TP4 : Introduction to software traceability with Spoon

Exercise 1 : Logging with Spoon for profiling

Question 1

Create a simple application (with a CLI or a GUI) that allows you to :

1. create a user with an ID, name, age, email, and password.
2. provide a user with a menu through which (s)he can :
   - display products in a repository, where every product has an ID, a name, a price, and a expiration date.
   - fetch a product by its ID (if no product with the provided ID exists, an exception must be thrown).
   - add a new product (if a product with the same ID already exists, an exception must be thrown).
   - delete a product by its ID (if no product with the provided ID exists, an exception must be thrown).
   - update a product's info (if no product with the provided ID exists, an exception must be thrown) .

Question 2

Choose a logging utility among the following and play around with it a bit to understand its different facets :

1. **The Java Logging API** (recommended, check JavaLoggingAPIBooklet.pdf on Moodle)
2. **Log4j** : official website
3. **SLF4J** : official website

Question 3

Use Spoon to trace the code of your application using the logging utility of choice, such that the generated logs can be leveraged to create profiles of users as follows :

- a profile for those that mostly performed read operations on your repository.
- a profile for those that mostly performed write operations on your repository.
- a profile for those that searched for the most expensive products in your repository.

The definition of your profile's structure is up to you. The most important information to include is the user's info, and the marking features of your profile (*e.g., the read/write operations performed by a user for the mostly performed read/write profiles*).

Question 4

Define and execute a sequence of execution scenarios with different users to generate your logs. For example, you can create 10 users, and let each user execute around 20 different scenarios involving the above operations with different input values. Make sure that your scenarios are diverse enough to simulate a real-world experience to have properly crafted profiles at the end.

Question 5

Propose a way to parse the generated logs and extract the required information to construct your user profiles. Note that structured logs require the reification of an LPS and the definition of its construction and printing mechanisms (*e.g.,* `Timestamp: <date_time>, Event: <event_info>+, User: <user_info>+, Action: <method_info>+, etc.`). In this case, each part of the LPS' content can be built separately and then aggregated to form the resulting LPS (*think about the `Builder` Design Pattern*). You can also choose to store and display them as JSO[illegible]es if you wish.

Useful links

How to choose levels of logging