AP COMPUTER SCIENCE B – CSV DATABASES

1) Put the file `Cereal.xlsx` into Excel. This Excel spreadsheet contains data concerning a bunch of breakfast cereals. From the spreadsheet, export the file `Cereal.csv` (File → Export → CSV). The first few lines of the file should look like

```
Name,Type,Calories,Protein,Fat,Sodium,Fiber,Carbohydrates,Sugar,Potassium,Vitamins,Shelf,Weight,Cups,Rating
100% Bran,C,70,4,1,130,10,5,6,280,25,3,1,0.33,68.402973
100% Natural Bran,C,120,3,5,15,2,8,8,135,0,3,1,1,33.983679
All-Bran,C,70,4,1,260,9,7,5,320,25,3,1,0.33,59.425505
All-Bran with Extra Fiber,C,50,4,0,140,14,8,0,330,25,3,1,0.5,93.704912
Almond Delight,C,110,2,2,200,1,14,8,-1,25,3,1,0.75,34.384843
.
.
.
```

Note that all of the tokens are separated by commas (CSV stands for Comma Separated Values), and the `.csv` format is an industry standard for databases. The first line of the file contains titles for the columns (called the *field names*). Subsequent lines are rows of data. A line of data is called a *record*, and a single entry in a record is called a *field value*.

2) Double click on the `csv_doc` folder, and then on the `index` icon. The documentation for three classes, *i.e.*, `CSValueDB`, `CSValueRecord` and `FieldValue`, should appear. The class `CSValueDB` is a container that may hold any `.csv` file. It might also be worthwhile to look at the source code in `Notepad++` at this point, *i.e.*, the files `CSValueDB.java`, `CSValueRecord.java` and `FieldValue.java`.

3) Type in and run the program `TestDB.java` pretty much as listed below. The program needs the files `CSValueDB.java`, `CSValueRecord.java`, `FieldValue.java`, and `package IO`.

```
import IO.FileOutput;
/**/
public class TestDB {
  /**/
  public static void main ( String [] arg ) {
    /**/
    int numFields, fw;
    char [] ft;
    CSValueDB db;
    FileOutput fo;
    String fmt;
    String [] fn;
   /*
    * Load the database from a file and print it nicely to a .txt file.
    */
    db=new CSValueDB("Cereal.csv");
    fo=new FileOutput("Cereal.txt");
    fo.println(db);
    fo.close();
   /*
    * Print the field names and field types to the console.
    */
    ft=db.getFieldTypes();
    fn=db.getFieldNames();
    numFields=fn.length;
    /**/
    fw=getFieldWidth(fn);
    fmt = "%" + fw + "s";
    System.out.println();
    for ( int i=0; i<numFields; ++i ) System.out.println( String.format(fmt,fn[i]) + " = " + ft[i] );
    /**/
    return;
  }
 /*
  * Calculate the field width of the field names.
  */
  private static int getFieldWidth ( String [] fn ) {
```

```
    /**/
    int numFields, len, maxLen;
    /**/
    numFields=fn.length;
    maxLen=0;
    for ( int i=0; i<numFields; ++i ) {
      len=fn[i].length();
      if ( len > maxLen ) maxLen=len;
    }
    /**/
    return maxLen;
  }
}
```

When you run the program, it should make a file `Cereal.txt`, which is just a nicely formatted copy of the database. Go ahead and look at `Cereal.txt` in `Notepad++`. The program's output to the console should look like

```
         Name = s
         Type = s
     Calories = i
      Protein = i
          Fat = i
       Sodium = i
        Fiber = d
Carbohydrates = d
        Sugar = i
    Potassium = i
     Vitamins = i
        Shelf = i
       Weight = d
         Cups = d
       Rating = d
```

which lists the type for each column of field values, *i.e.*, `int` (`i`), `double` (`d`) or `String` (`s`). Methods `public int getIntegerValue ( )`, `public double getDoubleValue ( )` and `public String getStringValue ( )` of class `FieldValue` can be used, as appropriate, to get a field value of the correct type.