UML: Activity & Sequence Diagram

Imane Fouad, UM6P

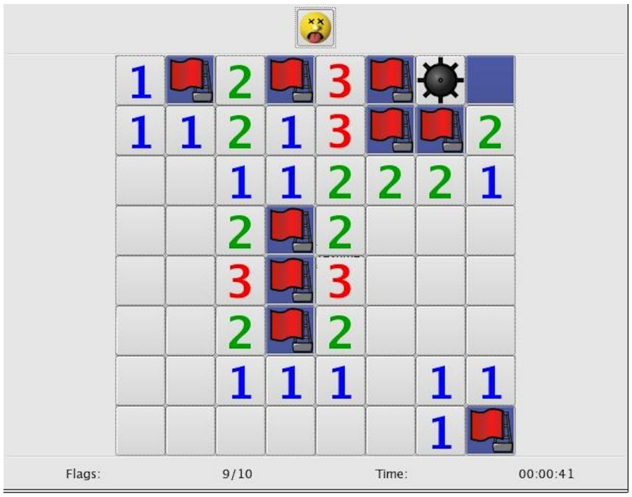
Exercise 1:

To create a repair record, the workshop manager enters the car search criteria into the system. The repair management software gives him the list of cars matching the entered criteria. If the car exists, the workshop manager will select the car. The software then provides the information about the vehicle. If the car is under warranty, the manager must enter the repair request date. If the car does not exist, the manager will enter the information regarding this new vehicle. In all cases, the workshop manager must enter the reception and return dates. If the car damage is paid by the insurance, the software provides a list of insurances to the workshop manager. The manager selects the appropriate insurance. Finally, the software records the repair form.

Question: Model this process using an activity diagram (Use partitions).

Exercise 2:

The goal of this exercise is to model the behavior of a Minesweeper game when a player reveals a cell. Minesweeper is a puzzle game in which the player uncovers cells on a grid, trying to avoid hidden mines while revealing all safe cells.



Flags: 9/10 Time: 00:00:41

This exercise **focuses only on the action of revealing a cell.** Depending on the cell's content:

- If the cell contains a mine, the game is immediately lost.
- If the cell is numbered, the number is revealed and the game checks whether the player has won.
- If the cell is empty, the game recursively reveals all neighboring cells.

Question: Build a simple sequence diagram to illustrate these interactions.

Exercise 3:

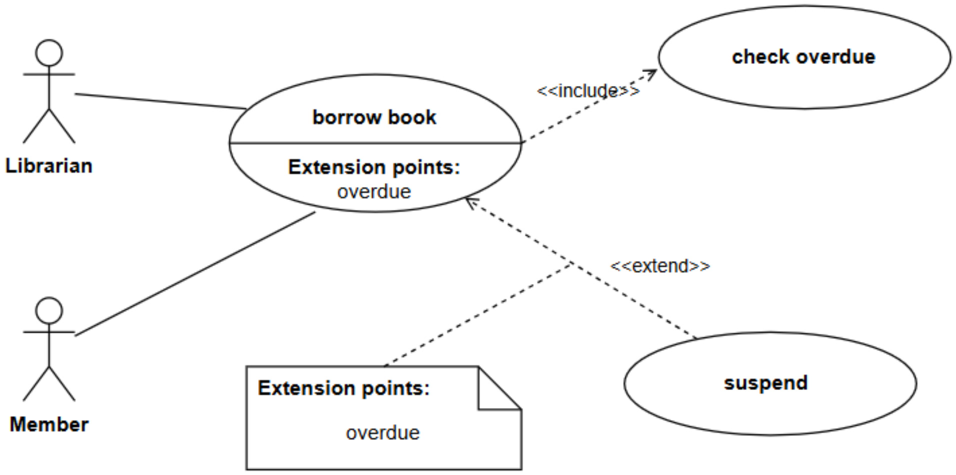
Here is a simplified recipe for chocolate mousse:

- Start by breaking the chocolate into pieces, then melt it.
- Concurrently, break the eggs separating the whites from the yolks.
- Once the chocolate is melted, add the egg yolks.
- Whisk the egg whites until they are stiff.
- Gently fold them into the chocolate mixture without breaking them.
- Pour into individual ramekins.
- Chill for at least 3 hours in the refrigerator before serving.

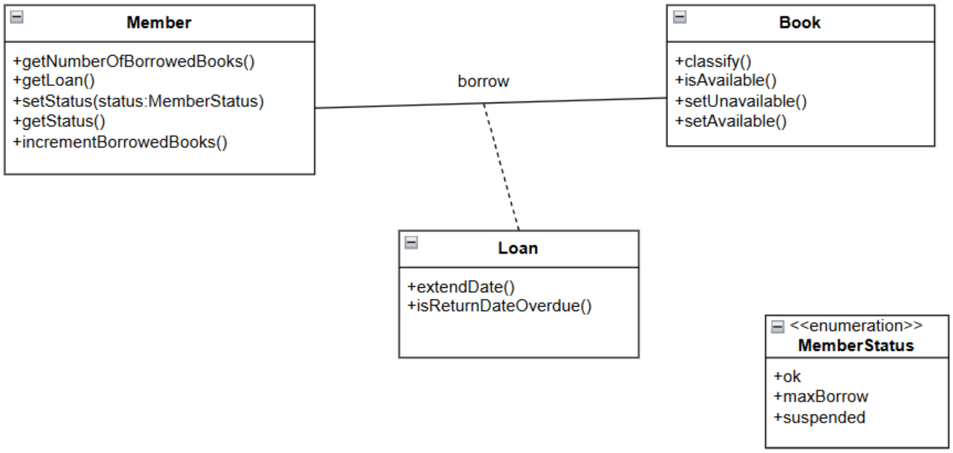
Questions: Represent the chocolate mousse recipe with an activity diagram.

Exercise 4:

We are interested in the **borrow** use case, which uses the **check overdue** and **suspend** use cases of a library management software.



In the following, we present part of the class diagram:



Scenario: check overdue

The application checks if there is any overdue in the member's loans:

1. It retrieves all the loans of the member.
2. For each loan, it checks if the return date has passed.
3. If a return date has passed, it sets the member's status to suspended.

Scenario: borrow a book

- The member goes to the counter, and the librarian selects the borrow functionality in the application.
- First, the system must check if the member has the right to borrow books (is the member already suspended? Can they still borrow books?).
- Then it checks if there are any overdue books and updates the member status accordingly.
- Next, it must verify that the book is available.
- If everything is fine, a new loan is created with the loan date and return date, associated with the member and the chosen book.
- The book is marked as unavailable.
- The number of books borrowed by the member is incremented.
- The member's status is updated.

Questions: Build sequence diagrams for the **Check Overdue** and **Borrow a Book** use cases.