Problem Statement

In CyberCop 3.0, you will add the following new functionalities
1. Handle some runtime exceptions related to data (e.g., Fig.1)
2. Allow the user to make changes to case file and save it as a TSV.
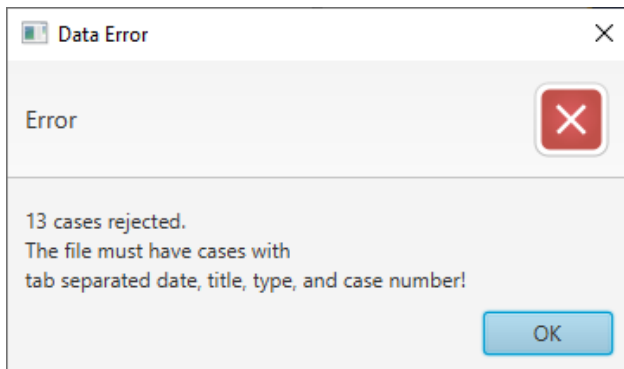3. Attach a class provided to you to display a chart as shown Fig.2.
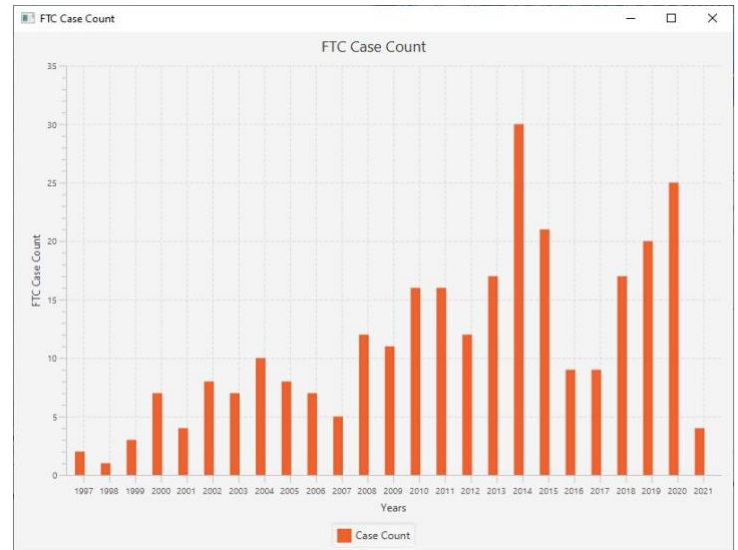

Figure 1 Exception handling

Figure 2 Bar-chart based on FTC case count

For more details on these functionalities, please refer to the demo video-clips provided on Canvas.

Solution Design

A class diagram for Cyber Cop 3.0 is provided in Fig.3 and 4. A brief description of each class and any potential changes that you may need to make are given in Table 1. Note that these changes are based on the design used in HW2. So, your changes may be a little different. In this version, you can make changes to any of the given files, and therefore you must **submit all Java code files** except the Java UT files.

Table 1: Class descriptions

Class name	Description
CyberCop	This will have new handlers and any new bindings, as needed
CCView	A new method showChartView() has been added which needs to be invoked as part of CaseCountChartMenuItemHandler. It is fully coded.
CCModel	New method to be added writeCases(): This method writes caseList elements in a TSV file. If the write is successful, it returns true. In case of IOException, it returns false.
Case	No change
CaseReaderFactory	No change
CaseReader	No change
CSVCaseReader	No change

TSVCaseReader	Handles DataFormat exception when any of first four columns is missing (date, title, type, or case number)
CaseView	No change.
AddCaseView	No change.
ModifyCaseView	No change.
DeleteCaseView	No change.
CCChart	New class provided fully coded to create the chart shown in Fig.2. This class is used by CCView's new showChartView() method.
DataException	New class that extends RuntimeException and shows the message Alert as shown in Fig.1. This exception class will be used in the following cases: 1. In TSVCaseReader: when a case has missing data in first four columns, e.e. date, title, type, and case number. 2. In AddButtonHandler: when the user tries to add a case with either missing date, title, type, or case-number, or the case-number already exists. 3. In ModifyButtonHandler: when the user tries to modify a case in a way that creates the problem mentioned above.

Event Handlers

OpenFileMenuItemHandler	No change
CloseFileMenuItemHandler	No change.
ExitMenuItemHandler	No change.
SearchButtonHandler	No change.
ClearButtonHandler	No change.
CaseMenuItemHandler	No change.
AddButtonHandler	See Table1: Data Exception
ModifyButtonHandler	See Table1: Data Exception
DeleteButtonHandler	No change.
SaveFileMenuItemHandler	Opens the file-dialog box in DEFAULT_PATH for user to enter a filename for the file in which to save the data. It then invokes ccModel's writeCase() method. If the writeCase() method returns true, it displays the "filename saved" message on messageLabel.
CaseCountChartMenuItemHandler	Invokes ccView's showChartView(), passing ccModel's yearMap to it.

CyberCop

DEFAULT_PATH: String = "data"
DEFAULT_HTML: String = "/Cybercop.html"
APP_TITLE: String = "Cyber Cop"
currentCase: Case
ccView: CCView
ccModel: CCModel
caseView: CaseView
cyberCopRoot: GridPane
stage: Stage

main(args: String[]): void
start(stage: Stage): void
setupBindings(): void

Application

...some other methods

start(stage: Stage): void

RuntimeException

CCModel

caseList: ObservableList<Case>
caseMap: ObservableMap<String, Case>
yearMap: Observable<String, List<Case>>
yearList: Observable<String>

readCases(filename: String): void
writeCases(filename: String): boolean
buildYearMapAndList(): void
searchCases(title: String, type: String,
    year: String, number: String): List<Case>

DataException

DataException(message: String)

CaseView

CASE_WIDTH: int = 800
CASE_HEIGHT: int = 500

updateButton: Button
clearButton: Button
closeButton: Button

...all GUI elements

CaseView(header: String)
buildView(): Stage

CCView

webEngine: WebEngine
ccWidth: double
ccHeight: double
isFileOpen: BooleanProperty
... all gui components

setupScreen(): GridPane
setupMenus(): MenuBar
showWebView(): StackPane
showChartView(yearMap: ObservableMap<String, List<Case>>): void

CCChart

xAxis: CategoryAxis
yAxis: CategoryAxis
yearSeries: XYChart.series<String, Number>

updateChart(yearMap: ObservableMap<String, List<Case>>):
    CarChart<String, Number>

CaseReaderFactory

createReader(filename: String): CaseReader

AddCaseView

AddCaseView(header: String)
buildView(): Stage

ModifyCaseView

ModifyCaseView(header: String)
buildView(): Stage

DeleteCaseView

DeleteCaseView(header: String)
buildView(): Stage

CaseReader

filename: String

CaseReader(String filename)
readCases(): List<Case>

YearCount

YearCount (year: String, count: int)
compareTo(yearCount: o): int

TSVCaseReader

TSVCaseReader(String filename)
readCases(): List<Case>

CSVCaseReader

CSVCaseReader(String filename)
readCases(): List<Case>

Case

-caseDate: StringProperty
-caseTitle: StringProperty
-caseType: StringProperty
-caseNumber: StringProperty
-caseLink: StringProperty
-caseCategory: StringProperty
-caseNotes: StringProperty

Case(caseDate: String, caseTitle: String, caseType: String,
    caseNumber: String, caseLink: String, caseCategory: String,
    caseNotes: String)

+getCaseDate(): String
+setCaseDate(date: String): void
+caseDateProperty(): StringProperty

+getCaseTitle(): String
+setCaseTitle(title: String): void
+caseTitleProperty(): StringProperty

+getCaseType(): String
+setCaseType(type: String): void
+caseTypeProperty(): StringProperty

+getCaseNumber(): String
+setCaseNumber(number: String): void
+caseNumberProperty(): StringProperty

+getCaseLink(): String
+setCaseLink(link: String): void
+caseLinkProperty(): StringProperty

+getCaseCategory(): String
+setCaseCategory(category: String): void
+caseCategoryProperty(): StringProperty

+getCaseNotes(): String
+setCaseNotes(notes: String): void
+caseNotesProperty(): StringProperty

+compareTo(case: Case): int
+toString(): String

Comparable

compareTo(case: Case): int

Figure 3 Cyber Cop 3.0 Class Diagram

Legends:
Grey: Library class
Green: Fully coded
Yellow: Partially coded
Orange: To be modified or added new
No color: To be coded
Note that you are free to change any class in this version

The following diagram shows the event
handlers required in Cyber Cop. You may code
them as member classes, inner classes, or
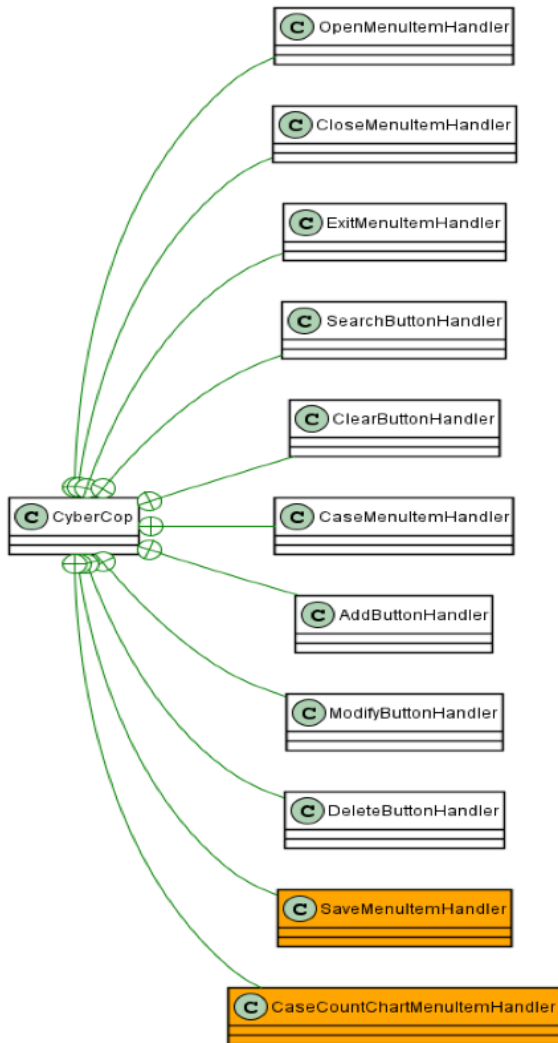lambda functions, as per your preference.

.

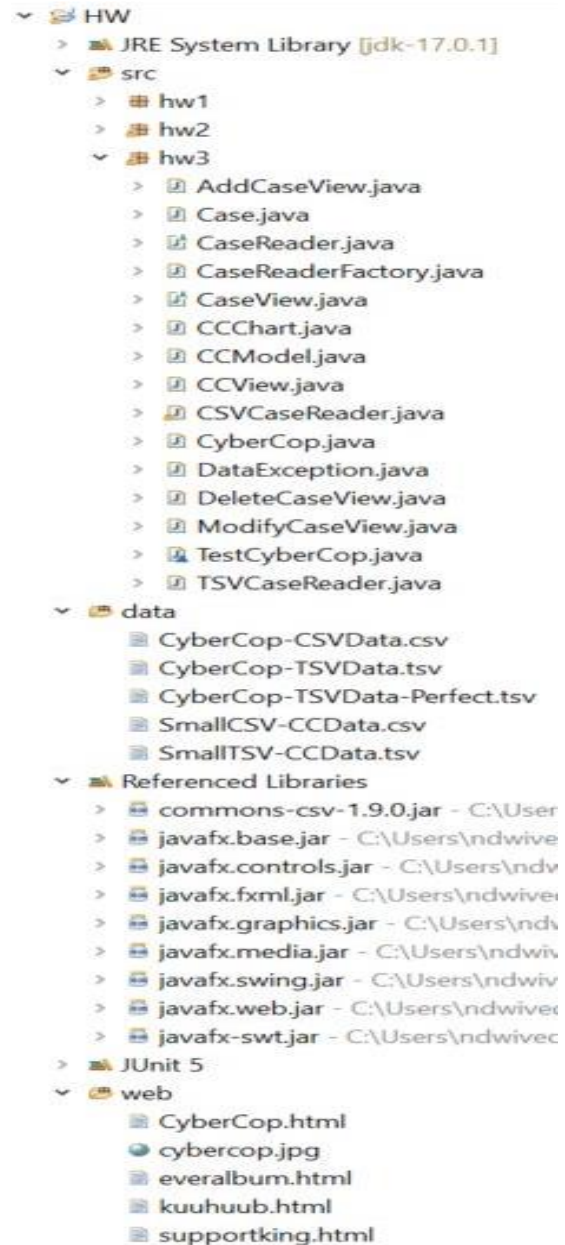*Figure 4 Handlers used in Cyber Cop 3.0*

*Figure 5 Project setup. Note special icon for data and web
folders. They are created as 'source folder'. While not necessary
for data folder, source folders are automatically included in
build path.*

Instructions:

- Download the files from Canvas and store them in a project and a package as shown in Fig. 6.
- Add the following VM options to your run configuration: --module-path="YOUR PATH TO JAVAFX LIB" --
  addmodules=javafx.controls,javafx.fxml,**javafx.web**
- Complete your code.
- You can modify any class (except test file) in this version. And you must submit all Java files (except test file)
- Write your name and Andrew id as comments at the top in all Java files.

- You will test your program in two ways: GUI interaction (refer to scenarios in video-clips on Canvas) and TestCyberCop.java. These two tests will get you 80% of the points.

Other criteria applied to evaluate your program are:

1. Documentation (5%): Your code should be well-commented, i.e., neither too many comments, nor too few. Yes, this requires a little bit of your judgment! Name your variables in a selfexplanatory way. Write your name and Andrew id at the top in the comments in each class. Indent your code properly. (In Eclipse, press Ctrl-A to select all your code and then Ctrl-I to indent)

2. Code quality (5%): coding conventions, no unused variables/libraries, etc. Use your judgment to assess these criteria.

3. Code robustness (5%): Your program should not throw any errors while processing. You can safely assume that the user will not enter any garbage input.

4. Submission instructions (5%): Zip your java files into AndrewId- hw3.zip. Do not submit any other folders, class files, test file, text files, and rest of your kitchen sink! Only last submission will be graded. Wrong files, incorrect package name, etc. may cost some points.

NO LATE SUBMISSIONS PLEASE! If you are unable to submit on time, you lose all the points. Please avoid last minute submission as Canvas may decide to quit on you! Learn to trust technology only to the extent you should! Do not take that risk! **I will not accept late submission. Good luck.**