# Homework 2: Citation Support System v.1
# Due: Sunday, October 29, 11:59 pm

You will write the second version of the Citation Support System, building on your Homework 1 solution. Make sure you fix any problems noted in the grading of Homework 1. The user interface and the design have changed. You'll be able to reuse some code from Homework 1 – but note below where specifications have changed. For example, packages are now used to partition the program code. The CitationList class had several methods that did keyboard input and screen output in Homework 1; it does neither in this version – all input and output is done in main( ) and its helper methods.

## Changes to Homework 1 Classes
**Package CitationPackage**

**Class Citation:** adds new member data, userID, so several methods change to account for it.

implements Comparable<Citation>: **CHANGED**: compareTo( ) by Citation number. This is for option 7, sort by number.
int userID: **NEW**: a new data member to store the id of the person in charge of this citation. Create a getter and setter.
Citation constructor: **CHANGED**: add userID to the end of the list of parameters.
toString( ), toCSV( ) – **CHANGED**: add userID to each.

**Class CitationList:** adds new functionality as shown in the revised menu, plus a few other changes.

readCitationFile(String filename) – reads a Citation CSV file. **CHANGED**: read a new field at the end, userID.
writeCitationFile(String filename) – writes a Citation CSV file. **CHANGED**: write a new field at the end, userID.
String displayCitationType(String typeOfOffense) – **CHANGED**: return one String containing any Citations with that typeOfOffense; return "None found" if no Citations match this typeOfOffense.
String dispayCitation(int number) – **CHANGED**: return one String containing the Citation with that id number; return "None found" if no Citation matches this number.
String displayCitation(String lastName) – **CHANGED**: return one String containing any Citations whose Person object has that last name; return "None found" if no Citations match this lastName.
newCitation( ) – **DELETED**; now in hw2Main
String delete(int number) – **NEW**: delete this Citation from listOfCitations. Return "Deleted" if it was deleted, "None found" if not. Menu option 6.
void sortByNumber( ) – **NEW**: sort the list by citation number using Citation.compareTo(). Menu option 7.
void sortByName() – **NEW**: sort the list by last name, first name using an anonymous inner class. Menu option 8.
void sortByType( ) – **NEW**: sort the list by typeOfOffense using TypeSorter. Menu option 9.
String findByUser(int userID) – **NEW**: return one String containing any Citations associated with this userID, if any; return "None found" if none. Menu option 10.

**Class TypeSorter: NEW**
public int compare(Citation, Citation) – **NEW**: used for sorting by typeOfOffense.

**Package src:** New menu items, many changes to main( )
**Class Menu:**
String[] menuChoice: **CHANGED**: value check is between 0 and 10.
static int displayMenu( ): **CHANGED**; new menu options 6-10 added (described below).

**Class hw2Main:** Contains the main program and some data.
private static HashMap<String, String> configData – **NEW**: a map of <config item, value> pairs read by readConfigFile(); use the HashMap methods put(key, value) and value = get(key).
private static CitationList citationList – **MOVED**: now outside of main; the list of Citation objects.
private static UserList userList – **NEW**: a list of the users
public static void newCitation( ) – **CHANGED**: in a new place; prompt the user for the information for one new Citation, including the userID, and add it to CitationList
private static HashMap<String, String> readConfigFile( ) – **NEW**: read the configuration file; hard-code the file name "configuration.csv"; return the hash map

main( ) – **CHANGED**:
1. Read the file "configuration.csv" at the start using readConfigFile() into configData. It contains {key, value} pairs with keys input file, output file, and user file; use the associated values for the file names.
2. Read the citation input file to create CitationList's data; use the file name stored in configData.
3. Read the user input file to create UserList's data.
4. For all display options, String data will be returned here and displayed (instead of being displayed in CitationList or elsewhere).
5. Write the citation output file; use the file name stored in configData.

## New packages and classes
**Package UserPackage:**

**For the User hierarchy,** each class should have an overloaded constructor, getters, setters, and a toString( )
**Class User:** Models one user; abstract
id: the user's id number
String userName: the user's screen name

**Class Administrator:** child of User
String department: where this person works

**Class Officer:** child of User
String rank: one of: Regular, Sargent, Captain, Inspector

**Class CourtOfficial:** child of User
String title: what this person's role in the court system is.

**Class UserFactory:** contains the Factory Method
public static User createUser(int id, String userType, String name, String other) – factory method pattern to create one of the child classes of User using the userType in {Administrator Officer, CourtOfficial}; parameter other's use depends on the type of User.

**Class UserList:** models the list of users
ArrayList<User> listOfUsers: user data
readUserFile(String filename): read the user data into listOfUsers
toString( ): return one String with all the user data
ArrayList<User> getListOfUsers( ): return listOfUsers (for testing only)

## New user interface menu options

6. Delete an existing Citation by number.
7. Sort Citations by number and display them (using toString)
8. Sort Citations by Person last name, first name and display them (using toString)
9. Sort Citations by type and display them (using toString)
10. Display Citations by User – display the userList, prompt the user for a user id, display all citations associated with that user id.

## New data requirements
**CHANGED**: The Citation input file has the format:
```
number,typeOfOffense,description,date,first name,last name,address,phone number,user id
```
as in:
```
1,Speeding,50mph,23-Aug-2023,Mary,Smith,123 Main Street,412-555-5555,2
```
The last value, userID, is a link to the User data.

**NEW**: The User input file has the format ("other" depends on the user type):
```
user id number,user type,name,other
```

as in:
```
1,Administrator,Jones,Cyber
```

**NEW**: The configuration file has the format:
```
key,value
```
as in:
```
input file,citations.csv
```

**Deliverables**: Zip up *only* your .java files; name it <your Andrew id>_hw2.zip; do *NOT* include test files. Upload it to Canvas.
**Hints:** Don't try to code the whole assignment at once. The ordering of the methods above may not be the order you want to code them in, either. If something is unclear, or if you're just wondering about how to do something, ask!

**Grading Rubric:**
1. 80% on console output and test cases (these will be posted later).
2. 5% documentation: document each class and each method. Document your code only where needed – where the person grading it might have trouble understanding.
3. 5% Code quality: variable names, optimal loops, etc.
4. 5% Robustness: your code should not crash while processing reasonable data.
5. 5% Following submission instructions. NO LATE SUBMISSION, PLEASE!