeric Data Types

- Variables that hold whole numbers (e.g., 1000, -22) should normally be declared with one of these integer data types – int, long.
- Range of values that can be stored in an int variable: -2 billion to +2 billion
- Range of values that can be stored in a long variable: $-9x10^{18}$ to $+9x10^{18}$
- Example integer variable declarations:
  - int studentId;
  - long satelliteDistanceTraveled;
- Recommendation: Use smaller types for variables that will never need to hold large values.
- The int data type is Java default for whole numbers.

- Variables that hold decimal numbers (e.g., -1234.5, 3.1452) should be declared with one of these floating-point data types – float, double.
- Range of values that can be stored in a float variable: $-3.4*10^{38}$ to $+3.4*10^{38}$
- Range of values that can be stored in a double variable: $-3.4*10^{308}$ to $+3.4*10^{308}$
- The double type stores numbers using 64 bits whereas the float type stores numbers using only 32 bits. That means that double variables are better than float variables in terms of being able to store bigger numbers and numbers with more significant digits.
- The double data type is Java default for floating point numbers.

- Assigning an integer value into a floating-point variable works just fine. Note this example:
  - double bankAccountBalance = 1000;
- On the other hand, assigning a floating-point value into an integer variable is like putting a large object into a small box. By default, that's illegal. For example, this generates a compilation error:
  - int temperature = 26.7;
- This statement also generates a compilation error:
  - int count = 0.0;

Arithmetic Operators

- Java's +, -, and * arithmetic operators perform addition, subtraction, and multiplication in the normal fashion.
- If you divide 7.0 by 2.0 on your calculator, you get 3.5, because 7.0 and 2.0 are doubles.
- In the following example: 5 / 4.0, 5 is an int and 4. is a double. This is an example of a mixed expression. A mixed expression is an expression that contains operands with different data types. Whenever there's a mixed expression, the JVM temporarily promotes the less-complex operand's type so that it matches the more-complex operand's type, and then the JVM applies the operator. In the 5 / 4. expression, the 5 gets promoted to a double and then floating-point division is performed. The expression evaluates to 1.25.
- The % operator (called the modulus operator) also performs "grade school" division and generates the remainder. For example: 7 % 2 !
- Use the increment operator (++) operator to increment a variable by 1. Use the decrement operator (--) to decrement a variable by 1. Here's how they work:
  - `x++;  // x = x + 1;`
  - `x--;  // x = x - 1;`
- The compound assignment operators are: **+=**, **-=**, ***=**, **/=**, **%=**
- The variable is assigned an updated version of the variable's original value. Here's how they work:
  - `x += 3;  // x = x + 3;`
  - `x -= 4;  // x = x - 4;`

Homework

- Arithmetic operations in Java are very similar to JavaScript.
- Create a program that calculates the approximate year of birth from the user's age input

```
int age, dob;
Scanner input = new Scanner(System.in);
System.out.println("What is your age?");
age = input.nextInt();
// your year of birth calculation goes here
System.out.println("You were born around year " + dob);
```

• NetBeans basic tutorial can be found here: https://docs.oracle.com/javase/tutorial/getStarted/cupojava/netbeans.html
• Go through steps 1-5 only.

```
/*
     Program shell
*/
package com.schoolnova.it101;
import java.util.Scanner;
public class DOBCalculator {
   public static void main(String[] args) {
     // your code goes here
   }
}
```

If you are not using NetBeans

Helpful command line operations, **if you are not using NetBeans**:

1) Install JDK from http://www.oracle.com/technetwork/java/javase/downloads/index.html
2) set PATH=%PATH%;C:\Program Files\Java\<your JDK>\bin
3) cd to the directory where you saved Class12.java
4) javac com/schoolnova/it101/Class12.java
5) java -cp . com.schoolnova.it101.DOBCalculator