Essentially the design is such that one can use the ClaimController to
access everything, with a given index. When a claim is created it is added
to the ClaimList, which is a property of the CLaimController. When a claim is
viewed or edited, it is accessed via the claimController's ClaimList methods.
The same goes for ExpenseItems, but via the Claim class's methods. This way
ensures one can access any claim at anytime, through the ClaimController.

**Because I failed to implement Serialization, I did not include all classes
seen in the project.

Lazy Singleton controller,
used to access claims throughout
application.

ClaimController

+claimList: ClaimList

+addIt(c:Claim): Claim

INTERFACE Listener

+update(): void

The ClaimList class serves a useful
purpose as it holds an ArrayList of
all the claims together in one place.
It has methods to change the list contents
as well as notify a listener to when the
list has been modified.

ClaimList

+claimList: ArrayList<Claim>
+listeners: ArrayList<Listener>

+addClaim(claim:Claim)
+removeClaim(claim:Claim)
+size(): int
+notifyListeners(): void
+addListener(): void
+removeListener(): void

Claim also implements
generic getters and setters
for all attributes

0...*

Claim

+Name: String
+dateFrom: Date
+dateTo: Date
+Description: String
+eItems: ArrayList<Expense_Item>
+Status: String
+dateFGiven: String
+dateTGiven: String

+toString(): String
+toEmailString(): String

The Claim class is mainly
responsible for storing information
about individual claims. It contains
methods to perform operations on its
data, such as manipulation and formatting

Expense Item also
contains generic
getters and setters
for each attribute.

0...*

Expense Item

+item: String
+date: String
+category: String
+description: String
+amount: int
+currency: String

+toString(): String
+toEmailString(): String

This class keeps track of information for
individual Expense items, for which a claim
can have many.It has methods for manipulating
its data as well as formatting.