How to THINK like a Programmer

Problem Solving for the Bewildered

paul vickers

Problem Solving 3

Introducing pseudocode

John Brosnan,
Computing Dept, ITT

Aim of Lesson

- The aim of this lesson is to introduce the semi-formal notation called pseudocode, that we will use to formulate algorithms in this module.

Computer language

- For computers to ‘understand’ us the words we use must carry the whole meaning, we cannot be ambiguous.

- Programmers must express themselves concisely and clearly:
  - computers will not interpret what we say or try to infer what we mean
  - we must tell them precisely what we want them to do
  - computers will follow your instructions to the letter **even if the instructions are wrong**

Pseudocode: a language for description

- We will use pseudocode for writing solutions to problems
- Pseudocode has some formal rules which make it unambiguous
- It is half-way between loose natural language and the very logically precise real computer programming languages
- We have used it very informally so far but we will use it more precisely from now on
  - Many programmers use pseudocode to express their ideas in a structured way without the complexities of a real programming language

Pseudocode example

- A pseudocode description for the process of adding 2 user-supplied numbers and displaying their sum would be

```
1. Prompt for the first number
2. Read in the first number
3. Prompt for the second number
4. Read in the second number
5. Add the first number and second number
6. Display the sum
```

* There are 6 distinct actions here

* The actions are the instructions that should be executed
* By numbering and putting each action on a separate line the sequence of actions is very clear.

Importance of Sequence

- The idea of **sequence** (the order in which actions are carried out) is a vitally important programming concept
  - **Things must be done in the right order to get the right results,** for example, a cake must not be put in the oven before the ingredients have been mixed
  - In the adding 2 numbers example, does order matter at any stage?
- Sometimes the ordering of certain actions is unimportant - does it matter whether I put my left shoe on before my right shoe? Does it matter, when subtracting 2 numbers, which number I enter first?

ACTIVITY

Now consider the following problem description and attempt to apply pseudocode to it yourself. When you have the pseudocode written see if you can translate it into a Java program:

Write a program which asks the user for 3 numbers and displays the product of these to 3 decimal places.

Possible Pseudocode Solution

```
1. Prompt for the first number
2. Read in the first number
3. Prompt for the second number
4. Read in the second number
5. Prompt for the third number
6. Read in the third number
7. Multiply the three numbers together
8. Display the product to 3 decimal places
```

Problem-Solving Example

- Consider having to write a program which asks the user for a number of ounces, converts that to pounds and ounces and displays the result.
(N.B. 1 pound = 16 ounces)

- Applying the problem-solving strategy, start by trying to understand the problem fully.
  - What do we know? That 16 ounces = 1 pound
  - What do we want to know? How many pounds and ounces is a certain number of ounces equal to?
  - Does the problem have subproblems? Possibly, not sure yet.
  - Can restating the problem help? Think about the output here, with a few examples: if I have 20 ounces then that should produce as output 1 pound and 4 ounces, if I have 42 ounces?

Problem-Solving Example

- Once we understand what the program should do, we can attempt to devise a plan to solve the problem.
  - We need to identify all subproblems and deal with these individually. Note from the last part, 20 ounces became 1 pound and 4 ounces, 42 ounces became 2 pounds and 10 ounces.
  - Note that we have 2 individual subproblems here really, from the original quantity of ounces, we must find (i) the total number of whole pounds and (ii) the remaining ounces.
  - These will be determined separately.
  - We will try to find the total number of whole pounds first – how did we arrive at our values for pounds earlier? By simply dividing the original ounces value by 16 (since 1 pound = 16 ounces) and “throwing away” anything after the decimal point.
  - To determine the remaining ounces we can use modulus 16 arithmetic on the original ounces value.

Problem-Solving Example

- Next we need to execute the plan which just means writing down the sequence of actions, in the correct order which will lead to the problems solution. Here, this is:

```
1. Prompt for the original amount in ounces
2. Read in the original amount in ounces
3. Calculate the whole pounds by dividing the
original ounces by 16 and ignoring any
fractional part
4. Calculate the remaining ounces by using
modulus 16 arithmetic on the original ounces
5. Display the result
```

Problem-Solving Example

- The next stage in the problem-solving strategy is to look back and reflect by assessing the result. All this involves here is plugging in some numbers to our pseudocode solution to check that it works properly and efficiently.

Say total ounces = 23 =>

pounds = 23/16 = 1.4375. Drop the fractional part => pounds = 1 (correct)
remaining ounces = 23 mod 16 = 7 (correct)

Say total ounces = 56 =>

pounds = 56/16 = 3.5. Drop the fractional part => pounds = 3 (correct)
remaining ounces = 56 mod 16 = 8 (correct)

Problem-Solving Example

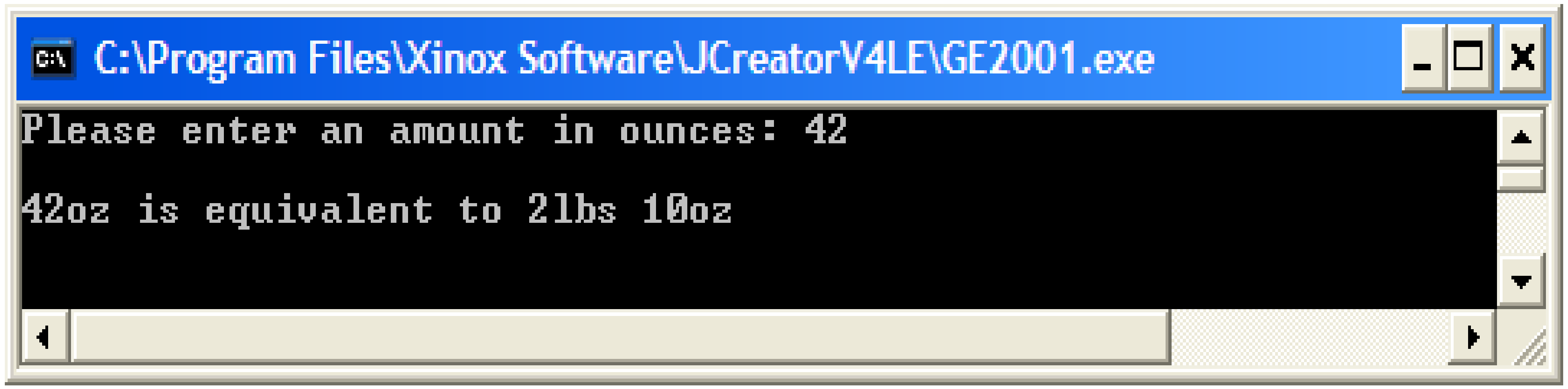
- The final part in the problem-solving strategy is documenting the solution. In this case, the following would suffice:

  "This pseudocode outlines a method for reading in a quantity in ounces and converting that to its equivalent in pounds and ounces."

  Now anyone reading your pseudocode will be able to see immediately what its purpose is.

ACTIVITY

Now you should try to write a Java program for the problem just solved.
It might run as follows (assume a whole number of ounces here)



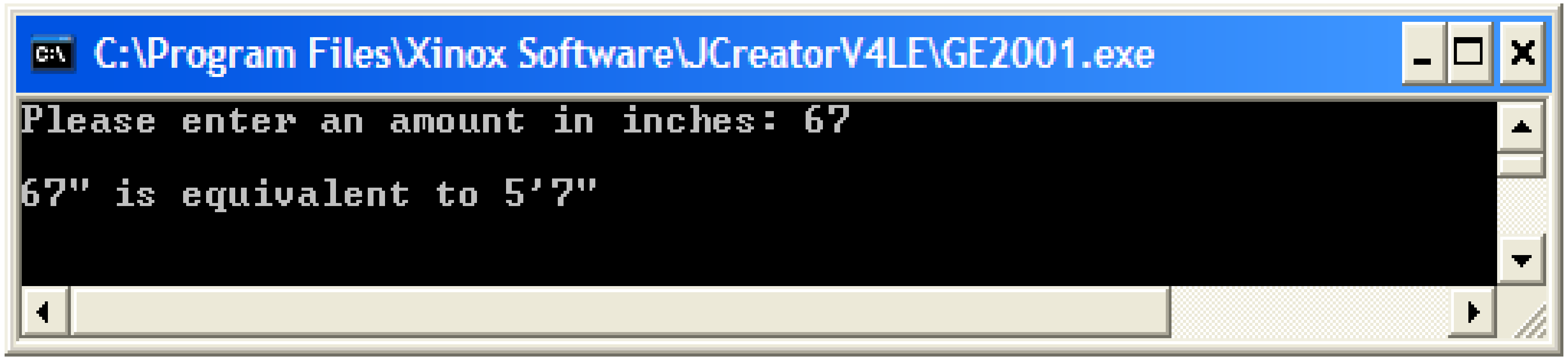
```
Please enter an amount in ounces: 42

42oz is equivalent to 2lbs 10oz
```

ACTIVITY

Now consider the following problem description and attempt to apply pseudocode to it yourself. When you have the pseudocode written see if you can translate it into a Java program:

Write a program which asks the user for a whole number quantity in inches and displays to the screen its equivalent in feet and inches.

N.B. 1 foot = 12 inches



```
Please enter an amount in inches: 67

67" is equivalent to 5’7"
```