

How to **THINK** like a Programmer

Problem Solving for the Bewildered

paul vickers



Problem Solving 9

Determinate Loops

Aim



- ▶ In this lesson we will continue our study of looping control structures
 - ▶ Determinate loops
 - ▶ Counter-controlled iteration using `while` and `for`
 - ▶ Solving a problem involving counter-controlled iteration using the problem-solving strategy.

Determinate loops



- ▶ Recall that **determinate loops** are ones where we can determine in advance the number of times the loop will iterate.
- ▶ Many programming situations require the use of such loops.
 - ◉ An algorithm that reports the average rainfall for each month of the year could use a determinate loop: there are 12 months, so the loop will iterate 12 times
 - ◉ An algorithm that calculates the number of leap years between two years chosen by the user would also use a determinate loop. The years are not known in advance, but once the user has selected the start and end years, the number of iterations of the loop to process them can be calculated

Example of a Determinate Loop



- ▶ Imagine the simple case of adding sugar to tea.
- ▶ In this case, once we determine how many sugars are required, we continually add spoonfuls until we have reached the upper limit.
 1. Initialise sugarsAdded to 0
 2. Find out how many sugars are required
 3. while (sugarsAdded < sugarsRequired)
 - 3.1 Add a spoonful of sugar
 - 3.2 Set sugarsAdded to sugarsAdded + 1
 - endwhile

Counter-controlled Iteration



- ▶ The example we have just seen is said to be a **counter-controlled loop** because a **counter variable** is used to determine when to terminate. It is `sugarAdded` in this case and its value changes as the loop iterates.
- ▶ Such loops follow a general pattern:

```
initialise counter to some starting value
while (counter < some finishing value)
    Actions for loop body
    Add increment to counter
endwhile
```

Counter-controlled Iteration



- ▶ Because counter-controlled loops are so common Most languages have a dedicated iteration structure for them which is normally **more efficient** than the equivalent while loop version.

Counter variable Counter starting value Counter finishing value Increment value

for variable goes from initial value to final value in steps of step value

Actions for loop body

endfor

- ▶ It corresponds to a **for-loop** in many languages including Java.

Operation of for Loop



- ▶ The starting value of the counter variable is set to the value of initial value. Note that **this only happens once, before the first execution of the loop.**
- ▶ Then the value of the **counter variable is compared to the value of final value.** Assuming final value is bigger than initial value (it doesn't have to be but normally is), then the test will evaluate to true and the actions in the loop body will execute.
- ▶ After the last action in the loop body, the **value of the counter variable will automatically increase or decrease** depending on the value of step value.
- ▶ Most often the step value is 1.

Operation of for Loop



- ▶ When the counter variable has been updated, it is compared to the value of final value once more and this process continues as long as final value has not been reached.

- ▶ Here's an example

for month goes from 1 to 12 in steps of 1

Prompt the user for the month's rainfall

Read in the month's rainfall

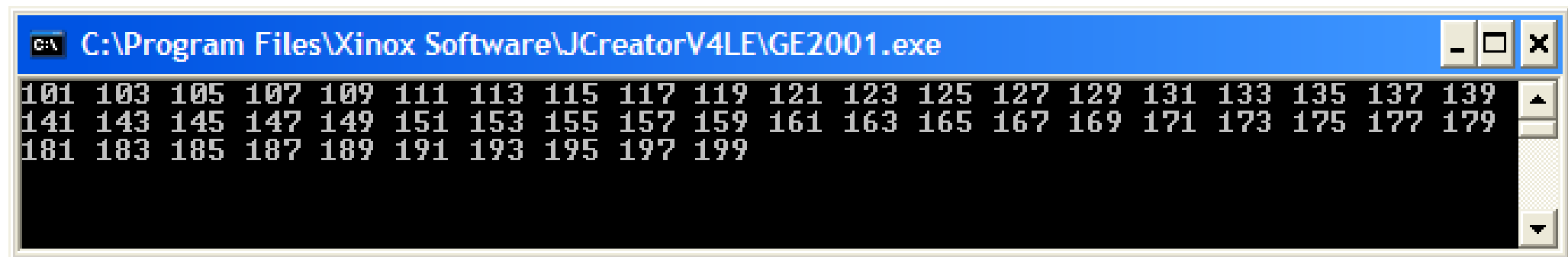
Set totalRainfall to totalRainfall + monthRainfall

endfor

- this loop adds up all the monthly rainfall values for the 12 months

ACTIVITY

Write a pseudocode solution which determines and displays all the odd numbers between 100 and 200. Then write the Java program based on the pseudocode



```
C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe
101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139
141 143 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179
181 183 185 187 189 191 193 195 197 199
```

Solution



```
1. for i goes from 100 to 200 in steps of 1
    1.1 if (i mod 2 ≠ 0)
        1.1.1 Display i
    endif
endfor
```

ACTIVITY

Write a pseudocode solution which displays every 10th number from 300 down to 100 in reverse order i.e.

300
290
280
:
110
100

Solution



```
1. for i goes from 300 to 100 in steps of -10
    1.1 Display i
endfor
```

Counter-controlled Looping - understanding the problem



- ▶ Imagine that we need to write an algorithm that involves **determining whether a user-supplied value is an integer or not** and displaying the outcome.
- ▶ Note that this problem statement is very short but tells us one important thing.
 - ▶ We know that we have to check to see if the value inputted is an integer
- ▶ So the first question is, what is an integer? – we cannot attempt the problem without knowing this.
- ▶ We should know that an integer is a **positive or negative whole number**.

Counter-controlled Looping - devising the plan



- ▶ A good way to approach this type of problem is to give it some thought. Think about what is being asked for again – “determine if a user-supplied value is an integer”.
- ▶ In this type of question you should **think about what the possible input values could be** and work from there
 - ▶ the value could be textual such as “Microsoft”, “bears”, “123xyz”, “%\$72”
 - ▶ the value could be floating-point such as 34.56, -21.89, 90876.56
 - ▶ the value could be an integer such as 9878, 2008, -567, -2345.67
- ▶ So the question is – how do valid integer values differ from other forms of input values.
- ▶ Notice that if we are dealing with a valid integer then it **can only begin with a digit or a minus** and **all the remaining characters must be digits from 0-9 inclusive**.

Counter-controlled Looping – devising the plan



- ▶ So how do we check whether the characters conform to a valid integer? Each one will have to be checked in turn. Notice that this checking is a **repetitive action** as each character undergoes the same test (except the first character which is also allowed to be a minus). Therefore a loop could be used for all the characters from position 2 onwards.
- ▶ Is the loop determinate or indeterminate? Here, once the user has supplied their value, it will contain a certain number of characters. Therefore, we know in advance of the loop executing how many characters need to be tested, so we can use a **determinate loop** here.
- ▶ We have a choice as to whether to use a while loop or a for loop but because **for loops are considered more efficient** we will go with this option here.

Counter-controlled Looping – devising the plan



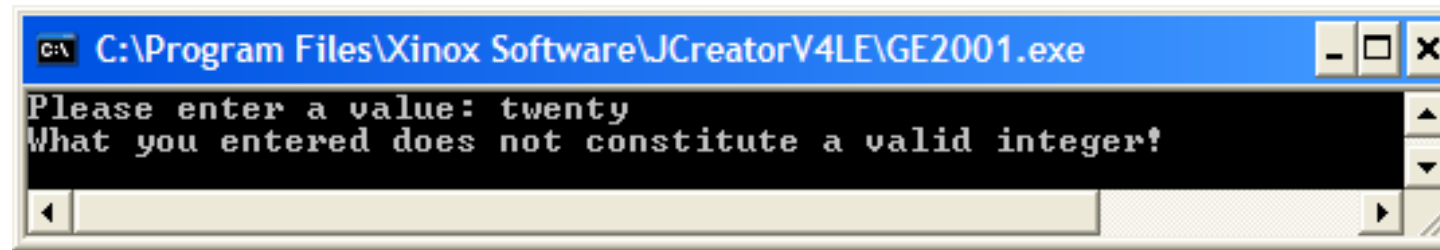
- ▶ We have figured out that we need a loop for testing all the characters from position 2 onwards. What about the first character? We should simply test whether the first character is a digit or a minus. If it is not, then we should not process the rest of the characters, since it cannot be an integer. If it is, we must progress to checking at least the next character.
- ▶ Note that for efficiency purposes, our **loop should stop as soon as we find a character that is not a digit**. This can be achieved by just **breaking out of the loop** if this condition arises.
- ▶ If our loop completes fully then the **value of the loop counter must be 1 more than the number of characters in the value entered by the user**. In any other case, the value of the loop counter will be less than this number. We can use this test to verify a valid integer.

Counter-controlled Looping executing the plan

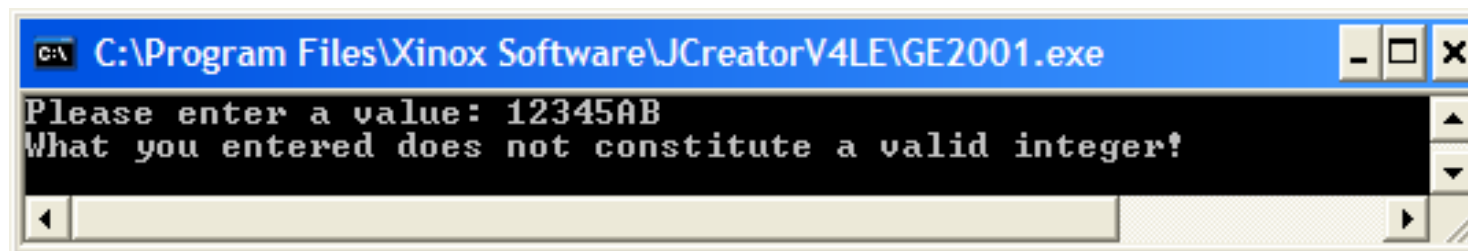


- ▶ Our final pseudocode draft here might be:
 - 1. Prompt for the value
 - 2. Read in the value
 - 3. if (the first character is a digit or a minus)
 - 3.1 for characterPosition goes from 2 to (number of characters in the value) in steps of 1
 - 3.1.1 if character at characterPosition is not a digit
 - 3.1.1.2 Break out of the loop
 - endif
 - endfor
 - 3.2 if (characterPosition is equal to number of characters in the value + 1)
 - 3.2.1 Display “Valid Integer” message
 - otherwise
 - 3.2.2 Display “Invalid integer” message
 - endif
 - otherwise
 - 3.3 Display “invalid integer” message
 - endif
- ▶ Now you should assess the solution and make sure it works according to plan – again plug in values to check it out.

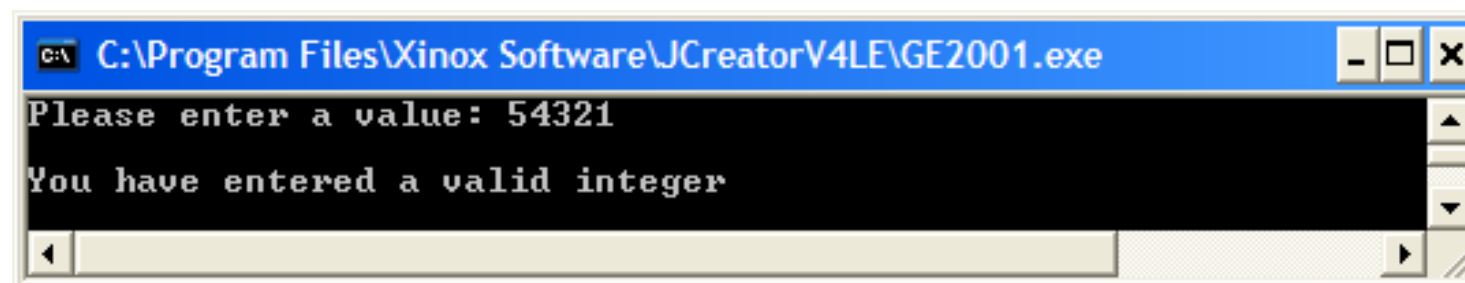
Now you should write the Java program based on the above pseudocode solution. Your program should run as indicated in the following sample runs:



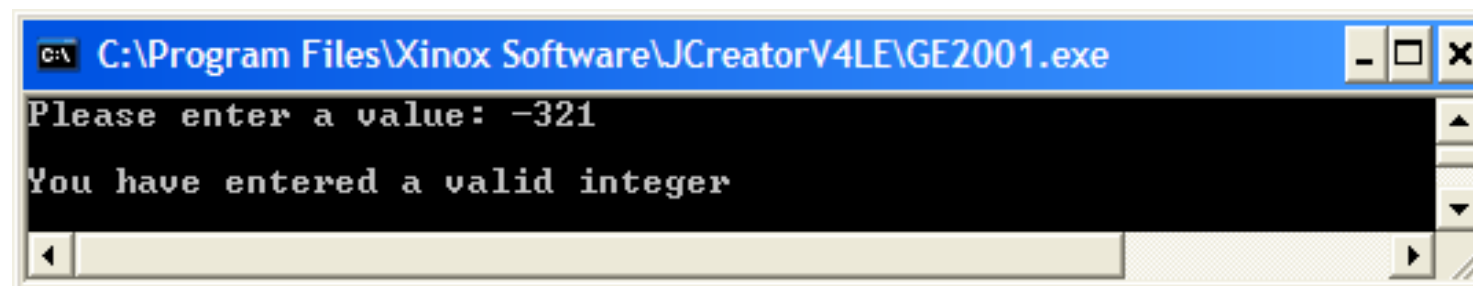
```
C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe
Please enter a value: twenty
What you entered does not constitute a valid integer!
```



```
C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe
Please enter a value: 12345AB
What you entered does not constitute a valid integer!
```



```
C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe
Please enter a value: 54321
You have entered a valid integer
```



```
C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe
Please enter a value: -321
You have entered a valid integer
```