Homework 3-1: Calculate a circle's circumference and area

Console

```
Welcome to the Circle Tester

Enter radius:  3
Circumference: 18.85
Area:          28.27

Continue? (y/n): y

Enter radius:  6
Circumference: 37.7
Area:          113.1

Continue? (y/n): n

Goodbye. You created 2 Circle object(s).
```

Operation

- The application prompts the user to enter the radius of a circle.
- If the user enters invalid data, the application displays an appropriate error message and prompts the user again until the user enters valid data.
- When the user enters a valid radius, the application calculates and displays the circumference and area of the circle to the nearest 2 decimal places.
- The application prompts the user to continue.
- When the user chooses not to continue, the application displays a goodbye message that indicates the number of Circle objects that were created by the application.

Specifications

- Create a class named Circle to store the data about this circle. This class should contain these constructors and methods:

```
public Circle(double radius)
public double getCircumference()
public String getFormattedCircumference()
public double getArea()
public String getFormattedArea()
private String formatNumber(double x)
public static int getObjectCount()
```

- The formulas for calculating circumference and area are:

```
circumference = 2 * pi * radius
area = pi * radius^2
```

- For the value of pi, use the PI constant of the java.lang.Math class.
- Create a class named CircleApp that gets the user input, creates a Circle object, and displays the circumference and area.
- Create a class named Validator like the one shown in chapter 7 and use its static methods to validate the data in this application.

Homework 3-2: Roll the dice

Console

```
Welcome to the Paradise Roller application

Roll the dice? (y/n): y

Roll 1:
2
5
Craps!

Roll again? (y/n): y

Roll 2:
2
1

Roll again? (y/n): y

Roll 3:
4
6

Roll again? (y/n): y

Roll 4:
6
6
Box cars!

Roll again? (y/n): y

Roll 5:
1
1
Snake eyes!

Roll again? (y/n): n
```

Operation

- If the user chooses to roll the dice, the application rolls two six-sided dice, displays the results of each, and asks if the user wants to roll again.

Specifications

- Create a class named Die to store the data about each die. This class should contain these constructors and methods:

```
public Die()              // default to a six-sided die
public Die(int sides)     // allow a variable number of sides
public void roll()
public int getValue()
```

- Create a class named PairOfDice to store two dice. This class should contain two instance variables of the Die type, an instance variable that holds the sum of the two dice, and these constructors and methods:

```
public PairOfDice()           // default to six-sided dice
public PairOfDice(int sides)  // allow a variable number of sides
public void roll()
public int getValue1()        // get value of die1
public int getValue2()        // get value of die2
public int getSum()           // get the sum of both dice
```

- You can use the random method of the Math class to generate a random number from 1 to the number of sides on a die like this:

```
int value = (int) (Math.random() * sides);
```

- Create a class named DiceRollerApp that uses the PairOfDice class to roll the dice. This class should display special messages for craps (sum of both dice is 7), snake eyes (double 1's), and box cars (double 6's). For this application, assume that two six-sided dice are used.
- Create a class named Validator that contains static methods that can be used to validate the data in this application.

Homework 3-3: Translate English to Pig Latin

Console

```
Welcome to the Pig Latin Translator.
Enter a line to be translated to Pig Latin:
this program translates from english to pig latin

isthay ogrampray anslatestray omfray englishway otay igpay atinlay

Translate another line? (y/n): n
```

Operation

- The application prompts the user to enter a line of text.
- The application translates the text to Pig Latin and displays it on the console.
- The program asks the user if he or she wants to translate another line.

Specifications

- Parse the string into separate words before translating. You can assume that the words will be separated by a single space and there won't be any punctuation.
- Convert each word to lowercase before translating.
- If the word starts with a vowel, just add *way* to the end of the word.
- If the word starts with a consonant, move all of the consonants that appear before the first vowel to the end of the word, then add *ay* to the end of the word.
- If a word starts with the letter *y*, the *y* should be treated as a consonant. If the *y* appears anywhere else in the word, it should be treated as a vowel.
- Check that the user has entered text before performing the translation.

Enhancements

- Keep the case of the original word whether it's uppercase (TEST), title case (Test), or lowercase (test).
- Allow punctuation in the input string.
- Translate words with contractions. For example, *can't* should be *an'tcay*.
- Don't translate words that contain numbers or symbols. For example, 123 should be left as 123, and bill@microsoft.com should be left as bill@microsoft.com.

Notes

- This application requires the use of string handling to parse the input string into separate words, to analyze letters at the beginning of each word, to identify consonants and vowels, and to add Pig Latin word endings.
- There are no official rules for Pig Latin. Most people agree on how words that begin with consonants are translated, but there are many different ways to handle words that begin with vowels.

Homework 3-4: Convert numbers to words

Console

```
Welcome to the Number to Word Converter.

Enter the number you want to convert to words: 3842
three thousand eight hundred forty two

Convert another number? (y/n): y

Enter the number you want to convert to words: 2001
two thousand one

Convert another number? (y/n): y

Enter the number you want to convert to words: 4815
four thousand eight hundred fifteen

Convert another number? (y/n): y

Enter the number you want to convert to words: 400
four hundred

Convert another number? (y/n): n
```

Operation

- The user enters a value from 0 to 9999, and the program converts it to an English representation of the value as shown in the console output.
- To program continues until the user responds to the "Convert another number" question with a value other than Y or y.

Specifications

- You are free to use any technique you can devise to split the number entered by the user into its thousands, hundreds, tens, and ones digits and to create the string representation of the number.
- You can use arrays of String objects for units, teens, and tens words. For instance, the teens array may include {"ten", "eleven", "twelve"…and so on}. The tens array may include {"twenty", "thirty", "forty"…and so on}.
- If the tens digit is greater than 1, the word will use "twenty", "thirty", "forty", and so on. Then, the last word be a units digit.
- If the number ends with 00, only the thousands and hundreds places are printed.
- If the number ends with 01 through 09, the last word will be a units digit.
- If the tens digit is 1, the last word will be "ten", "eleven", "twelve" and so on.

Hints

- One way to split the user's input into separate digits is to treat the value as a string and use the substring method to extract the separate digits. If you use this technique, be sure to account for values with fewer than four digits.

- Another way to extract the separate digits is to treat the number as an integer and use a combination of integer division and modulo division. (Remember that integer division truncates the results.)
- Don’t forget to provide for an entry of zero!

Enhancements

- Validate the input to make sure it isn’t negative or greater than 9,999.
- Allow negative numbers on input.
- Allow double input with up to two decimal digits, and format the string as it would be written on a check. (For example, “One hundred thirty two dollars and 38 cents.”)