Practical exercise 3

Expend our Employee Expenses Management Application project as follows:

Earlier in the course we created a class called ExpenseItem – this class is intended to represent the different elements that make up an expense claim. For example a member of staff who takes a business trip might make a single expense claim covering the cost of travel, a hotel, and food. Each of these different costs will be a separate expense item.

1. Amend the ExpenseClaim class so that it contains a list of ExpenseItem objects.
2. Add a method to the ExpenseClaim class to allow an expense item to be added to it.
3. Remove the totalAmount field of the Expense claim class and replace its get method with a calculation which sums the amounts of each ExpenseItems. Don't worry about the fact that you are summing Doubles here rather than BigDecimals, so it may not be completely accurate. Remove the total amount from the toString method, and ensure you regenerate equals and hashcode.
4. Add a method to the ExpenseClaim class which will print out the expense items to the console.
5. Adjust the console application so that when you create a new expense claim, you are asked to enter the expense items.
6. When you print the employees, change this so that it prints each employee, then a list of their claims and for each claim a list of their claim items.

Another team will be building some separate code which will provide various analysis and filtering functionality for the expense claims in the system. We will not get access to this code, so for now we will do the following:

7. Create an interface in the utilities package called `ExpenseAnalysis` to act as the contract between ourselves and the other team. The interface should provide the following methods. These are all void methods.
   - Print a list of all outstanding expense claims (will not take any parameters)
   - Print a list of all paid expense claims within a date range (will take 2 LocalDate objects as parameters)
   - Print all expense claims where the amount claimed is above a specified amount (will take a double parameter)
8. Create a temporary implementation of the interface called `ExpenseAnalysisTempImpl` that for each method simply prints "this feature is not currently available" to the console. We'll then replace this implementation with the real one later on.
9. Use the temporary implementation to integrate the functionality into our console application. Do not worry about validating the data input as part of this process