PAM: A Swing Personal Activity Monitor

Update #1  You can use Java Calendar or Date class.

Design and implement a simple, easy to use, prototype for a Personal Activity Monitor (PAM) application using Java / Swing. Life can be thought of as a “project”. In project management one often monitors (records) the effort and result of critical project components. Watts Humphrey has ported software project management to the person. His PSP (Personal Software Process) is an interesting read. Our PAM project is weakly inspired by PSP and more focused on life than transitory projects. PAM should monitor personal goals of the user. These goals should be record (date, activity, numerical value, description, and comment/evaluation) daily with summary statistics for the day, week, and log.

For example, my PAM would monitor diet, exercise, meditation, work, and sleep. These are life activities (habits) I’d like to know more about. On a daily basis I’d like to know how much I: eat (calories, meal), exercise (calories, type), meditate (frequency, time), work (task, time), and sleep (time). The daily statistics might be calories (food calories – exercise calories), exercise (hours) meditation (hours), work (hours), and sleep (hours). The weekly statistics might show daily statistics by day of current week, with totals and weight. The log might show all the weekly statistics (with dates). The numerical values can be approximations. For example calories in units of 50 and time in multiples of 0.5 hours.

You can design a PAM for your own life tasks, or make up some (eg. music practise). There should be at least 3 activities your design can monitor. I advise against making a lot of activities (say more than 5). Remember, this assignment is a prototype.

PAM should have the following features:

- Should use JTable to maintain an ordered set of information consisting of at least 3 fields. The user should be able to enter, delete, and edit data easily -- consider using table column editors.
- There should be three views of the data: day, week, log. Each view should also display summary information. There could be multiple instances of some views. For example, there could be 7 day views (Mon, Tues ... Sun).
- PAM must have date information and must use java Date or Calendar. You should also look into the SimpleDateFormat, and System classes; you might want to use them. (See third party UI Controls.)
- Since it is a prototype, you can hard code the activity labels (categories), or you can make them user editable (higher grade).
- PAM should be private – personal information should be hidden. The application should have a password using JPassword. Passwords could also support multiple users – but not for the prototype. With your submission there should be data for one users (provide the passwords) with data for 3 weeks. There doesn’t have to be a lot of data.
- PAM should maintain a persistent storage of user data that can be opened and saved automatically (constructor/destructor) or with a file dialog. (This is not a database, or web-services class. Do not use a database for persistence.) You must use Java for persistence (file i/o, serialization or XML encoding/decoding).
- There must be a "Usage" and "Author" menu item. The Usage menu displays information on how to use PAM. The Author menu item displays information about the author (name, email, and image).
- There must be a comment, “diary” that uses a scrollable JEditorPane. This can be an additional “Dairy” view or can be part of one of the other views.
- You can add other information or features to your PAM if you choose.

Design Before Implementation ...

John Culkin (1967) writing about Marshall McLuhan’s concepts in Understanding Media (1964) stated, “We become what we behold. We shape our tools and then our tools shape us”. The use of interface builders (IDEs) like Java’s NetBeans that allow you to "drag and drop" components to make an interface are not necessarily the best way to learn how to design and develop GUIs. These tools facilitate rapid application development for professional developers. Use them if you wish, but be sure you understand what code they are generating. I suggest you have a GUI design first and then use the tools that allow you to implement your design. If this is your first (or nearly first) GUI project consider designing and building your application by hand with an IDE.

For your initial design consider drawing sketches of the interface’s view. You could use paper and pencil or drawing tools (like PowerPoint). GUI applications are state based (events change states), make an event-state transition graph as part of your design. Use simple UML class diagrams (Violet in a simple UML tool). Consider design and implementation as an iterative (spiral), expanding process. Do not initially make an overly complex or complete design and then start to implement it. Instead prototype a simple design so you can understand the time and complexity involved in GUI implementation. As you understand the GUI "media" you can design and implement more complex parts of the assignment. I strongly recommend that you develop a design / implementation plan.

Third Party UI Controls.

You must talk with me and get my permission to use third party UI controls. These are controls outside the Oracle Java / Swing API. An example third party UI control that I will accept is a dateChooser. You must cite and reference the use of all accepted third party UI controls in your design documentation and in beginning comments of source files that use them.

Managing Views

Views could be managed via a view menu, or, with a JTabbedPane. Views should be scrollable where applicable. I suggest you start with the day view and work out data persistence possibly using the JFileChooser (optional). Once you have all that stuff working you can then start to implement the other two views of the data.
Persistence of data

A user's data should be stored in a subdirectory relative to the application's directory. How the data is stored depends on the type of table model you use/write and your approach to persistence: serialization (ObjectOutputStream, ObjectInputStream) or XML (java.beans.XMLEncoder XMLDecoder). The data can be stored in one or several files. How this is done depends on how you create the "models" of your application's data. The simplest model is a vector that contains each entry in the diet (date, day of week, meal, description, and calories). This vector could be serialized to a file. Read the API documentation on JTable concerning serialization. Persistence of data should help in the development/testing of your assignment: you won't have to enter data every time!

Implementation

You can work in groups of 2 if you wish. Each group has one submission and grade. If a member is not contributing their "fair-share" to the project the group can divorce before September 30. The group must share all work done up to the divorce. The divorcing member should send me and copy the other member an email requesting a divorce by 9/21/2014. After 9/21/2014 groups can't divorce. This is a “no fault” divorce class. I suggest the group talk, or talk with me, before a divorce to see if they can reconcile (do this before 9/14/2014).

Submission

There is no printed submission. Submit all your work as a single compressed ("zipped") file on Moodle.

Your submission must include:

1. The executable application, PAM data (persistence), and all source files. I do not use Eclipse. I have JGrasp and NetBeans installed on the system I will grade your work. If you use either of those IDE’s you can submit your project directory. Be sure to have no version control directories or system-dependent links in your submission. All paths (file names) should be local to your submission’s “root” directory.
2. UML “lite” class diagram – showing classes, important fields, important methods (this can be hand drawn, drawn in paint or PowerPoint program like the class notes, or with a tool like Violet or Visio.
3. A "top level" event state transition table or diagram. This can be a UML state transition diagram.
4. A brief description of how to run your program.
5. Screen capture images of the 3 views.
6. Any other information documentation you actually used to develop the application.
7. Any features you want to point out for my appreciation/evaluation.

If you are pointing out parts of your project for me, please also have appropriate comments in your source files so I can search for the relevant code. Be sure the names of all group members
are in (on) all materials. For class files this should be as an introductory comment block at the
beginning of the file.

I must be able to run your program -- this is your responsibility. I don’t want to be compiling
(and/or debugging) your programs at the point of evaluation. You must test that your submitted
program runs on a system different from the one you developed it on. You cannot assume I have
any interface builder installed on the system I will grade your assignment on. I have a standard
jdk1.8.0 installation. I will recompile your submission. If you use a non-Oracle java compiler
(like with a Linux distro) you must follow Java’s style of capitalizing the first letter of Classes.

If you have questions concerning these specifications please ask in class, on the course’s Moodle
forum, or in email. I reserve the right to post relevant emails and replies on Moodle for all
students to read.