Homework 3

Use Case Diagram

Due by Feb 2nd 2018 11:59 pm

Related Readings:

Reading 11 - Larman on use cases
UML Notations - "Diagrams" section in this link and Chapter 9 "Use Case Diagrams", in the same book.

Definition of basic actors w.r.t use case: 'Actors' are not only used for people involved in the system. An Actor could be anything which interacts, initiates, or is closely connected to the events in the use case.

Primary Actors: Primary actors are actors with goals and who start the chain of events leading to execution of a use case. A primary actor might use services provided by other actors to achieve the goal. Don't overthink the distinction between primary, secondary, and intermediate actors. (See Chapter 9 above for Fowler's comments.)

Secondary Actors: Secondary actors are those that provides some service. Example - in Thermostat displaying temperature - the thermal sensor is a secondary actor which provides with temperature, when queried. Inanimate sources of events (like the thermal sensor) are also sometimes assigned to a class of actors known as "adjacent systems".

Stakeholders or Offstage actors: Stakeholders in a use case are those which get affected by the outcome of the use case. It could be the utility meter which records the usage. In battery warning, it could be home user who is notified of the low battery. Do *not* expect that every stakeholder (or even most stakeholders) will appear in the use case diagram.

Things to turn in :

The EvalSorts program we have been using as an long running example needs to be redesigned. Please read through the description of the desired **Sorting System** (the next version of EvalSorts) below and submit the following

1. A list of scenarios with a short description (less than 3 sentences) for each. Be sure to include the actor/role, goal, and name of the associated use case in your description (so that we can easily connect the description to the appropriate part of the use case diagram.)
2. A use case diagram for the system.

Sorting System and its users:

Scientist: The owner is a computer scientist who has a grant to study subtle, data-sensitive behavior of various sorting routines. He commissioned the original EvalSorts program before he received the grant to make it easy for him to run exploratory experiments with a few standard sorting algorithms. Now, with the grant funds available, he wants to greatly increase the volume of tests and the number of comparative runs executed. To that end, he has hired a research assistant to manage the execution of the tests. At this point he wants to be able to tell his research assistant to prepare data with certain characteristics (to be placed in data files), and then have the assistant run a sort comparison for each those files (maybe hundreds of different files). The research assistant will then perform various analyses on the results from those runs, perhaps make small changes and then repeat the set of experiments again.

Research assistant (RA): The RA is the primary direct user and experiment observer. The research assistant likes being able to interactively run tests on a single data file, but would also like to be able to run tests (or some subset of the tests) as large batch operations. He thinks the ability to generate data sets randomly is less useful (since his role in the research is focusing on the effects of specific patterns in the input data), but, small, one-off generated data sets are still good for validating that sort-eval is working correctly.

After consulting with the RA and departmental tech support, the analyst and the stakeholders have agreed that if the program supported the following set of command line options, it would better address their needs:

- no command line options (i.e., just >evalsort): the program would behave as it does now.
- -g -n dddd -s xxxx : generate a data set randomly, using long seed xxxx, and including dddd values to sort.
- -f datafile: run a single experiment with the data in datafile
- -b batchfile: read file "batchfile", and interpret each line as command line options describing one experiment to be run. For example a batch file containing:

  ```
  -f data01
  -f data02
  -f data03
  ```

  Would run EvalSorts three times, once for each of the three data files named.

Since the research team plans to perform extensive testing of the sorts, they would like to have results from batch runs be written to a file in formats that support visualization and analysis through spreadsheet applications and various special utilities. Thus they would like two additional options:

-o filename.csv
-a filename.csv

The -o option places the results in a new file named "filename.csv". The -a option appends the results to an existing file. In both cases, the output should be in comma separated variable format that loads transparently to excel.

Finally, if running interactively (with manual input), the users would like to be able to request that screen output be captured to a log file. Thus, if invoked with an argument of the form
"-log fname",
the program will ignore all other options that might be present, and will run in interactive mode (as if no arguments had been supplied), but will log all *standard out* and *standard in* to the file "fname".