Java How to Program

Early Objects

TENTH EDITION

Paul Deitel & Harvey Deitel

Chapter 4 Control Statements: Part 1; Assignments, ++, -- Operator.

Algorithm & Pseudocode

Before written a program to solve a problem, you should have a through understanding of the problem and carefully planned approach to solving it.

Any problem can be executed through some of **actions** in some **Order**

These actions and order to solve a problem can be described in informal language which is the language of **Algorithm** and it's called **Pseudocode**, it's like your daily English or can be formed using your main language.

The **Pseudocode** does not execute on you computer it just language to describe how your **Algorithm** can be executed your program in the two main points **Actions** and **Order**, then you can write this in the programming language you need.

Control Structure

In general program is executed in some of **statements**, these **statements** are executed one after the other, and this process is called **Sequential execution**.

But java and other programming language help you to skip one or some of statements based on **Selection structure**, and it doing a process called **Transfer of Control**.

As well as java help you to transfer you program and skip some statements, it also help you to repeat some statements which is called **Repetition Structure**.

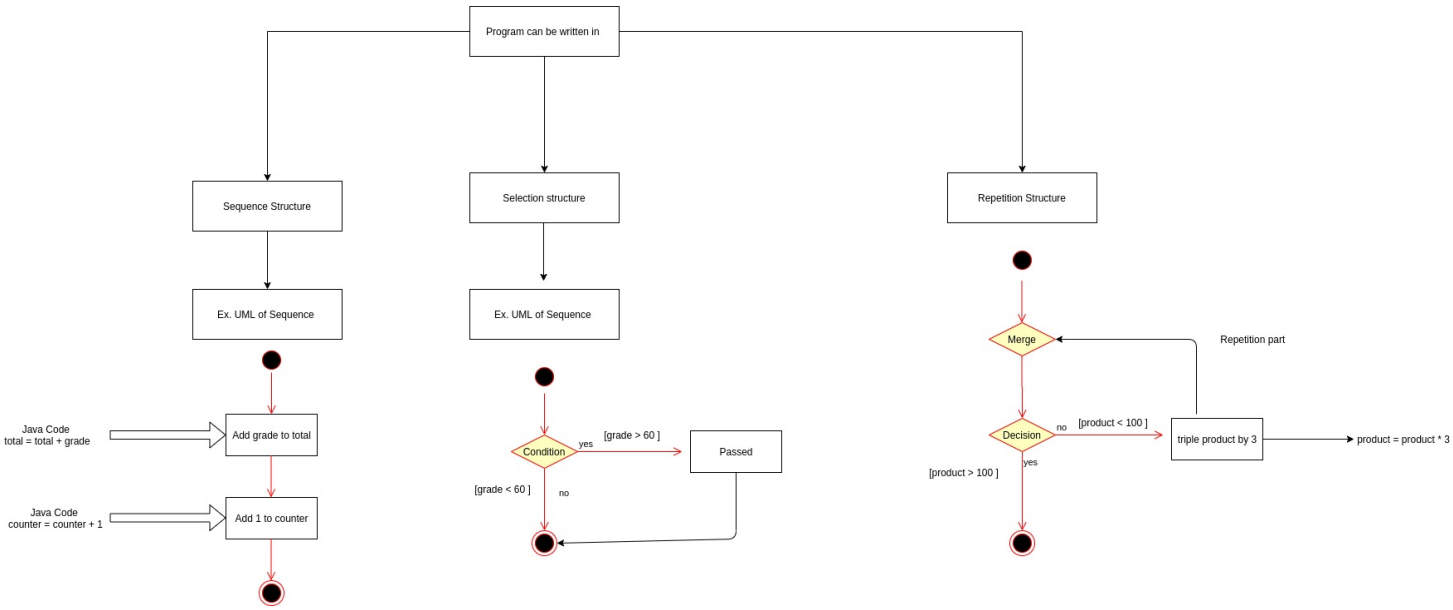
So program can be written using **3** main structure:

- Sequence Structure
- Selection structure
- Repetition Structure

The **Sequence Structure** as we said it executed on statement after the other to end of the program, but once there is **Selection structure**, it will transfer your program execution as the **Selection structure** in the program, and some of statements will be repeated as **Repetition Structure** required**.

Note !

We should replace **yes** and **no** in last graph of the loops.



UML Activity Diagram

Activity diagram is represent a portion of your program in the steps should be executed.

- Solid Circle is the start of the program called (initial statement).
- Solid Circle surrounded by a hollow circle called (final statement), the end of program.
- Rectangle, for action statement.
- Arrow, show how you should execute the program.
- some comments like **java code with arrows**.
- Diamond, or what we called descision symbol for **Selection**
- Guard Condition, which way the algorithm should take based on decision symbol.
- Merge and Condition the two are same using Diamond, or what we called decision symbol.

Selection Statements in java

The **Selection Statements** in java are three types:

- Signle-Selection just if ... statement
- Double-Selection if ... else
- Multiple-Selection like Switch

The **Single-Selection** is just, if true do something, or skip this statements or block of statements(body).

The **Double-Selection** is just, if true do something or block of statements(body)., else do another something or or block of statements(body).

The **Multiple-Selection** is to choose one of the things to do or block of things to do(body).

Repetition Statements in java

The **Repetition Statements** or **Iteration Statements** or **Looping Statements**, enable your program to repeat some something as your condition is true.

Also three statements for **Repetition Statements** which are:

- While
- Do-while
- For

The different is that **While and For** do the repetition **0 or more**, but **do ..while** do it **one or more**.

if Single-selection statement

Suppose you need to know if you just fail in the exam.

Pseudocode

if your grade less than 50

```
print "failed"
```

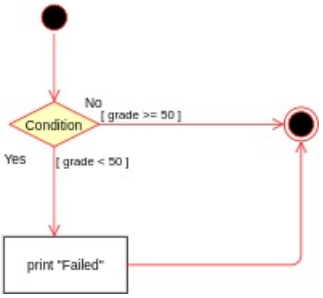
Now you are do Single-selection, once true do something otherwise will skip the print line., then the execution move to next statement in the sequence.

- Java corresponding code

if (grade < 50)

```
System.out.println("Failed");
```

- UML Graph



As you can see it print **Failed** when the condition is **True**, then go to final statement (terminate program). other wise terminate the program go to final statement.

if ... else Double-selection statement

Suppose you need to know if you pass **or** failed in the exam.

Pseudocode

if your grade less than 50

```
print "failed"
```

else

```
print "passed"
```

Now you are do Double-selection, once true do something else do another thing., then the execution move to next statement in the sequence.

- Java corresponding code

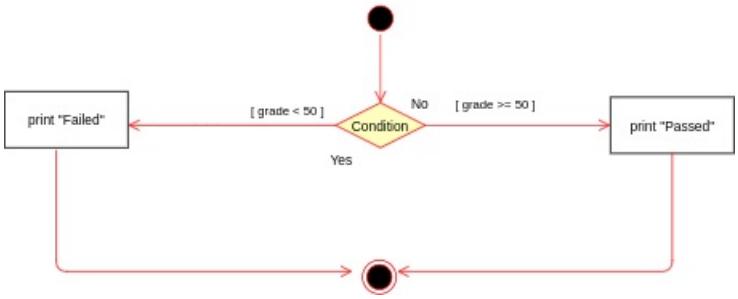
if (grade < 50)

```
System.out.println("Failed");
```

else

```
System.out.println("Passed");
```

- UML Graph



As you can see it print **Failed** when the condition is **True**, then go to final statement (terminate program). else ir print **Passed** then go to final statement (terminate program).

Nested if .. else statements (Multiple Selection) We can have if .. else statements inside another if .. else statements. Like if you need to see what grade from [A, B, C, D, F] you have get then. - Pseudocode if your grade greater than or equal to 90 print "A" else if grade greater than or equal to 80 print "B" else if grade greater than or equal to 70 print "C" else if grade greater than or equal to 60 print "D" else print "F" Now you are do multiple-selection, once true do something else do another thing., then the execution move to next statement in the sequence. - Java corresponding code if (grade >= 90) System.out.println("A"); else if (grade >= 80) System.out.println("B"); else if (grade >= 70) System.out.println("C"); else if (grade >= 60) System.out.println("D"); else System.out.println("F"); - Another good form if (grade >= 90) System.out.println("A"); else if (grade >= 80) System.out.println("B"); else if (grade >= 70) System.out.println("C"); else if (grade >= 60) System.out.println("D"); else System.out.println("F"); # Dangling-else problem if (x> 5) if(y > 5) System.out.println("X and Y are > 5"); else System.out.println("X is <= 5"); The else is associated with nested if(second if which is if y>5) even you have some indetation, java ignore these indentation unless you specify this in {} bracases, because java compiler always associates an else with immeditaly preceding if. In this case if x > 5 and y not > 5 it will print x <= 5. So like in this case we can do it like that: if (x> 5) { if(y > 5) System.out.println("X and Y are > 5"); } else System.out.println("X is <= 5");# Block of Code Each if or else or else if except that you have either one-line or block of lines, one line then no need to {}, but multiple line or block of code associated should have {}, and in general have {}. like this java code: if(grade >= 60) System.out.println("Passed"); else { System.out.println("Falied"); System.out.println("You should take this course again"); }

Condition Operation

As java provide you with **Unary** and **Binary** operator it also provide us with **ternary** operator, which is **?:** and it work with **Three operands, and write as follow:

- Before **?** should have a condation like :(grade >= 60). (first operand)
- After **?** Should have what you need in case of **True** like: System.out.println("Passed"); (second operand)
- After **:** Should have what you need in case of **False** like: System.out.println("Falied"); (third operand)
- Java Code

System.out.println(grade >= 60 ? "Passed" : ""Falied");

Student Class using if .. else statements

Class Student is a class that store instance variables student name and the average of the grades, provide methods to manipulate these instance variables.

Student class that contain:

```
Instance private Variables:
    - name: String
    - Average: double

Constructors:
    <<Constructor 1>> Student(name: String, average: double)

public Methods:
    + setName(name :String)
    + getName(): String
    + setAverage(average: double)
    + getAverage(): double
    + getBalance(): double
    + getLetterGrade(): char
```

Look at java apps direction for class Student and Student Test for the code, all of the code are documented.

```
Abdelrahman Rezk letter grade is: A
Abdallah Ezz letter grade is: C
```

While Repertition Statement

Repetition Statement (While)

One of the Repetition Statement is **while** which allow you to repeat some statement for **0** or **more**, depends on the state you in, while it still true the statements are executed.

The statement may be just a single statement or group of statements called **Block**, but it's common to use the body braces in general.

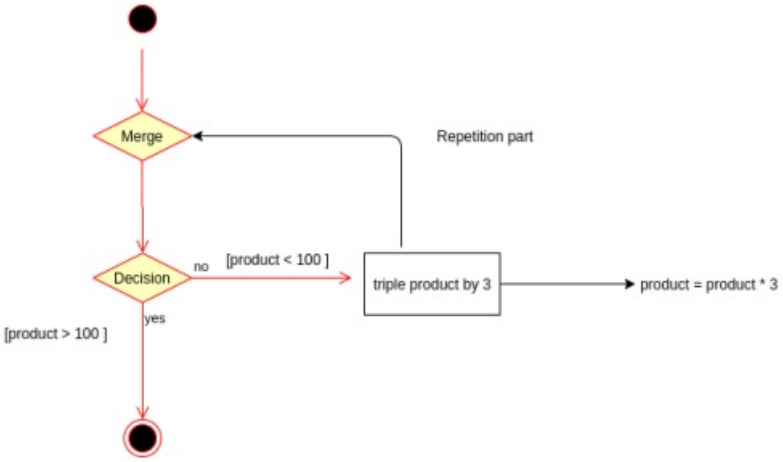
Take care of your condition and body that may cause some of logical error that make the loop infinite.

- Pseudocode While there are more items on my shopping list purchase next item and cross it off my list Java Code to get first power of 3 greater than or equal 100. int product = 100; while(product < 100){ product *= 3; }

Note !

We should replace **yes** and **no** in last graph of the loops.

Uml



We can distinguish between the decision and merge symbols in the incoming arrow the decision has just one transaction arrow incoming and two or more out from it., also, none of the transaction arrow associated with the merge symbol has a guard condition

Formulating Algorithm: Counter-Controlled Repetition

The most difficult part of solving a problem on a computer is developing the algorithm for the solution. Once a correct algorithm has been specified producing code is straightforward.

Let us discuss a problem of ten student took a quiz and you take a grade for each student from 0-100, then get the average on this quiz.

- Pseudocode Set total to zero Set grade counter to one While grade counter is less than or equal to ten Prompt the user to enter the next grade Add the grade to total Add one to counter Set average to the total divided by 10 print the average

```
To-Program/chapter_4/java_apps$ javac AverageQuiz.java
(base) abdelrahman@abdelrahman:/media/abdelrahman/SSD1/web deve
/
m/chapter_4/java_appsabdelrahman@abdelrahman:/media/abdelrahman
ment & cources/Courses&links&books/JAVA Mustafa saad/javabook/c
To-Program/chapter_4/java_apps$ java AverageQuiz
Enter Grade: 67
Enter Grade: 78
Enter Grade: 89
Enter Grade: 67
Enter Grade: 87
Enter Grade: 98
Enter Grade: 93
Enter Grade: 85
Enter Grade: 82
Enter Grade: 100
The total of the grades is: 846
The Average of the quiz is: 84(base) abdelrahman@abdelrahman:/m
SSD1/web development & courses/
```

Look at java apps direction for class AverageQuiz for the code, all of the code are documented.

Formulating Algorithm: Sentinel-Controlled Repetition

Instead of just 10 grades defined by your program, what if we need to read arbitrary numbers ? handle such logical error in previous Algorithm to see what is different ?

Sentinel Value or Single Value or Dummy Value or Flag Value, is a value that help you terminate the loop as in counter-contoller when value be larger than 10, but here we will assume value that not included in the user input which will flag of remaining out loop or be in.

Lets Go On

In this Algorithm we will using Top-down approach start with the top (single statement that cover the overall function of the program:

- Determine the Average of some quiz for students.

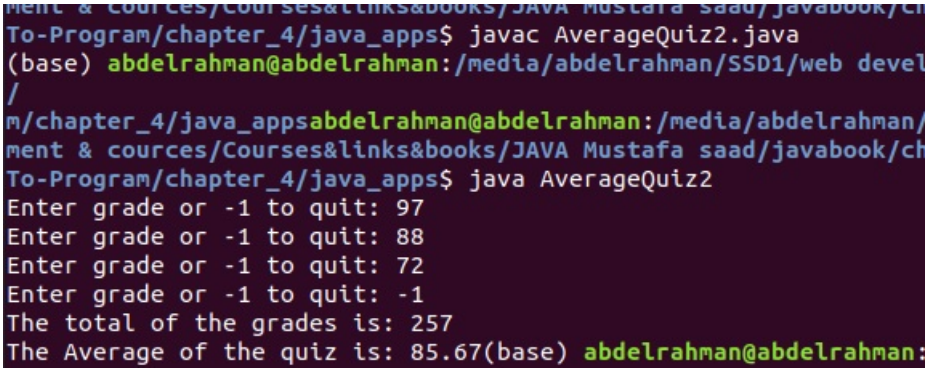
After the top which not have sufficient details what we need to write in the program, we go to the **refinement process**, by Divide the top into series of smaller task, in the order should be performed.

- Initialization Variables
- Input, sum and count the quize grades.
- Calculate and print the Average.

The **refinement** uses the sequence structure, steps that should be implemented in order one after another.

From **Top Statement** to **First Refinement**, we go in the top-down process to next level of refinement, by represent which variables we should use and name it how the input will be taken from the user and other specification to make the written program is straightforward as much I can.

- Pseudocode: just remove the statement of each phase - Initialization Variables - Initialize total to 0 - Initialize counter to 0 - Input, sum and count the quize grades. - Prompt the user to enter grade - Input the first grade - while the user has not enter the Sentinel Value - Add grade to total - Add one to counter - Prompt the user to enter grade - Input the next grade - Calculate and print the Average. - if counter is not equal to 0 - Set the Average to total divided by counter - print the Average - else print No grades have been entered



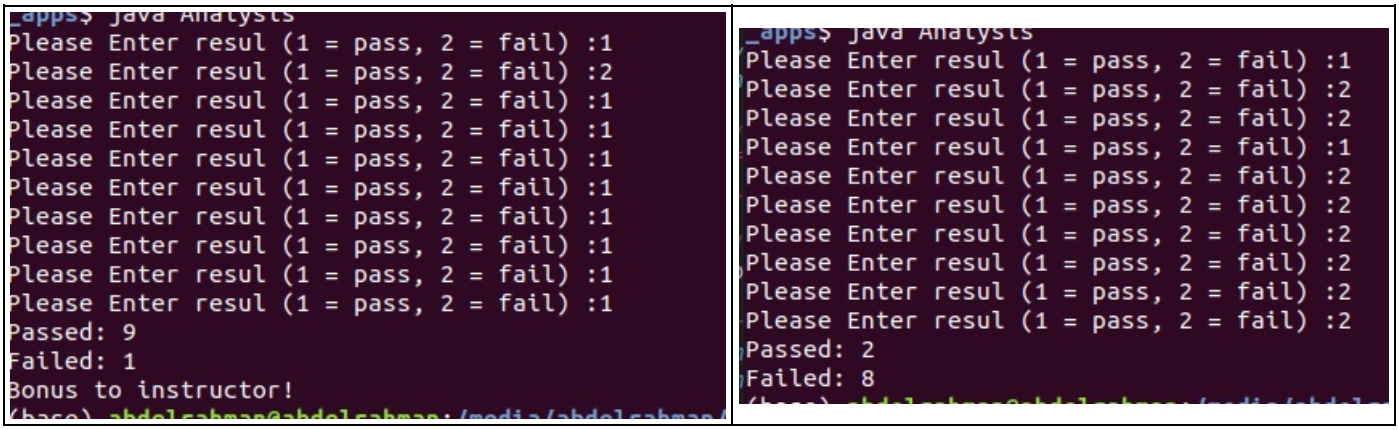
Look at java apps direction for class AverageQuiz2 for the code, all of the code are documented.

Formulating Algorithm: Nested Control Statements

Consider the problem of ten students have an exam and you need to know the number of students passed the exam and the number of who are not, the passed exam of some student entered as 1 and not passed entered as 2, and if the number of passed the exam is greater than 8 give bouns to the lecturer of the course.

Lets devide the problem into top-down and the three phases we mentioned above.

- structure of the program Top-Sentence - Analyze exam result and decide whether a bonus should be paid. The first refinement - Initiation of Variables - Procssing 10 exam result, and count passed or not - Print result of passed and not and if there is a bouns - Pseudocode Initialize passes to zero Initialize failures to zero Initialize student counter to one While student counter is less than or equal to 10 Prompt the user to enter the next exam result Input the next exam result If the student passed Add one to passes Else Add one to failures Add one to student counter Print the number of passes Print the number of failures If more than eight students passed Print “Bonus to instructor!”



Look at java apps direction for class *Analysis* for the code, all of the code are documented.

Compound Assignment Operators

c = c + 1; can be written as c +=1;

Add Value of right side to the variable on the left, then assign the result in the variable on the left.

Increment and Decrement Operators

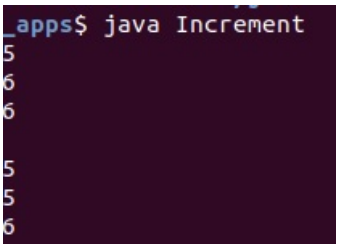
instead of c = c+1; or c +=1;

Unary Operators

We can use Unary to add 1 to variable that we need to increase or decrease by only 1.

- ++a Pre-increment (add 1 to a before use it).
- --a Pre-decrement (subtrict 1 from a before use it).
- a++ Post-increment ("Use the current value of a in the expressionin which a resides, then incrementa by 1".).
- a-- Post-decrement ("Use the current value of a in the expressionin which a resides, then incrementa by 1).

Look at java apps direction for class *Increment* for the code, all of the code are documented.



Note !

When incrementing or decrementing a variable in a statement by itself, the prefixincrement and postfix increment forms have the same effect, and the prefix decrement andpostfix decrement forms have the same effect.It’s only when a variable appears in the con-text of a larger expression that preincrementing and postincrementing the variable havedifferent effects (and similarly for predecrementing and postdecrementing).

"Attempting to use the increment or decrement operator on an expression other than one towhich a value can be assigned is a syntax error. For example, writing++(x + 1)is a syntaxerror, because(x + 1)is not a variable."

Operator Precedence and Associativity

Operators						Associativity	Type
++	--					right to left	unary postfix
++	--	+	-	(*type*)		right to left	unary prefix
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment

Fig. 4.16 | Precedence and associativity of the operators discussed so far.

Optional GUI see java codes direction