(48024) Applications Programming Lab 2 guide

Below is the plan that most groups will have arrived at based on guidance from their tutor (although of course other plans are possible). If you do not yet have a workable plan for lab 2, you may use the one described below.

The framework of your solution is a read loop:

```
System.out.print(“Number: “);
int number = In.nextInt();
while (number != -1) {
   -- show the number --
   System.out.print(“Number: “);
   number = In.nextInt();
}
```

In place of -- show the number -- you must write code to show the number in English words. When you encounter a difficult goal such as this, try to start off by writing a much simpler program that only achieves part of the goal, and then gradually add more to it until it achieves the full goal.

Incremental goals

1. Try to write a program that works with a number in the range 0-19. The reason this is a better start than just 0-9 is because all the numbers from 0-19 have special names, so you can actually handle them all the same way. Of course, different incremental goals are possible, and you could also have started by dealing with numbers in the range 0-9 first.

   ```
   Sample I/O:
   Number: 8
   eight
   Number: 13
   thirteen
   Number:19
   nineteen
   Number: -1
   ```

   Now here is the plan:
   Define an array of all the special number names:
   String[] names = { “zero”, “one”, “two”, …… “nineteen” };

Then you can simply use the variable called `number` (the one from your read pattern) as the index into the array to find the word. So the key expression is names[number]. For example, if the number is 7, then names[7] would give you the string “seven” and you just print it out. All up, this plan is two lines of code. One to declare the array, and one to print the word from the array.

2. Try to write a program that works with a number in the range 0-99.

   Now here is the plan:

   ```
   if (the number is less than 20) {
      do it the way you did it before.
   }
   else {
      do it a different way.
   }
   ```

   Let’s look at how to do it a different way when the number is >= 20. You’ll need to extract the two digits individually out of the number.
   e.g. the user enters the number 47. You want to extract the 4 and the 7 digits.
   The “tens” digit is 4, and the “ones” digit is 7. Extract the two digits and store them into variables:

   ```
   int tensDigit = …..;      // formula shown in the lecture slides
   int onesDigit = …..;      // formula shown in the lecture slides
   ```

   You want to show this as: forty seven.
   For the tens digit, you’re going to need a second array of special names for the tens:

   ```
   String[] tensNames = { “”, “”, “twenty”, “thirty”, “forty”, ….. “ninety” };
   ```

   Notice the first two elements are empty strings just so that the words line up with their array positions. So “forty” is at position [4] and “ninety” is at position [9].

   So, if you print tensNames[4] + “ “ + names[7], you will get “forty seven”.
   Because you already extracted the two digits 4 and 7, you can use tensDigit instead of 4, and you can use onesDigit instead of 7..

3. Now you can try to write the program to support all numbers from 0-999.
   This is pretty much the same idea but you should now extract out all 3 digits:

   ```
   int hundredsDigit = …..;
   int tensDigit = …..;
   int onesDigit = …..;
   ```

   In the number 423 “four hundred and twenty three”, the hundreds digit is just using the regular name so you can use the names[] array and print the word “hundred” after it. Then you print “and” (unless the next two digits are zero), and then you print the last two digits in the same style as you did in the previous goal.

Test cases

Some ideas for what to test:

- Check 1 digit numbers, teens, 2 digit numbers and 3 digit numbers.
- Check the grammar. “and” always comes after the hundreds, except when the last two digits are 00.
- Check the spelling. e.g. 4 is four. 14 is fourteen. 40 is forty.
- Check your zeros. Not every 0 becomes “zero”. e.g. 40 is just “forty”, not “forty zero”. In fact the only time you use the word “zero” is when the whole number itself is 0.