Lab 2 Objectives

Part 1 of a three part lab!

File processing lab. Includes array & arraylist logic. Also basic setup of an abstract class with abstract methods.

Includes csv for downloading and reading into your java file and lab objectives!

**[ UML Design of classes ]**

|  |
| --- |
| ***Client*** |
|  |
| +*readData*()  +*processData*()  +*printData*() |

|  |
| --- |
| **BankRecords** |
| -robjs[] : new BankRecords[600]  -array: ArrayList<List<String>>  -id:string  -age:int  -sex:String  -region:String  -income:double  -married:String  -children:int  -car:String  -save\_act:String  -current\_act:String  -mortgage:String  -pep:String |
| +getId():String  +getAge():Int  ::  +setId(id:String)  +setAge(age:int)  ::  +*readData*()  +*processData*()  +*printData*() |

|  |
| --- |
| **BankRecordsTest?** |
|  |
| **+main()** |

1. **Client.java source set up**

**[ Create class file as abstract ]**

-Include 3 abstract methods *inheritable* for your BankRecords class.

Ex. of abstact method.

public abstract class Client {

    public abstract void readData();  //read file detail

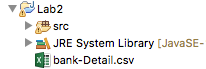
//finish remaining methods

}

1. **BankRecords.java source setup**

**[ Add .csv source file to your project folder ]**

Take your .csv file and add it to the root of your project folder.

****

**[ Create necessary class variables ]**

//setup static objects for IO processing

//array of BankRecords objects

static BankRecords robjs[] = new BankRecords[600];

//arraylist to hold spreadsheet rows & columns

static ArrayList<List<String>> array = new ArrayList<>();

//instance fields

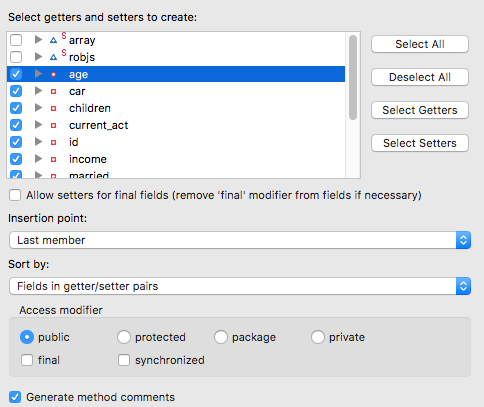
private String id;

private int age;

/\*create remaining instance fields with appropriate data types\*/

**[ Generate getters & setters for ALL your *instance* fields ]**

From your menu go to **Source > Generate Getters and Setters…**

****

Note- even though we don’t need all getters and setters for this lab, we may need them in subsequent labs! ☺

**Setup class functions to perform the following tasks**

**[ Read data from spreadsheet ]**

public void readData() {

BufferedReader br;

//initialize reader object and set file path to root of project

br = new BufferedReader(new FileReader (new File("bank-Detail.csv")));

String line;

//read each record in csv file

while ((line = br.readLine()) != null) {

//parse each record in csv file by a comma ( , )

//into a list stored in the arraylist-> Arrays

array.add(Arrays.asList(line.split(",")));

}

     processData(); //call function for processing record data

}

**[ Process data from arraylist ]**

public void processData() {

//create index for array while iterating thru arraylist

int idx=0;

    //create for each loop to cycle thru arraylist of values

    //and PASS that data into your record objects' setters

    for (List<String> rowData: array) {

      //initialize array of objects

    robjs[idx] = new BankRecords();

    //call setters below and populate them, item by item

    robjs[idx].setId(rowData.get(0)); //get 1st column

    robjs[idx].setAge(Integer.parseInt(rowData.get(1))); //get 2nd column

/\*continue processing arraylist item values into each array object-> **robjs[] by index**\*/

idx++;

}

printData(); //call function to print objects held in memory

}

**[ Print data from array ]**

public void printData() {

//1. Set appropriate headings for displaying first 25 records

//2. Create for loop and print each record objects instance data

//3. Within for loop use appropriate formatting techniques to print

// out record detail

}

**[ Incorporate main to run your app! ]**

**[ Error trappings! ]**

Make sure to include appropriate exception handling with *try catch* blocks for any file being processed.

\*Make use of such runtime exceptions such as *FileNotFoundException* or other possible exceptions that may occur during the processing of any file(s).

<https://www.tutorialspoint.com/java/java_exceptions.htm>

**[ Print formatting hints ]**

Use various print formatting for printing detailed record information especially to make output look professional and columnar like with the format specifier **%** symbol followed by a converter.

Popular converters to use include:

|  |
| --- |
| %f -> float  %d -> int  %s -> string  %n -> newline |

Ex. of an appropriate heading that can be printed may be as follows:

System.out.println("ID\t\tAGE\t\tSEX\t\tREGION\t\tINCOME\t\tMORTGAGE");