## Specifications

### Problem Description

You are going to write a program that reads book price data from a file, performs some computations, and displays a table of cost data.

Input: The file contains, on the first line, the title of the book. On the second line it contains the base price for a new copy of the book.

The rest of the file consists of lines in this format:

- Name of seller
- Shipping cost
- Quality of book (1 = new, 2 = like new, 3 = very good, 4 = good, 5 = acceptable)

You may not assume that the file exists. You may not assume that it contains any text. If it exists and contains text, it might contain the title and base price of a new copy of the book (the header information) only, or it might contain the header information and any number of complete records. You may assume that the header information, if present, is complete. You may assume that each record, if present, is complete. You may assume the correct data type for all data in the file.

Output: You will display the title of the book and a table that includes a header and an output line for each record from the file (seller, shipping, quality) which is the seller, the quality of the book, the cost of the book, and the cost of the book plus shipping.

Interface: Your program should prompt for the name of the input file using the following prompt:

- `What is the name of the file?`

Your program should use the following error message if the file is not found.

- `The file was not found.`

Your program should use the following message if the file is empty. (Substitute the name of the file provided by the user.)

- `empty.txt is empty.`

If the file contains the header record only, the program should produce the following output (header information read from the file is underlined).

```
There are no copies of Turing's Vision: The Birth of Computer Science for sale.
```

For a file that contains a header plus seller records, output should be formatted as follows. Cost and Cost + Shipping will be shown with two decimal places. The seller column should have a width of 30 characters. All other columns have a width of 20 characters.

```
Title: Turing's Vision: The Birth of Computer Science
```

```
Seller                        Quality             Cost                Cost + Shipping
BestBookDeals                 Like new            18.89               22.88
UsedTextbooksRUs              Acceptable          14.69               14.69
JohnsScienceBooks             Very good           16.79               20.78
ValueBookorama                New                 20.99               20.99
A1BookSellers                 Good                15.74               19.73
AlreadyUnderlined             Acceptable          14.69               18.68
QualityBooks                  New                 20.99               24.98
BooksBooksBooks               Like new            18.89               22.88
```

Design specifications: Use only techniques that are covered in the first four lessons. Use console and text file input and console output.

Do not, under any circumstances, use `System.exit`. Design your methods to have one `return` statement. The main method should have no `return` statements. Use `break` only in `switch` statements. Do not submit code that uses these constructs inappropriately. Submit whatever you can write that does not include these constructs.

For this program, you must write (and use) the following two methods:

- `calculateCost`
  `calculateCost` takes two parameters – the base cost of the book (`double`), and the quality (`int`)
  `calculateCost` returns the adjusted cost of the book (`double`) according to this formula: A "new" book's cost is the base cost. A "like new" book's cost is base cost – 10%. A "very good" book's cost is base cost – 20%. A "good" book's cost is base cost – 25%. An acceptable book's cost is base cost – 30%. So, for example, if a book's base cost is $25.99 and it's in good condition, its cost is $25.99 – 25% of 25.99, or $19.49.
- `getQualityString`
  `getQualityString` takes one parameter – the quality of the book (`int`)
  `getQualityString` returns the appropriate string (`String`) based on the quality (e.g. for a quality of 1, `getQualityString` should return "New")

## Testing

### Test Case 1

#### *Purpose*
Test a case with a file with header information and complete records with books of all quality types.

#### *Input (file)*
```
Turing's Vision: The Birth of Computer Science
20.99
BestBookDeals
3.99
2
UsedTextbooksRUs
0
5
JohnsScienceBooks
3.99
3
ValueBookorama
0
1
A1BookSellers
3.99
```

```
4
AlreadyUnderlined
3.99
5
QualityBooks
3.99
1
BooksBooksBooks
3.99
2
```

***Expected output***

```
Title: Turing's Vision: The Birth of Computer Science
Seller                        Quality             Cost                Cost + Shipping
BestBookDeals                 Like new            18.89               22.88
UsedTextbooksRUs              Acceptable          14.69               14.69
JohnsScienceBooks             Very good           16.79               20.78
ValueBookorama                New                 20.99               20.99
A1BookSellers                 Good                15.74               19.73
AlreadyUnderlined             Acceptable          14.69               18.68
QualityBooks                  New                 20.99               24.98
BooksBooksBooks               Like new            18.89               22.88
```

### Test Case 2

***Purpose***

An invalid file name.

***Input (console)***

doesnotexist.txt

***Expected output***

```
The file was not found.
```

### Test Case 3

***Purpose***

A file with header information but no records.

***Input (file)***

```
Turing's Vision: The Birth of Computer Science
20.99
```

***Expected output***

```
There are no copies of Turing's Vision: The Birth of Computer Science for sale.
```

### Test Case 4

***Purpose***

An empty file.

***Input (console)***

```
empty.txt
```

***Input (file)***

***Expected output***

```
empty.txt is empty.
```

## Rubric

An exceptional-quality assignment will meet the following standards:

- Meeting functional and design specifications
  The Java program works and meets all of the specifications, with no additional unspecified functionality. The programmer has used programming techniques from the first through fourth lessons only. The two required methods are written and used to specifications. If the program misses specifications or does not function correctly, errors are acknowledged with a thorough and reflective analysis in the testing section (points will be removed for missed specifications).
- Communicating with identifiers and white space
  The program makes appropriate use of variables. Variables, methods, and constants are named according to convention and are named for understandability and purpose. White space, both vertical and horizontal, is correctly used for readability and meets programming conventions.
- Communicating through documentation
  The Java program contains comments including the programmer's name and date. There are block comments (as many as necessary) for each distinct block of code which accurately describe what the block is accomplishing by relating the code to the problem being solved. Javadoc is included at the top of the program and for each method and meets the javadoc standards.
- Assumptions and Testing
  Testing is thorough. If there are errors, they are described in the testing section. If there are questions, they are answered thoughtfully in the testing section. All assumptions are made explicit.