AP COMPUTER SCIENCE A – EDIT CSV DATABASE

The goal of this assignment is to write a program `EditCSValueDB.java` that will be able to work with and edit "`.csv`" style databases. Your program will need to use the classes `CSValueDB`, `CSValueRecord`, `FieldValue`, and `package IO`. You should start with the file `EditCSValueDBStub.java`.

1) Add code to method `private static int readAction ( ConsoleInput ci )` to prompt the user on which action to perform. Your code should check that a correct action, *i.e.,* 1 through 6 (inclusive), has been entered. If not, the user should be re-prompted. The prompt should look something like

```
       Action:

 Print Record = 1
   Add Record = 2
Delete Record = 3
  Edit Record = 4
        Search = 5
          Exit = 6

     Action = ? 1
```

Note that the bold print corresponds to user input.

2) Add code to method `private static int readRecordNumber ( int numRecords, ConsoleInput ci )` which reads a valid record number, *i.e.*, from `0` to `numRecords-1` (inclusive). If the user enters an invalid record number, the user should be re-prompted.

3) Add code to method `private static void printRecord ( ArrayList<CSValueRecord> recordList, String [] fieldNames, ConsoleInput ci )` which will print a record to the console. In the stub code, note that the `String fmt` is a format specifier, *e.g.*, "`%13s`", which can be used with the `String.format` method to align the output. The output from this method should look something like

```
77 records loaded.

Record Number = ? 3

Record 3:

         Name = All-Bran with Extra Fiber
         Type = C
     Calories = 50
      Protein = 4
          Fat = 0
       Sodium = 140
        Fiber = 14
Carbohydrates = 8
        Sugar = 0
    Potassium = 330
     Vitamins = 25
        Shelf = 3
       Weight = 1
         Cups = 0.5
       Rating = 93.704912
```

where bold print corresponds to user input. Use method `getRawValue()` of class `FieldValue` to print the field values. Also, use `readRecordNumber` to read in the record number.

AP COMPUTER SCIENCE A – EDIT CSV DATABASE

4) Add code to method `private static String readRawValue ( char type, String prompt, ConsoleInput ci )` which reads a field value and returns it as a `String`. This method should check if the user input can be parsed according to its type, *i.e.*, '`i`' (`int`), '`d`' (`double`) or '`s`' (`String`). In other words, if the type is '`i`' or '`d`' and the user input cannot be parsed accordingly, then the user should be re-prompted.

5) Add code to method `private static void addRecord ( ArrayList<CSValueRecord> recordList, String [] fieldNames, char [] types, ConsoleInput ci )` that will add a record to the database. When this method is executed, the user prompts on the console should look something like

```
         Name (s) = ? Jo'Mama's Bamboozle Bits
         Type (s) = ? C
     Calories (i) = ? 700
      Protein (i) = ? 0
          Fat (i) = ? 20
       Sodium (i) = ? 50
        Fiber (d) = ? 0.0
Carbohydrates (d) = ? 150.0
        Sugar (i) = ? 500
    Potassium (i) = ? 0
     Vitamins (i) = ? 0
        Shelf (i) = ? 1
       Weight (d) = ? 1.3
         Cups (d) = ? 1.5
       Rating (d) = ? 4.38

Record Number 77 added.
```

where the bold is user input. Note that the prompts contain the type of the field, *i.e.*, '`i`' (`int`), '`d`' (`double`) or '`s`' (`String`). Also, in the stub code, note that the `String fmt` is a format specifier, *e.g.*, "`%13s`", which can be used with the `String.format` method to align the prompts. Use `readRawValue` to read in the user input.

6) Add code to method `private static void deleteRecord ( ArrayList<CSValueRecord> recordList, ConsoleInput ci )` that will remove, *i.e.*, erase, a record from the database. The `remove` method of class `ArrayList` will be useful. In any case, when executing method `deleteRecord`, the console should look something like

```
77 records loaded.

Record Number = ? 45

Record 45 deleted.

76 records loaded.
```

where, again, the bold corresponds to user input. Use `readRecordNumber` to read in the record number.

AP COMPUTER SCIENCE A – EDIT CSV DATABASE

7) Add code to method `private static int readEditAction ( int fw, String [] fieldNames, ConsoleInput ci )` that prompts the user for an edit action. When this method is executed, the console should look something like

```
    Edit Action:

         Name =  1
         Type =  2
     Calories =  3
      Protein =  4
          Fat =  5
       Sodium =  6
        Fiber =  7
Carbohydrates =  8
        Sugar =  9
    Potassium = 10
     Vitamins = 11
        Shelf = 12
       Weight = 13
         Cups = 14
       Rating = 15
         Done = 16

  Edit Action = ? 4
```

where the bold print corresponds to user input (thus the user has chosen to edit the "Protien" field). The method should check if a correct edit action has been entered, *i.e.*, `1` through `fieldNames.length+1` (inclusive), and if not, the user should be re-prompted. In the stub code, the variable `fw` passed in is that which has been returned by method `getFieldNameFieldWidth`, and `fmt` is a format specifier, *e.g.*, "`%13s`", which can be used in `String.format` to line up the lines of the prompt. The variable `intFmt` is also a format specifier, *e.g.*, "`%2d`", which gives the field width of the number of edit actions, and which can be used in `String.format`, again, to line up the lines of the prompt.

AP COMPUTER SCIENCE A – EDIT CSV DATABASE

8) Add code to method `private static void editRecord ( ArrayList<CSValueRecord> recordList, String [] fieldNames, char [] types, ConsoleInput ci )` that will edit the fields of an existing record. When this method is executed, the output to the console should look something like

```
77 records loaded.

Record Number = ? 3

    Edit Action:

         Name =  1
         Type =  2
     Calories =  3
      Protein =  4
          Fat =  5
       Sodium =  6
        Fiber =  7
Carbohydrates =  8
        Sugar =  9
    Potassium = 10
     Vitamins = 11
        Shelf = 12
       Weight = 13
         Cups = 14
       Rating = 15
         Done = 16

  Edit Action = ? 9

        Sugar (i) = ? 4
```

where bold corresponds to user input. As before, use `readRecordNumber` to read in the record number. Also, use `readEditAction` to read in the edit action, and `readRawValue` to read in the field value. In the stub code, the variable `fmt` may be used to line up the prompts. Finally, method `set` of `ArrayList` will prove useful.

9) Add code to `private static void search ( ArrayList<CSValueRecord> recordList, ConsoleInput ci )` that prompts the user for a token, searches the records of the database for that token, and then prints the record numbers in which the token appears. The output to the console should look something like

```
Token = ? Bran

Records containing "Bran":
 0 1 2 3 8 9 19 28 52 58 59 64 70
```

where, again, the bold print corresponds to user input. If no records are found, you should print something like

```
No records found.
```

to the console.

Once you test your program and are convinced it is working correctly, submit your file `EditCSValueDB.java`.