Programming in Java

Assignment 4

Number format converters

Marks available: 100

Date issued: 15/11/19

Deadline: 25/11/19, 17:00

You may not use any Java class or library that you did not create that converts between number formats, bases, etc. in your solution to this assignment. All conversions must be implemented 'manually' by you. If you use any such class or library your mark for this assignment will be zero.

This assignment is worth 12.5% of your final module mark.

The functionality component of the following exercises is worth 60% of the marks for each exercise. This means, the extent to which your methods provide the required behaviour as determined by the JUnit tests.

The remaining 40% for each exercise will be awarded for the 'quality' of your code. For this assignment this means **the quality of your solution design**. However code that uses bad layout and bad naming, making the design hard to discern, will receive a quality mark of zero.

The point of this assignment is for you to come up with your own design. That design must, however, fulfill the requirements below. A missing `Converter` class, for example, will result in an overall mark of zero. You should think about any other classes you might need, the relationships between those classes, whether you want to use `interface`s, etc.

A 'low quality' solution will be one that ignores object-oriented principles. A higher quality solution will attempt to apply those principles to create a sensible design.

Submission instructions

Submit your work by the deadline above as a zip archive. Use only 'zip' format and no other. Submit only .java source files in your archive. Do NOT submit .class files or any type of file. Please do not submit entire Eclipse (or other IDE) projects. In your zip you must submit all Java files necessary for the assignment, including any that were provided for you. You do not need to submit any test files that were provided.

To submit, gather all of the Java files into a folder, zip that folder, and then submit the zip. Do this using your operating system functionality. Do not use third-party tools or websites to compress your files.

In your programs, do not output anything to the console unless explicitly asked to. If you add `System.out.println()` calls in your code (e.g. for debugging), remove them before you submit.

Each Java file must have a package declaration at the topic of it (these declarations must be on the very first line of the file). The package declaration this assignment is:

com.bham.pij.assignments.converters;

All classes that you write for this assignment must have this package declaration. **Do not create new packages**. Do not forget to add the semicolon at the end of this line. All characters in a package name should be lower case.

In each of these exercises, you are not restricted to only creating the *required* methods. If you want to, you can create others. However, you must create **at least** the required methods.

Do not put any code that gets user input in the required methods below. If you do, your work can't be tested. This means that you should not use `Scanner` in any required method or in any method called by a required method, etc.

Please follow these instructions. If you do not you could get a lower mark than you would have done for following them or a mark of zero. Please ask if you are unsure.

Introduction

For this assignment you will write some code that allows numbers represented as strings to be converted from one number base to another.

The conversions your code must support are:

- Binary to hexadecimal.
- Hexadecimal to binary.
- Binary to decimal.
- Decimal to binary.

The range of all values in these exercises is the range of the 8-bit integers. This means that all binary numbers always contain 8 bits (range: 00000000 - 11111111), all hexadecimal numbers always contain 2 digits (range 00 - FF) and all decimal numbers contain 1, 2 or 3 digits (range 1 - 255). All input values and output values must be in the correct format for the range. If an input value is not in the correct range, or does not have the correct number of bits or digits, an exception must be thrown as discussed below.

Exercise 1: Number base conversion [100 marks]

You must write a class called `Converter`. Your class must fulfil the requirements in the supplied HTML documentation. This class must **not** be `abstract`.

This class has, as a member, an `enum`. To declare the `enum` for the class `Converter` add the following line of code to your `Converter` class:

```
public static enum ConvertMode {BIN2HEX, HEX2BIN, BIN2DEC, DEC2BIN};
```

When attempting to convert a number, the `Converter` constructor will be called with a value from this enum as a parameter. For example:

```
Converter converter = new Converter(ConvertMode.BIN2DEC);
```

This creates a converter that can convert from binary to decimal. Note that the constructor does not do the actual conversion. That occurs in the `convert()` method.

To convert a binary number to a decimal number (for example), using the converter declared above, the following method call would be used (for an example binary number):

```
String convertedValue = converter.convert("11111111");
```

This would set `convertedValue` to "255".

If an input value to the `convert()` method is not valid (i.e. it is not expressed in the appropriate format for the conversion process in question) then you must throw an exception of type `InvalidFormatException` and **not** attempt the conversion.

You need to create the `InvalidFormatException` class. It should extend `RuntimeException`. This exception class doesn't do anything. The `catch` block for this exception will simply call `printStackTrace()` which is a standard exception method that you don't have to write. You only need to provide a default constructor. The rest of the methods mentioned in the HTML documentation are provided by ancestor classes.