National University

Faculty of Computer Since and Technology

Lec 10
Object Oriented Programming

Control Statements
Part 2

while Repetition Statement

- Repetition statement—repeats an action while a condition remains true.
- Pseudocode

While there are more items on my shopping list
    Purchase next item and cross it off my list

- The repetition statement's body may be a single statement or a block.
- Eventually, the condition will become false. At this point, the repetition terminates, and the first statement after the repetition statement executes.

- Example of Java's while repetition statement: find the first power of 3 larger than 100. Assume int variable product is initialized to 3.

```
while( product <= 100)
product = 3* product;
```

- Each iteration multiplies product by 3, so product takes on the values 9, 27, 81 and 243 successively.
- When variable product becomes 243, the while-statement condition—product<=100—becomes false.
- Repetition terminates. The final value of product is 243.
- Program execution continues with the next statement after the while statement.

- The UML activity diagram in Fig.4.4 illustrates the flow of control in the preceding while statement.
- The UML represents both the merge symbol and the decision symbol as diamonds.
- The merge symbol joins two flows of activity into one.

- The decision and merge symbols can be distinguished by the number of incoming‖ and outgoing‖ transition arrows.
- A decision symbol has one transition arrow pointing to the diamond and two or more pointing out from it to indicate possible transitions from that point. Each transition arrow pointing out of a decision symbol has a guard condition next to it.
- A merge symbol has two or more transition arrows pointing to the diamond and only one pointing from the diamond, to indicate multiple activity flows merging to continue the activity. None of the transition arrows associated with a merge symbol has a guard condition.

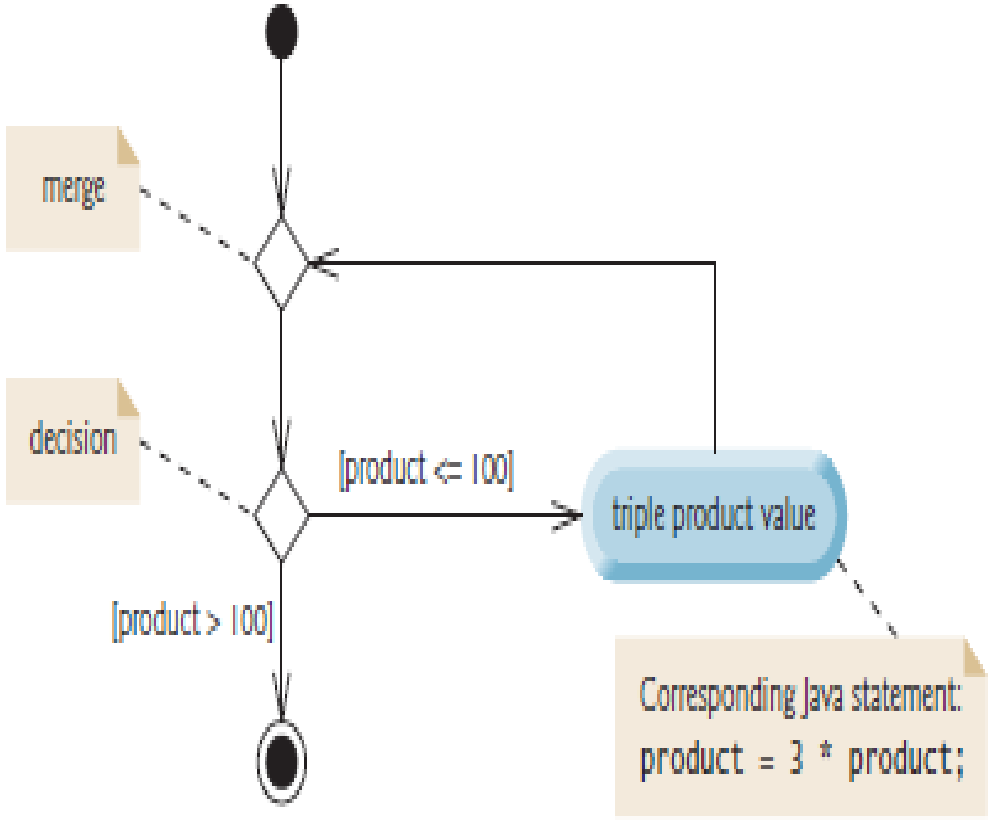

Fig. 4.4 | `while` repetition statement UML activity diagram.

Formulating Algorithms: Counter-Controlled Repetition

- *A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz.*
- The class average is equal to the sum of the grades divided by the number of students.
- The algorithm for solving this problem on a computer must input each grade, keep track of the total of all grades input, perform the averaging calculation and print the result.

- Use counter-controlled repetition to input the grades one at a time.
- A variable called a counter(or control variable) controls the number of times a set of statements will execute.
- Counter-controlled repetition is often called definite repetition, because the number of repetitions is known before the loop begins executing.

- A total is a variable used to accumulate the sum of several values.
- A counter is a variable used to count.
- Variables used to store totals are normally initialized to zero before being used in a program.

```
1  Set total to zero
2  Set grade counter to one
3
4  While grade counter is less than or equal to ten
5      Prompt the user to enter the next grade
6      Input the next grade
7      Add the grade into the total
8      Add one to the grade counter
9
10 Set the class average to the total divided by ten
11 Print the class average
```

Fig. 4.5 | Pseudocode algorithm that uses counter-controlled repetition to solve the class-average problem.

Formulating Algorithms: Sentinel-Controlled Repetition

- *Develop a class-averaging program that processes grades for an arbitrary number of students each time it's run.*
- Sentinel-controlled repetition is often called indefinite repetition because the number of repetitions is not known before the loop begins executing.
- A special value called a sentinel value(also called a signal value, a dummy value or a flag value) can be used to indicate ―end of data entry.‖
- A sentinel value must be chosen that cannot be confused with an acceptable input value.

Developing the Pseudocode Algorithm with Top-Down, Stepwise Refinement

- Top-down, stepwise refinement
- Begin with a pseudocode representation of the top—a single statement that conveys the overall function of the program:
  - Determine the class average for the quiz
- The top is a complete representation of a program. Rarely conveys sufficient detail from which to write a Java program.

- Divide the top into a series of smaller tasks and list these in the order in which they'll be performed.
- First refinement:
  - Initialize variables
  - Input, sum and count the quiz grades
  - Calculate and print the class average
- This refinement uses only the sequence structure—the steps listed should execute in order, one after the other.

- Second refinement: commit to specific variables.
- The pseudocode statement Initialize variables can be refined as follows:
  - Initialize total to zero
  - Initialize counter to zero

- The pseudocode statement
  - Input, sum and count the quiz grades
- requires a repetition structure that successively inputs each grade.
- We do not know in advance how many grades are to be processed, so we'll use sentinel-controlled repetition

- The second refinement of the preceding pseudocode statement is then:
  - *Prompt the user to enter the first grade*
  - *Input the first grade (possibly the sentinel)*
  - *While the user has not yet entered the sentinel*
    - *Add this grade into the running total*
    - *Add one to the grade counter*
    - *Prompt the user to enter the next grade*
    - *Input the next grade (possibly the sentinel)*

```
1  Initialize total to zero
2  Initialize counter to zero
3
4  Prompt the user to enter the first grade
5  Input the first grade (possibly the sentinel)
6
7  While the user has not yet entered the sentinel
8      Add this grade into the running total
9      Add one to the grade counter
10     Prompt the user to enter the next grade
11     Input the next grade (possibly the sentinel)
12
13 If the counter is not equal to zero
14     Set the average to the total divided by the counter
15     Print the average
16 else
17     Print "No grades were entered"
```

Fig. 4.8 | Class-average problem pseudocode algorithm with sentinel-controlled repetition.

Compound Assignment Operators

- Compound assignment operators abbreviate assignment expressions.
- Statements like

variable = variable operator expression;

where operator is one of the binary operators +, -, *, /or % can be written in the form

variable operator= expression;

- Example:

c = c + 3;

can be written with the addition compound assignment operator, +=, as

c += 3;

- The +=operator adds the value of the expression on its right to the value of the variable on its left and stores the result in the variable on the left of the operator.

Increment and Decrement Operators

- Unary increment operator, ++, adds one to its operand
- Unary decrement operator, --, subtracts one from its operand
- An increment or decrement operator that is prefixed to (placed before) a variable is referred to as the prefix increment or prefix decrement operator, respectively.
- An increment or decrement operator that is postfixed to (placed after) a variable is referred to as the postfix increment or postfix decrement operator, respectively.

Operator	Operator name	Sample expression	Explanation
++	prefix increment	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postfix increment	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	prefix decrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postfix decrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

Fig. 4.14 | Increment and decrement operators.