Criterion B - Solution Overview

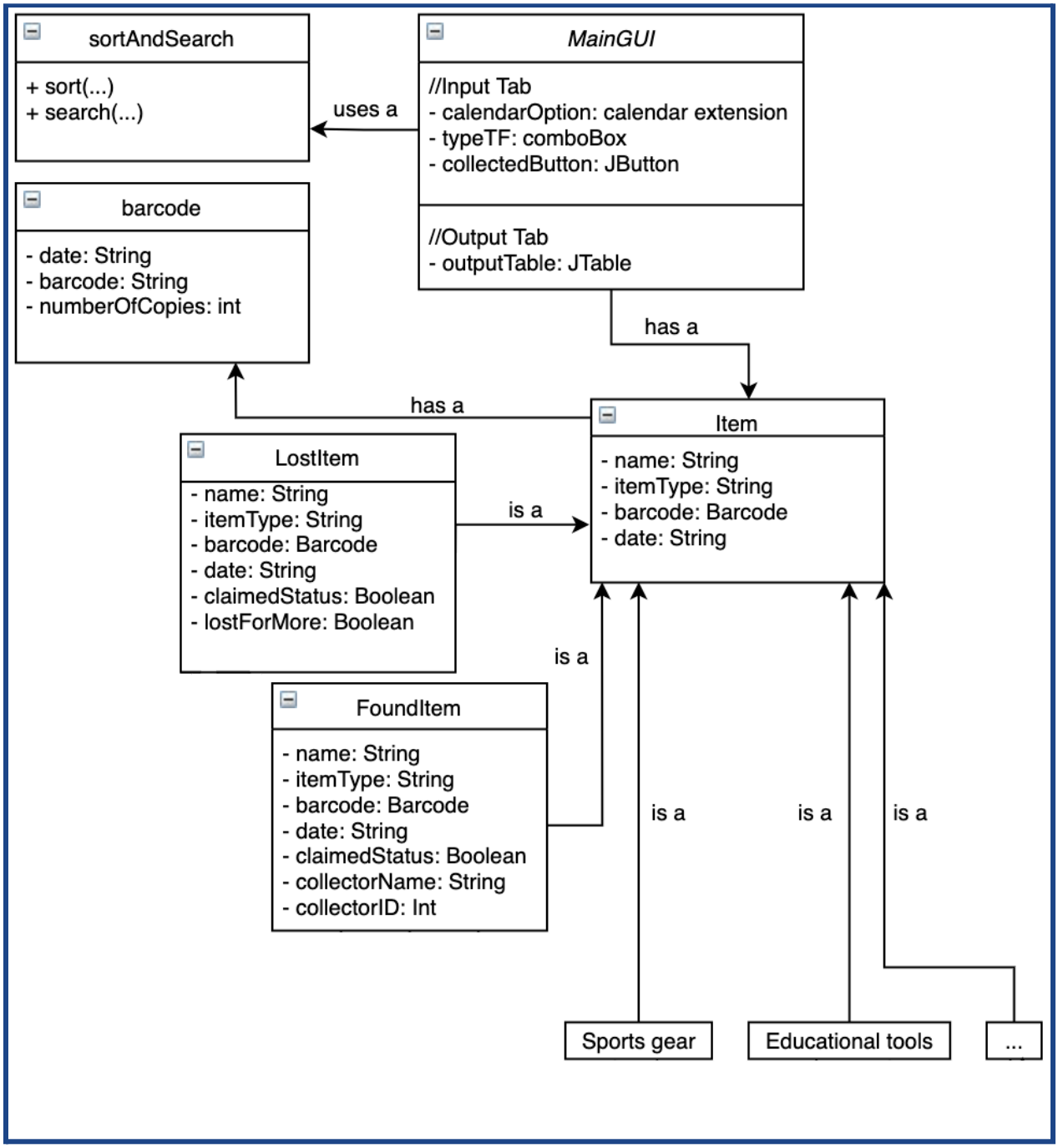
Input and Output Tables:

Input	Data Type	Normal Range etc.	Example
Date	Date Object	With the calendar option and some formatting, the input should first be a Date object (represented in the form yy-mm-dd when .getDate() is used)	19-02-20
Number of copies (of barcodes needing printing)	Integer	0-21	20
File name	String	A string of letters, numbers, '_' or '-'	"barcode_as_pdf"
Type of the item found	String	1-10 types listed in a combo box	"Water bottle"
Claimed status	Boolean	Two options to choose from	False (means hasn't been collected)
Lost status: has been lost for longer than four weeks?	Boolean	Can select radio button or leave unselected	True (means has been lost for four weeks)
Sort item by	String	Three options to choose from	"Name"
Name of collector	String	A string of characters consisting of the last name and first name	"Charlie Parker"
Collector ID	Integer	A five-digit integer	15638

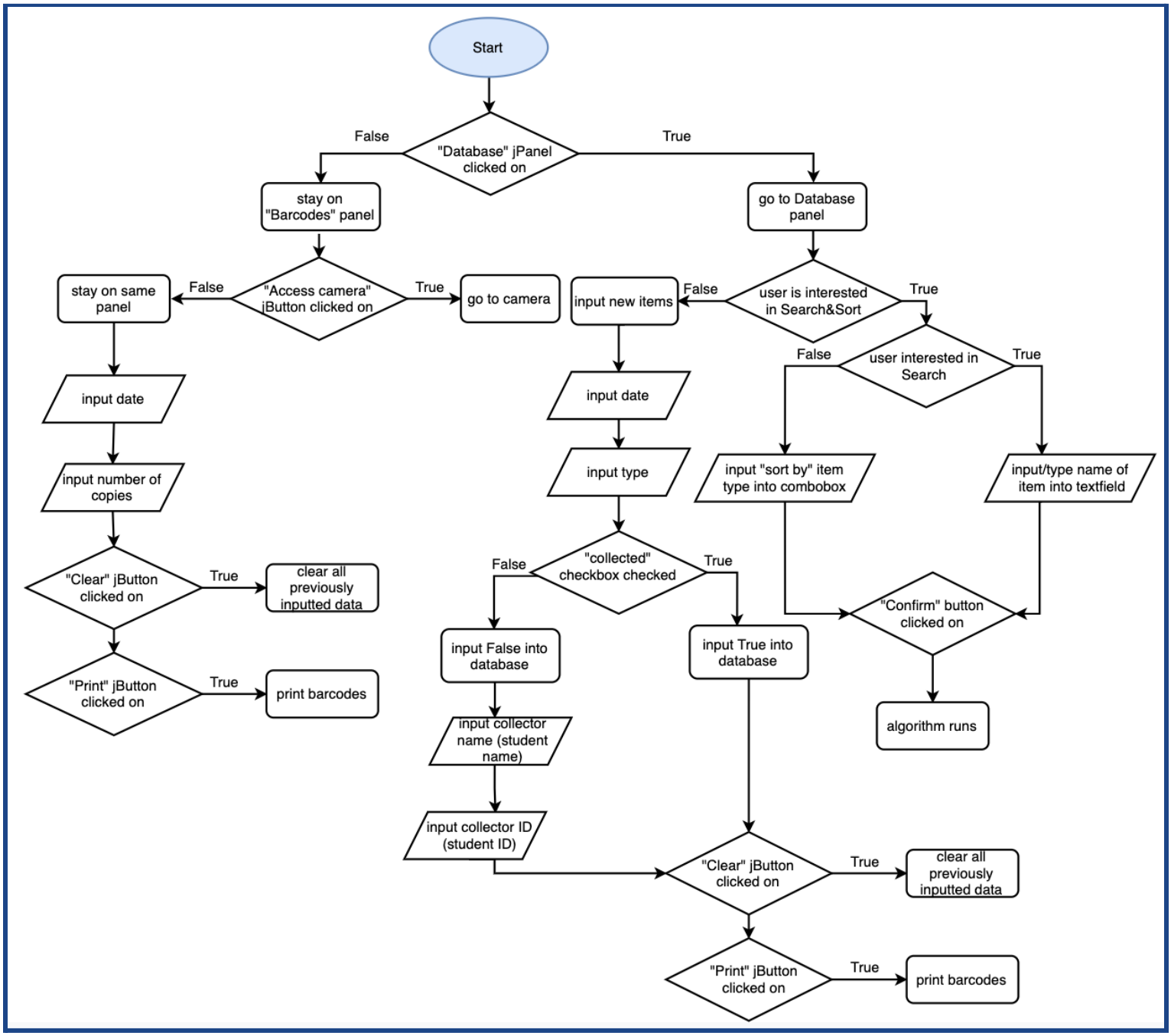
Output	Data Type	Normal Range etc.	Example
Item	String	Would be a large range of different strings since there are a variety of lost items.	"Blue Cap"
Date	String	Eight characters	"01-02-19"
Type	String	1-10 types listed in the combo box	"Hat"
Status	String	Two options to be displayed	"Hasn't been collected"

Has been lost for longer than four weeks	String	Two options to be displayed (e.g. Has or hasn't)	"Has been lost for four weeks"
Barcodes on a pdf file	Document	No range	11 identical barcodes arranged on a pdf file

Class Diagram:



Flowchart:



Note: Later on after creating the flowchart, I decided that the ‘main,’ ‘input items,’ ‘data’(for the jTable and searching and sorting data), and ‘barcodes’ sections should be on separate jTabbedPanes as formatting is more clear that way (seen in the final prototype below). Hence, aside from the mention of switching between two panels (there will be four), the main functionality of the program stays the same and is represented in the flowchart.

Pseudocode:

Displaying data table of items:

```
For (int row = 0; row < itemsArray.length; row++){
        Table[row][0] = item[row].getName();
        Table[row][1] = item[row].getType();
        Table[row][2] = item[row].getDate();
        Table[row][3] = item[row].getName();
        if (claimedStatus == true){
                Table[4] = "Collected";
        } else {
                Table[4] = "Uncollected";
```

Changing the GUI display depending on the claimed status the user chooses:

```
boolean claimedStatus = true;
if(collectedRadioButton.isSelected()){
        collectorNameTF.setVisible(true);
        collectorIDTF.setVisible(true);
        collectorNamejLabel.setVisible(true);
        collectorIDjLabel.setVisible(true);
}else{
        collectorNameTF.setVisible(false);
        collectorIDTF.setVisible(false);
        collectorNamejLabel.setVisible(false);
        collectorIDjLabel.setVisible(false);
}
```

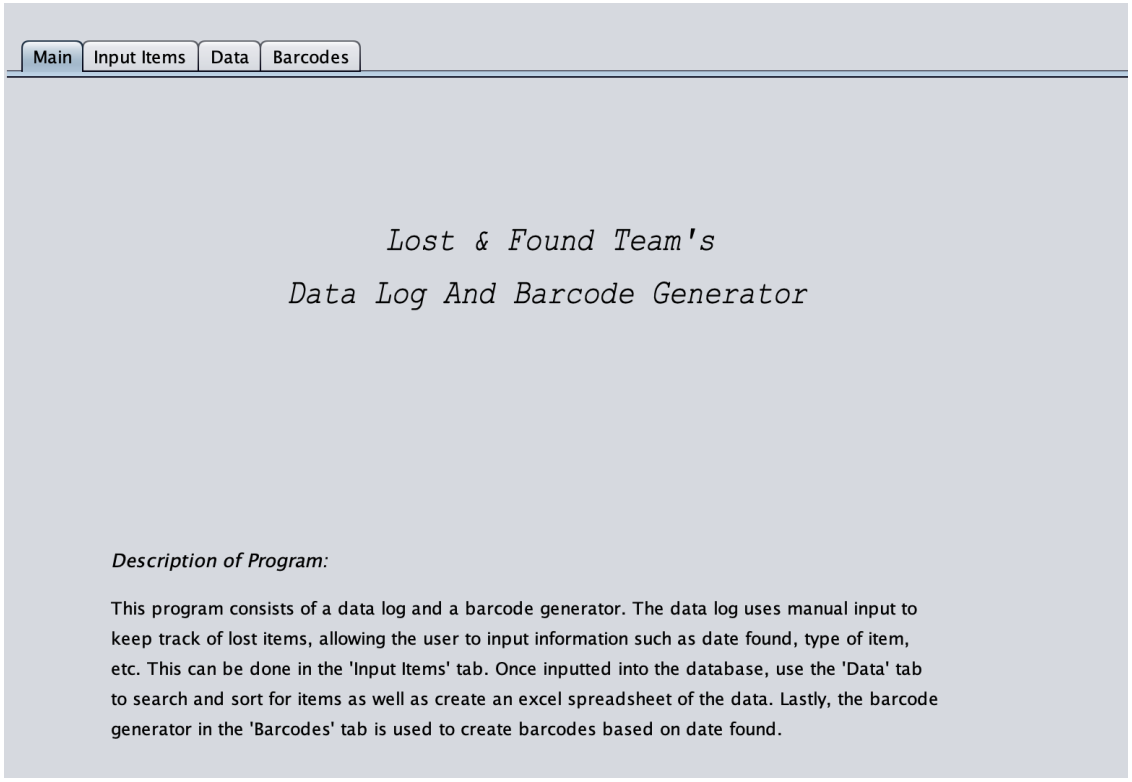
Barcode generator:

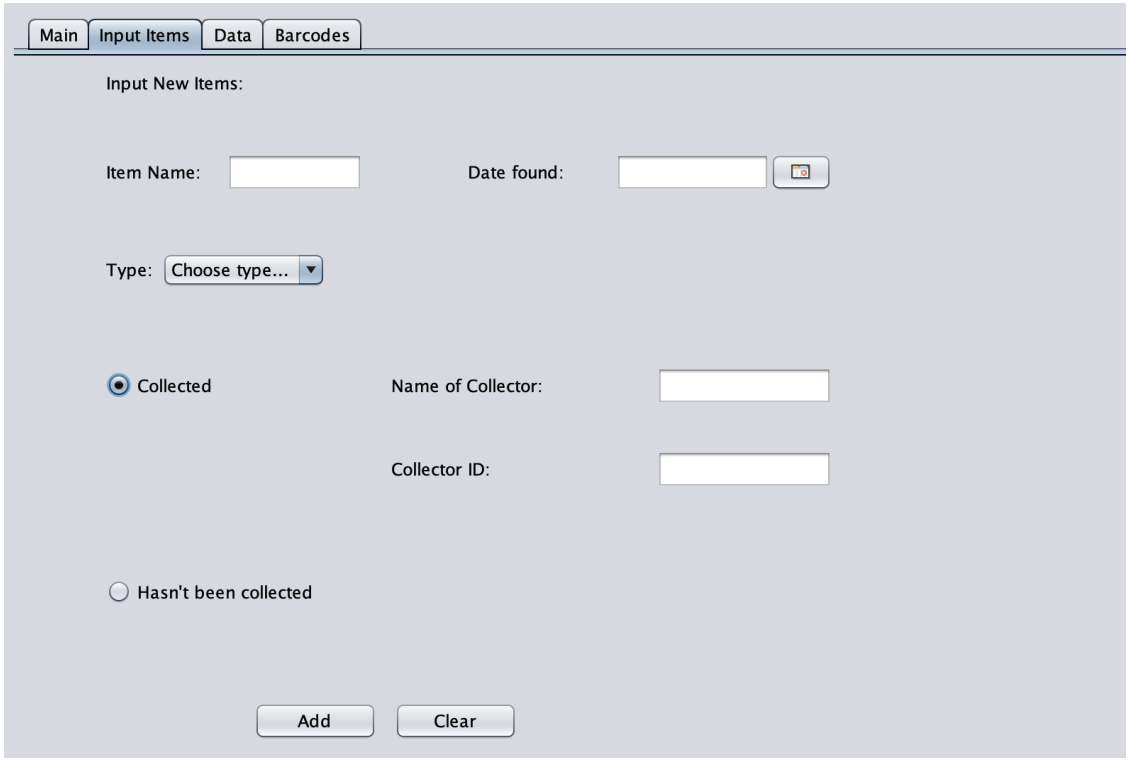
```
Public void createBarcodes(Image code128Image, int numberOfBarcodes, Document doc){
        if(numberOfBarcodes <= 7){
                Create appropriate number of barcodes in first row
        }else if (numberOfBarcodes > 7 and numberOfBarcodes <= 4)
                generate a full 7 in first column
                generate the appropriate number in second column
        }else if((numberOfBarcodes > 7) && (numberOfBarcodes <= 14)){
                generate two full columns
                generate the appropriate number on the third column
}

Public void barcode(String fileName, String toEncode, int numberOfBarcodes){
        Write pdf document doc;
        Barcode128 code128 = new Barcode128();
        code128.Encode(toEncode); // encode the string into a barcode
        Image code128Image = code128.makeImage();

        createBarcodes(code128Image, numberOfBarcodes, doc);
}
```

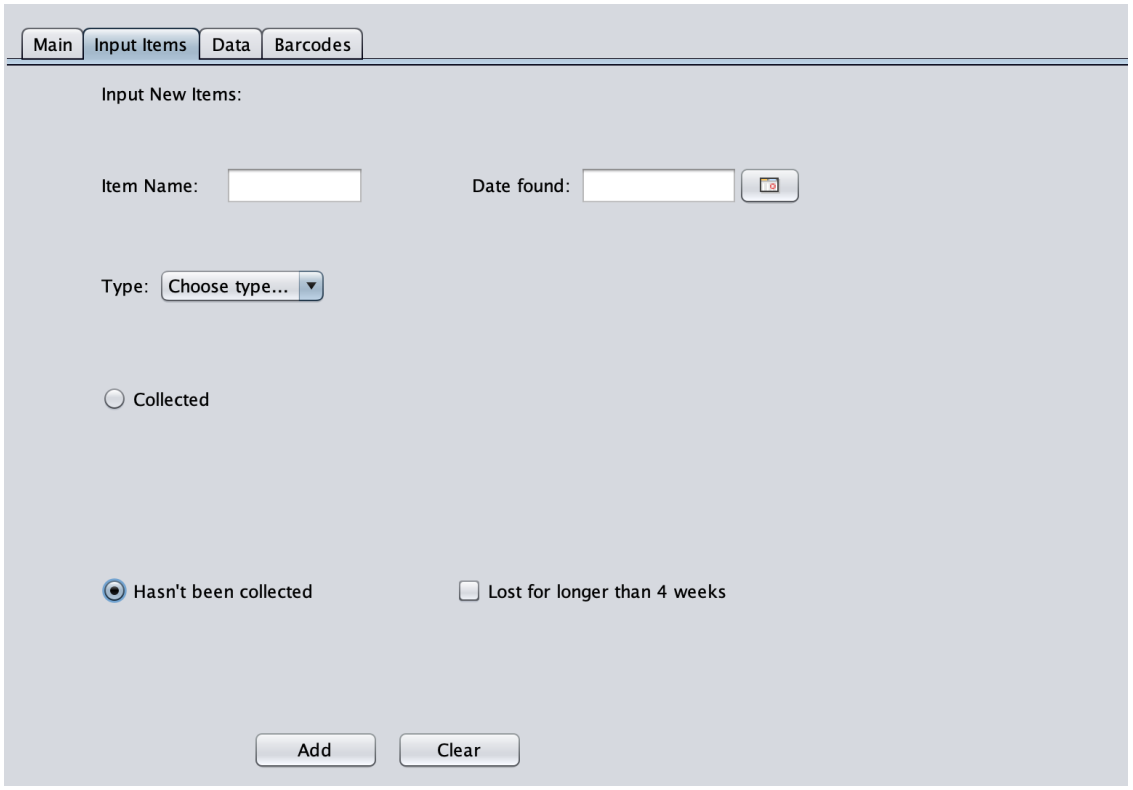
Sorting Algorithm (for item type):

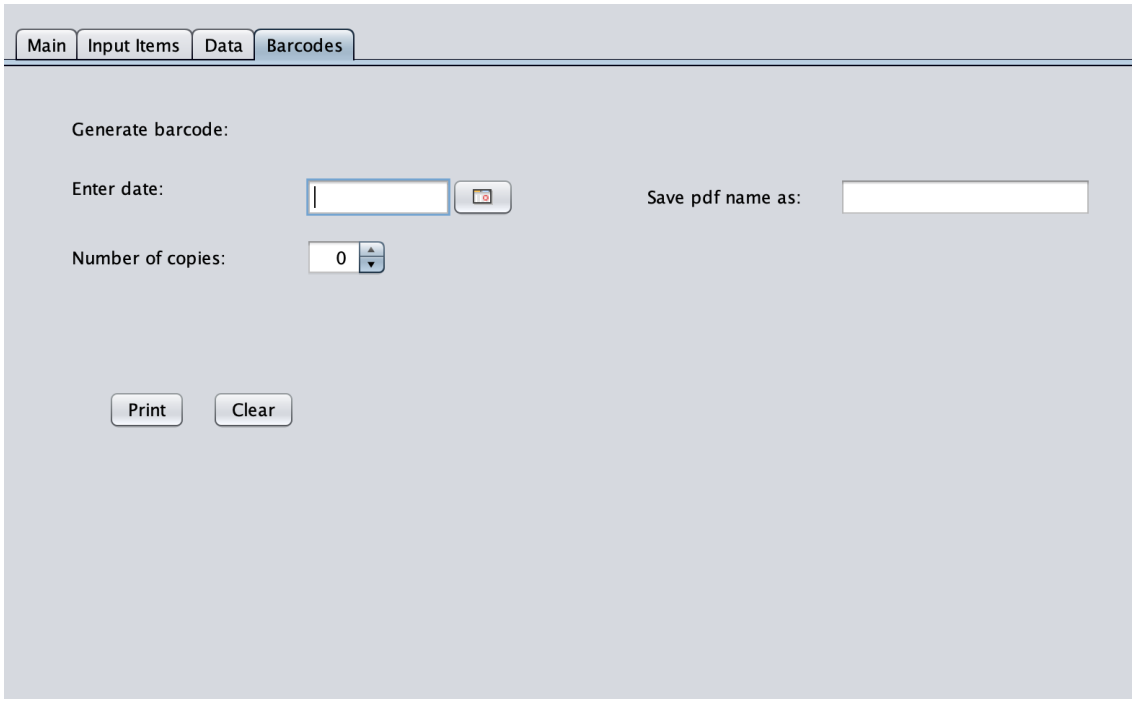
```
int n = counter;
boolean sorted = false;
while (!sorted) {
        n--;
        sorted = true;
        for (int i=0; i < n; i++) {
                if (itemsQueue[i].getItemType() > itemsQueue[i+1].getItemType()) {
                        switchElements(itemsQueue, i, (i+1));
                        sorted = false;
                }
        }
}
```

Final Prototype:









Testing Plan:

Input	Normal	Border	Abnormal	Extreme

Item type combo box input (in 'Input Items' panel)	For example, "Athletic Gear"	First and last options of the combo box	If the user does not select anything but the initial option, the program will select null as input.	No extreme since combo box
Sort by combo box (in 'Data' panel)	For example, "Name"	First and last options of the combo box	If initial value "sort by…" chosen, the program will not sort	No extreme since combo box
Search by combo box (in 'Data' panel)	For example, "Name"	First and last options of the combo box	If initial value "search by…" chosen, the program will not search	No extreme since combo box
Item name	For example, "Blue cap"	Long strings (10 characters or more)	Numbers, symbols	Very long strings (ex. 20 characters), will be accepted but wrapped in the jTable
Date found	For example, 01/02/22	Oldest date (when item collection is kept track of) and current date	If the user does not choose a date, the date attribute will be set to null.	Dates in the future, will not be accepted and an error message will show.
Claimed status	For example, "Collected"	Only two options, so those are the borders	Neither option is selected, the item will not be added	Both options are selected. This cannot occur since the radio button group is made.
Name of collector	For example, "Charlie Parker"	Long strings (10 characters or more), will still be accepted but text wrapped in jTable	Numbers, symbols will not be accepted. An error message with show.	Very long strings (ex. 20 characters), will still be accepted but text wrapped in jTable
Collector ID	For example, 15638	Oldest ID kept track of and most current ID number given. However, this is difficult to keep track of since most constantly update, so will allow.	Non-integer inputs, an error message will show.	Integers that are not five digits (are not valid ID numbers), an error message will show.

(table not included in word count)

Word count: 288