420-201-RE Assignment 2-A Due on 19 September 2016

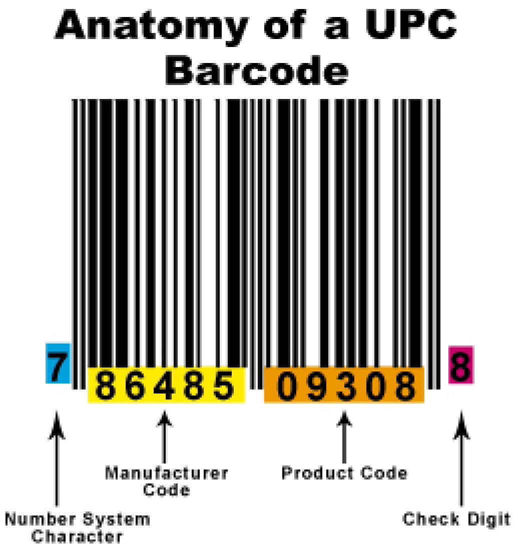
Objectives

1. Practice using relational and logical operators
2. Practice using **switch** construct
3. Practice using appropriate conditional construct for a given task

Background: UPC

UPC (Universal Product Code) is a 12-digit code, generally found on a product's original packaging next to its bar code, the machine-readable representation of the UPC.

Identifying manufacturers and their respective products, UPC codes are used to facilitate both the check-out process and inventory control.



UPC Validation Algorithm

To process a bar code, a UPC scanner implements the following algorithm:

Step 1. Read and scan the bar code is into a 12-digit code of the general form[1]

$$d_0 d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} d_{11}$$

where each of $d_0, d_1, d_2 \cdots, d_{11}$ denotes a single digit integer.

The last digit d_{11}, called the *scanned check digit*, plays an important role in Step 3.

Step 2. Determine $d_{11'}$, the *computed check digit*, as follows:

Step 2.1. Calculate $x = 3 * (d_0 + d_2 + d_4 + d_6 + d_8 + d_{10}) + (d_1 + d_3 + d_5 + d_7 + d_9)$

Step 2.2. let y be the last (rightmost) digit of x.

Step 2.3. If y is 0, then $d_{11'} = 0$; otherwise, $d_{11'} = (10 - y)$

Step 3. If $d_{11} = d_{11'}$ then the UPC is valid; otherwise, invalid.

[1] Indexing sequences starting at 0 is quite common in computer science.

Example 1

As an example, let's apply the algorithm to the example bar code shown above.

Step 1. We scan and read the 12-digit bar code 786485093088. We then dissect the bar code into its 12 digits: $d_0 = 7$, $d_1 = 8$, $d_2 = 6$, $d_3 = 4$, $d_4 = 8$, $d_5 = 5$, $d_6 = 0$, $d_7 = 9$, $d_8 = 3$, $d_9 = 0$, $d_{10} = 8$, and $d_{11} = 8$ as its scanned check digit.

Step 2. Determine $d_{11'}$, the *computed check digit*, as follows:

Step 2.1. $x = 3 * (7 + 6 + 8 + 0 + 3 + 8) + (8 + 4 + 5 + 9 + 0) = 122$

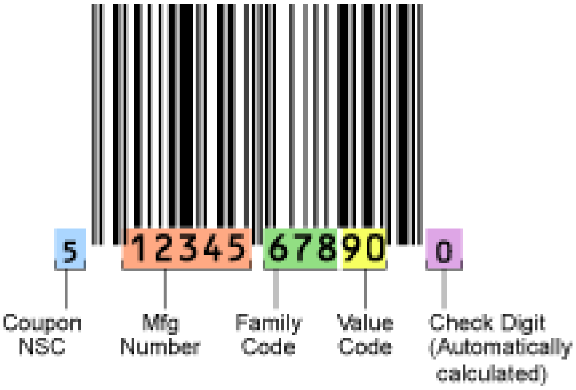
Step 2.2. $y = 2$, the last (rightmost) digit of 122.

Step 2.3. Since $y = 2$ is nonzero, we have $d_{11'} = (10 - 2) = 8$.

Step 3. Since $d_{11} = 8$ and $d_{11'} = 8$ are equal, we conclude that the UPC 786485093088 is valid.

Example 2

For the bar code at right, the scanned bar code is 512345678900 with $d_{11} = 0$. Next, from Step 2.1 we have $x = 3*(5+2+4+6+8+0)+(1+3+5+7+9) = 100$, and from Step 2.2, $y = 0$, and from step 2.3, $d_{11'} = 0$. Finally, since $d_{11} = 0$ and $d_{11'} = 0$ are equal we conclude in Step 3 that the UPC 512345678900 is valid.



Implementation of Scan and Dissect

Step 1 of the algorithm above can be implemented using one of the following two popular strategies:

Strategy 1:

1. Scan the input bar code as a **String** of characters.
2. Dissect the input string into 12 characters, $c_0, c_1, c_2 \cdots, c_{11}$
3. Convert character c_0 to digit d_0, character c_1 to digit d_1, character c_2 to digit d_2, $\cdots$, character c_{11} to digit d_{11}

For example, here is how to convert the first character c_0 to digit d_0

```java
 1  System.out.print("Enter a 12-digit bar code: ");
 2  String strBarcode = keyboard.next();
 3  // make sure barcode's length is exactly 12 -- not shown here
 4
 5  char c0 = strBarcode.charAt(0); // extract the first character
 6  if( c0 < '0' || c0 > '9')  // if c0 is not a digit
 7  {
 8     System.out.println("Invalid character " + c0 + " in " + strBarcode);
 9     System.out.println("A bar code consists of exactly 12 digits ");
10     System.out.println("Try again later. bye.");
11     System.exit(1);
12  }
13  // c0 is a digit character
14  int d0 = c0 - '0'; // convert the digit character c0 to digit integer d0
15
16  // repeat lines 5-14, 11 more times, one for each of the remaining 11 digits
```

Strategy 2:

1. Scan the input bar code as a **long** integer.
2. Using integer division and remainder operations, directly dissect the input **long** value into 12 digits, $d_0, d_1, d_2 \cdots, d_{11}$

For example, here is how to extract the last (rightmost) digit d_{11}:

```java
 1  System.out.print("Enter a 12-digit bar code: ");
 2  // make sure that the user has entered a number
 3  if(! keyboard.hasNextLong())
 4  {  // extract the bad bar code as a string
 5     String badLongBarcode = keyboard.next();
 6     System.out.println("Invalid bar code: " + badLongBarcode);
 7     System.out.println("A bar code consists of exactly 12 digits ");
 8     System.out.println("Try again later. bye.");
 9     System.exit(1);
10  }
11  //  we have a 12-digit bar code
12  long longBarcode = keyboard.nextLong();
13  // make sure longBarcode is positive and has no more than 12 digits -- not shown here
14
15  // start extracting bar code's digits right to left
16  int d11 = (int)(longBarcode / 10); // extract the rightmost digit
17  longBarcode = longBarcode % 10  // drop the rightmost digit
18  // repeat lines 15-16, 11 more times, one for each of the remaining 11 digits
```

Program 1: UPC

Write a program that implements a UPC scanner using *both* strategies above for **Stem 1** of the algorithm.

A sample run of your program should look like the following:

```
Enter strategy number (1 or 2): 1
Enter a 12-digit bar code: 512345678900

Anatomy of Your UPC Bar Code
==================================
UPC                  : 512345678900
NSC                  : 5
Manufacturer Number  : 12345
Product Code         : 67890
Family Code          : 678
Value Code           : 90
Scanned Check Digit  : 0
Computed Check Digit: 0
Validity status      : valid
```

Note: The user input above is shown in red for clarity only. Your program output will of course be all in the same color.

Use your program to validate following numbers as UPC bar codes:

```
179400804501
224000162868
311110856802
451000138107
```

If the user enters any value other than a 12-digit integer value your program should print an error message, inviting the user to try the program later, and then terminate.

Program 2: BMI

Body mass index (BMI) of a person is defined by the formula $\text{BMI} = \dfrac{703 \times weight}{height \times height}$ where *weight* and *height* denote a person's weight in pounds and height in inches.

Using the above formula, the Center for disease Control[2] classifies the *weight status* of a person as follows:

Condition	*Weight status*
$\text{BMI} < 16.0$	seriously Underweight
$16.0 \leq \text{BMI} < 18.5$	Underweight
$18.5 \leq \text{BMI} < 25.0$	Normal
$25.0 \leq \text{BMI} < 30.0$	Overweight
$30.0 \leq \text{BMI} < 35.0$	seriously Overweight
$35.0 \leq \text{BMI}$	Obese

Write a Java program that prompts the user to enter the height and weight of a person, and then reports the BMI and weight status of that person. A sample run of your program should look like this:

```
Enter a person's information:
Height (in inches)? 72
Weight (in pounds)? 210

Body mass index: 28.479
Weight status  : Overweight
```

Your program should validate the user input. Specifically, if either of the input values is negative, your program should display an error message and then terminate. Examples:

```
Enter a person's information:
Height (in inches)? -72

Invalid value for height: -72
You should enter a positive value.
Try again later.
goodbye
```

```
Enter a person's information:
Height (in inches)? 75
Weight (in pounds)? -214

Invalid value for weight: -214
You should enter a positive value.
Try again later.
goodbye
```

[2] www.cdc.gov/nccdphp/dnpa/bmi

Evaluation Criteria

Evaluation Criteria

Correctness of execution of your program	60%
Proper use of Java constructs	10%
Description of purpose of program , Comments on nontrivial steps in code	10%
Format, clarity, and completeness of output	10%
Choice of meaningful variable names, Indentation and readability of program	10%