How to THINK like a Programmer

Problem Solving for the Bewildered

paul vickers

Problem Solving 4

Making Choices

John Brosnan,
Computing Dept, ITT

Aim of Lesson

- The aim of this lesson is to apply the problem-solving strategy to problems that involve making choices.
  - Simple selection with the `if` structure
  - Extended selection with the `if-else` structure

Making choices

- Our problem-solving strategy asks whether all actions should be carried out in every circumstance, or should some actions/groups of actions be carried out only when certain conditions are met?
- It is very common for solutions to problems to include some decision-making. Consider again the problem of reading in 2 user-supplied numbers and determining their sum.
  - Recall our original pseudocode solution that involved 6 steps (can you recall them now?)
- One assumption that was made in this solution was that the user definitely supplied valid numbers. But what if they did not?

The problem with invalid input

- In some programming languages, the program will crash at runtime if the user supplies an invalid value for a numerical variable. In others, such as Just BASIC, the value will be accepted and the variable set to zero, leading to a logical error.

- Our aim is always to avoid runtime and logical errors so it is always good programming practice to make sure such invalid values are not accepted.

Solution as a sequence

A pseudocode description for the process of adding 2 user-supplied numbers and displaying their sum in sequential form is:

```
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
```

- This solution is valid as long as the user enters

valid numbers. To guarantee a valid solution requires a slightly different solution ……

Solution with Decisions

```
1. Prompt for the first number
2. Read in the first number
3. if (first number is valid)
       3.1 Prompt for the second number
       3.2 Read in the second number
       3.3 if (second number is valid)
             3.3.1 Add the first number and
                   second number
             3.3.2 Display the sum
           endif
   endif
```

- **Notice now that certain instructions will only execute provided that previous conditions have been satisfied.**
- **There is no point in reading in the second number unless the first number is valid. Likewise there is no point in trying to sum the numbers if either is invalid.**

The `if` construct

- We show decisions with the `if` selection construct

```
if (condition)
    action 1
    action 2
endif
action 3
```

Here, actions 1 and 2 are only carried out when the test condition evaluates to true. Action 3 gets executed irrespective of whether the condition evaluates to true or false. `endif` marks the end of the `if` structure.

- All programming languages include an if like structure to allow programs to make simple decisions.

ACTIVITY

Examine the following algorithm fragment:

```
put on hat
if (weather is sunny)
    put on sunglasses
endif
put on shoes
```

What items of clothing will be put on a) when it is raining and b) when it is sunny?

Now examine the following algorithm fragment:

```
go to shop
buy milk
if (today is Saturday)
    buy weekly newspaper
    buy peanuts
endif
buy bread
```

What items will be purchased a) on Thursday, b) on Saturday?

ACTIVITY

In the Stocksfield Diner customers can choose whether or not to have their hamburgers plain or with any combination of cheese, lettuce and tomatoes. Write three if constructs that add as required by the customer.

Solution

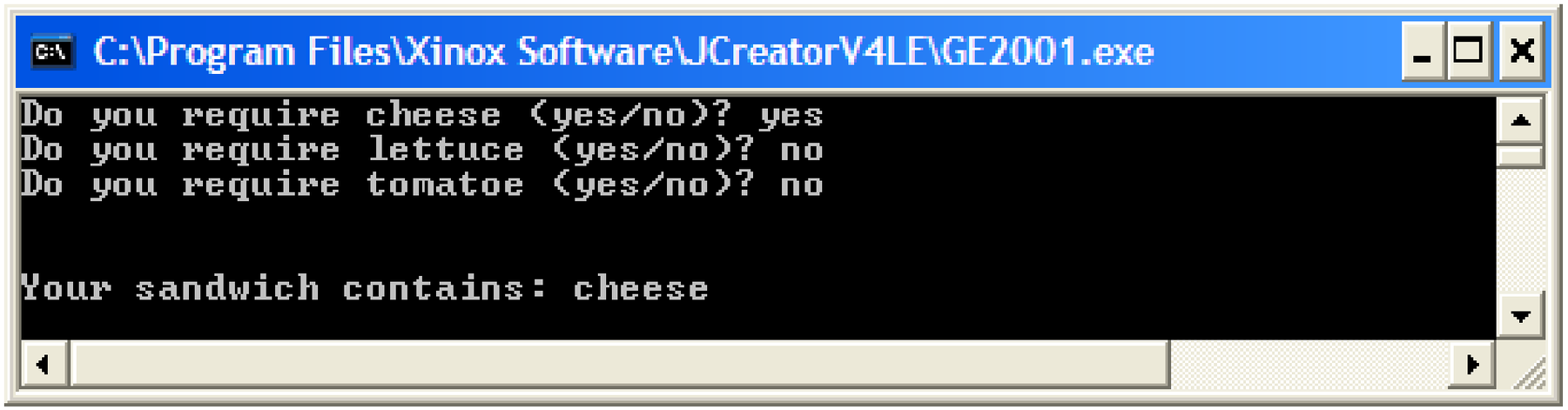
Pseudocode for adding cheese, lettuce and tomatoes to burger:

```
get cheese, lettuce, and tomatoes requirements
if (cheese required)
    add cheese
endif
if (lettuce required)
    add lettuce
endif
if (tomatoes required)
    add tomatoes
endif
```

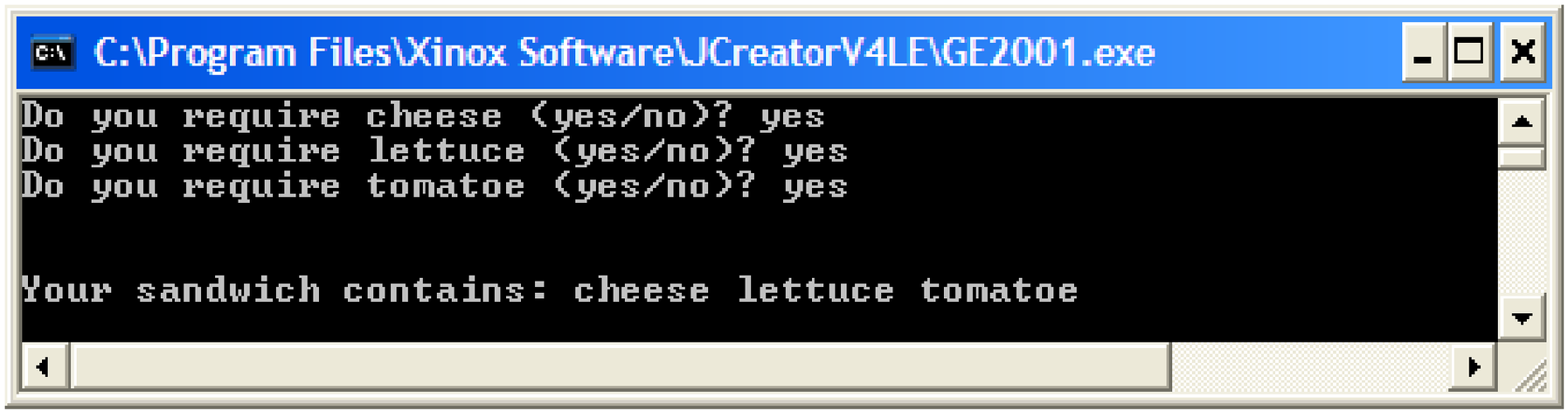
Now you should try to code the Java solution to this problem

Note that the output from the program depends on the choices made:

1st Run of the Program



2nd Run of the Program



ACTIVITY

The check digit is the last digit in a books ISBN and is calculated by a modulus 11 technique using the weights 10 to 2. This means that each of the first nine digits is multiplied by a number in a sequence from 10 to 2. If you add these products together and then add the value of the check digit, this total sum should be divisible by 11 without leaving a remainder. If it does give a remainder then the check digit does not match the rest of the number and we know the ISBN has been copied down incorrectly. The check digit is calculated in the following manner:

Multiply the first nine digits by 10, 9, 8, 7, . . ., 2 respectively and add the results. Divide this sum by 11 and take the remainder. Finally, subtract this remainder from 11 to give the check digit. If the value is 10 the check digit becomes “X.” For example, we can validate the ISBN 0-14-012499-3 as follows:

Check digit = (0×10)+(1×9)+(4×8)+(0×7)+(1×6)+(2×5)+(4×4)+(9×3)+(9×2) ÷ 11
= 118÷11 = 10, remainder 8
So, the check digit is 11 - 8 = 3 meaning this must be a valid ISBN

Solution

Pseudocode for determining the validity of an ISBN

1. Multiply first digit by 10 and add result to total
2. Multiply second digit by 9 and add result to total
3. Multiply third digit by 8 and add result to total
4. Multiply fourth digit by 7 and add result to total
5. Multiply fifth digit by 6 and add result to total
6. Multiply sixth digit by 5 and add result to total
7. Multiply seventh digit by 4 and add result to total
8. Multiply eighth digit by 3 and add result to total
9. Multiply ninth digit by 2 and add result to total
10. Divide total by 11 and take the remainder
11. if (11 - remainder equals check digit)

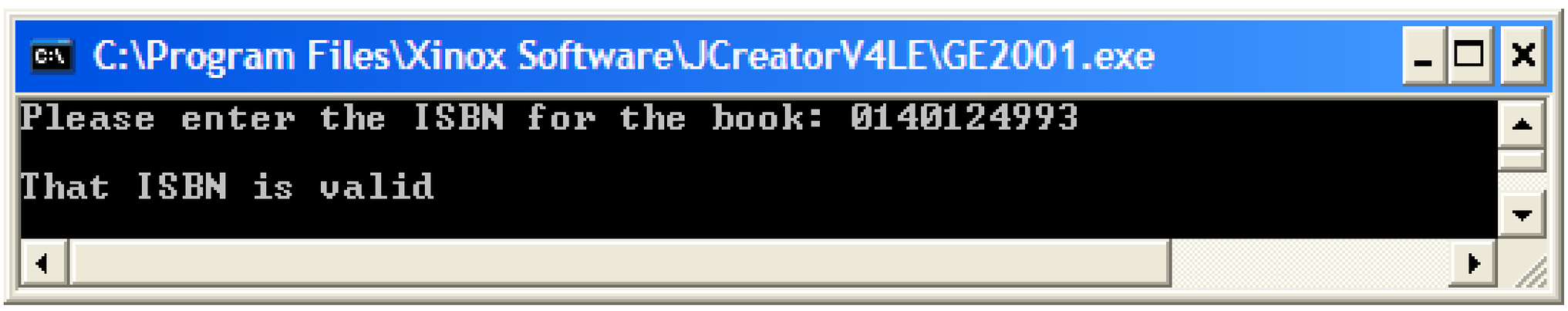
        11.1. Display 'ISBN valid'

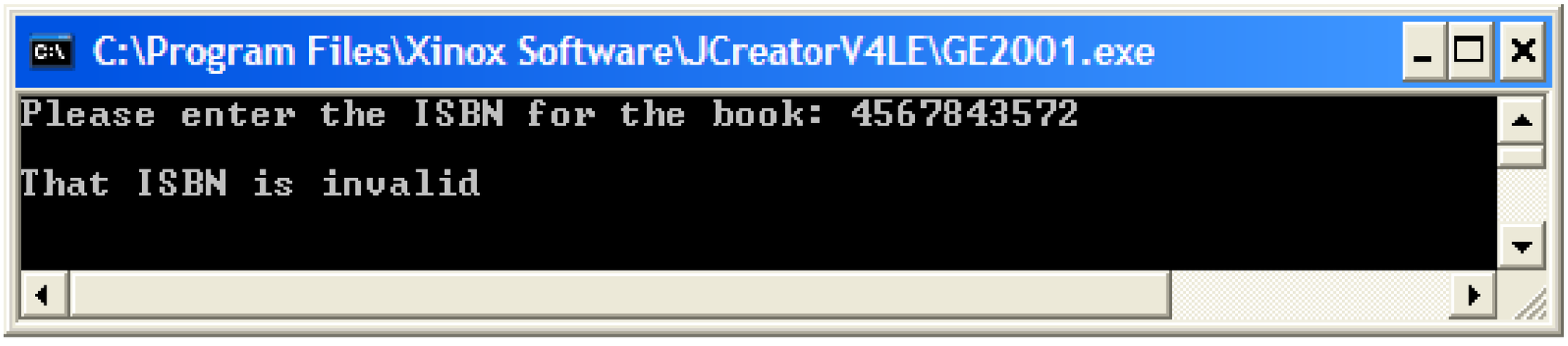
    endif

Now you should try to code the Java solution to this problem

Based on what we know so far, there is some **serious problem-solving** needed here to isolate the first digit, second digit etc. – spend a good while thinking about how you might get these – pen and paper will be very useful! You should use the long data type here to store the ISBN and the individual, extracted, digits. Ignore the possibility of the check digit being an "X" in this case also.

Some runs of the program are shown below:





Simple and Extended Selections

- All the decision-making we have looked at so far has just involved what is referred to as simple selections. In these cases, if the condition evaluates to false, then nothing happens – execution simply continues with the action immediately following the `endif`.

```
if (condition)
    action 1
    action 2
endif
action 3
```

- In the case above, if the condition is false then action1 and action2 do not execute at all and execution continues with action3

Extended Selections

- Sometimes the choice is between two alternative courses of action. Imagine withdrawing funds from an ATM machine. After doing the preliminaries and, provided you have enough money in your account, then the machine should dispense the cash. In pseudocode this might appear as:

```
if (funds available)
    dispense cash
endif
```

- However, what if you have insufficient funds? In this case, the

  ATM should decline your request and give you an appropriate message. The pseudocode for this part might be:

The `if-else` construct

```
if (insufficient funds available)
        display message of apology
endif
```

- Although the combination of the 2 simple if selections is valid here, it doesn't really match how we would phrase such as selection in reality.

- We might say "If there are sufficient funds then give the customer the requested cash *otherwise* apologise and explain that their request cannot be satisfied". The key word here is "otherwise" which sets out an alternative course of action to be followed if the first action cannot be performed.

The `if-else` construct

- In pseudocode, this situation can be realized through the use of the **`else`** keyword as follows:

```
if (condition)
        action 1
        action 2
else
        action 3
        action 4
endif
```

- Here, actions 1 and 2 are only carried out when the test condition evaluates to true. Actions 3 and 4 get executed only when the test condition evaluates to false.

The `if-else` construct

- We should rewrite our earlier pseudocode (with the 2 simple if structures) as follows:

```
if (funds available)
        dispense cash
else
        display message of apology
endif
```

- We should take advantage of an if-else structure whenever there are 2 action blocks that are dependent on the same, but opposite, condition.
- Programming languages generally support an if-else like structure