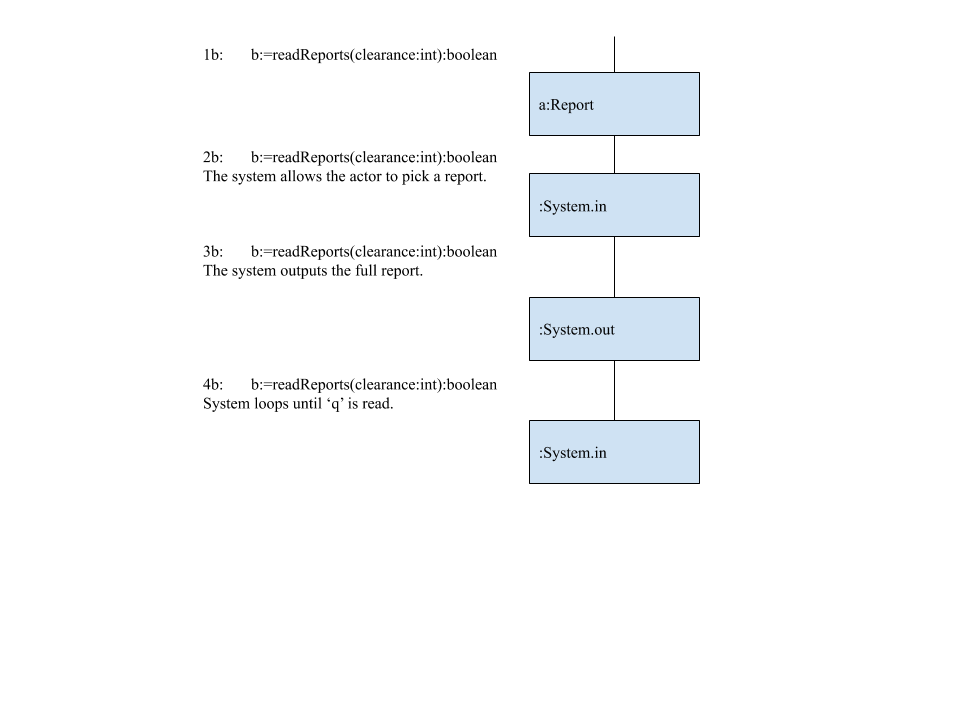
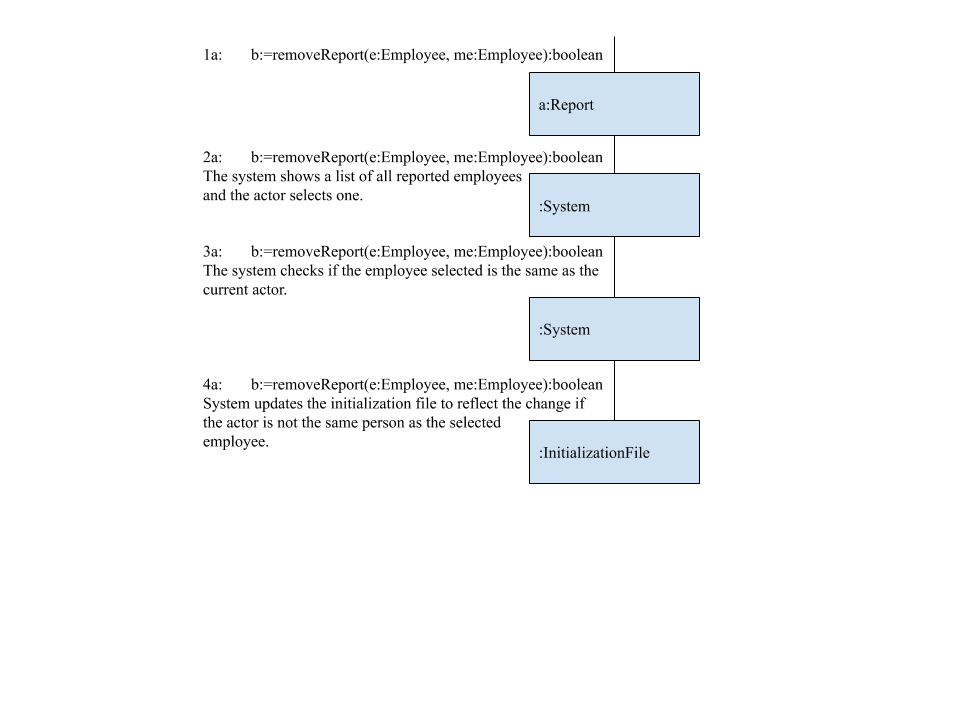
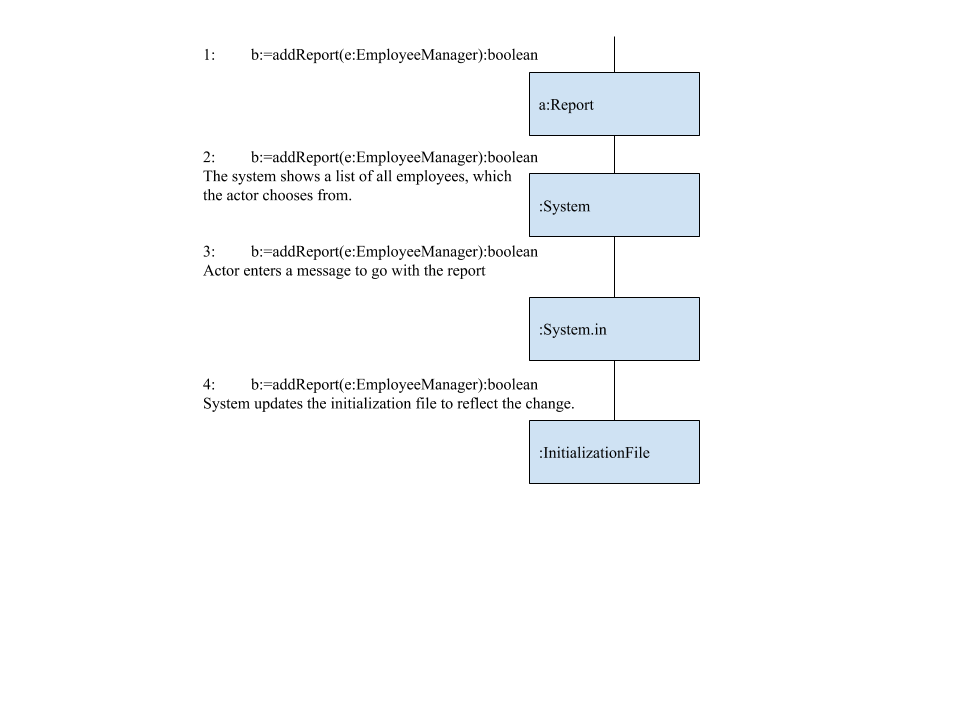
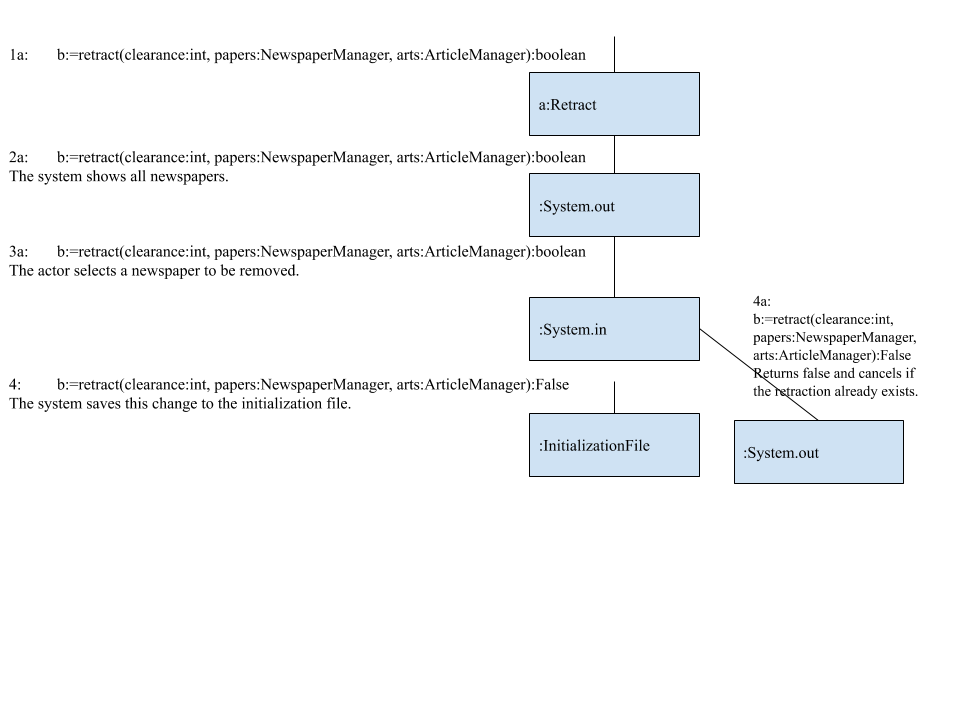
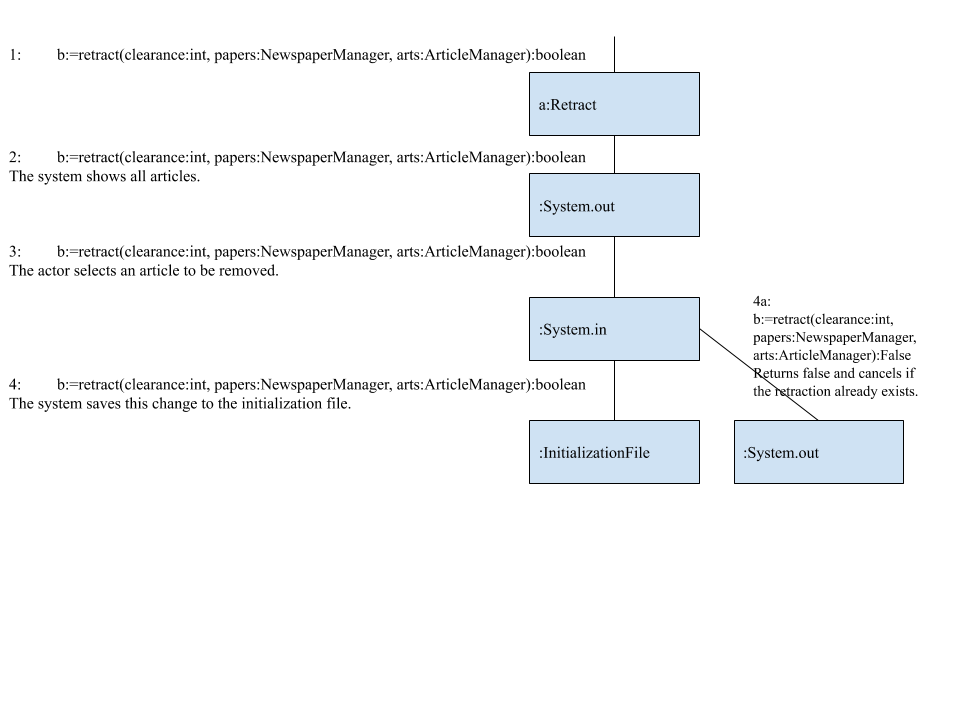
|  |  |
| --- | --- |
| Use Case: | Add/remove/read reports on Employees |
| Actor: | Employees |
| Precondition: | The database exists how it was initially designed, and the actor has clearance to do the command. |
| Main Success  Scenario: | 1: The actor chooses to report another employee.  2: The system shows a list of all employees, which the actor chooses from.  3: The actor enters the message that goes with the report.  4: The system updates the initialization file. |
| Extensions: | 1a: The actor chooses to remove a report.  2a: If 1a occurs, the system shows a list of all reported employees.  3a: If 1a and 2a occur, the system looks for if the employee selected is the same as the actor.  4a: If 1a, 2a, and 3a occur, the system updates only if the selected employee is not the actor.  1b: The Actor chooses to read reports.  2b: If 1b occurs, the system allows the actor to pick a report.  3b: If 1b and 2b occur, the system outputs the whole report for the selected employee.  4b: If 1b, 2b, and 3b occur, the system loops until only ‘q’ is read on a line. |



|  |  |
| --- | --- |
| Use Case: | Retract a newspaper/article |
| Actor: | Employees, Managers, Owners |
| Precondition: | The database exists how it was initially designed, and the actor has clearance to do the command. |
| Main Success  Scenario: | 1: The actor chooses to retract an article.  2: The system shows all articles.  3: The actor chooses to retract an article.  4: The system saves this change to the initialization file. |
| Extensions: | 1a: The actor chooses to retract a newspaper.  2a: If 1a occurs, the system shows all newspapers.  3a: If 1a and 2a occur, the actor chooses to retract a newspaper.  4a: If the newspaper/article has already been retracted, then it won’t save another retraction performed on it. |

\*\*NOTE\*\* Searching, reading, editing, etc. newspapers and articles have been changed to disallow access if the newspaper or article was retracted to complement/reinforce this use case.



Class Diagrams:

|  |
| --- |
| Report |
|  |
| b:=readReports(clearance:int):boolean  b:=addReport(e:EmployeeManager):boolean  b:=removeReport(e:Employee, me:Employee):boolean  h:=getReports():HashMap<Employee,String> |

|  |
| --- |
| Retract |
|  |
| b:=retract(clearance:int, papers:NewspaperManager, arts:ArticleManager):boolean  a:=getArticles():ArrayList<Article>  a:=getPapers():ArrayList<Newspaper> |

Justification for Report class: Has access to the database, information stored in the EmployeeManager class, and its stored Employees. It needs access to the database so that it can consistently start-up and initialize it from its file. It needs to see EmployeeManager and its Employees because it uses those Employees as its data. The class should exist on its own because reports on employees should not be stored close to them to discourage the mass deleting of reports.

Justification for Retract class: Has access to the database as well as the information stored in NewspaperManager and ArticleManager. It needs access to the database so that it can consistently start-up and initialize it from its file. It also needs to see the information in NewspaperManager and ArticleManager because it uses the objects stored in them. The retracted class should exist on its own because when you think of a retracted paper or article in real life, you think of it as something completely different from a normal article or paper. It is thought of as a mistake more than a manipulable object.