CMPS 356 Enterprise Application Development - Spring 2018

**Lab 5 – OO JavaScript**

|  |  |
| --- | --- |
| **Student Name** | Riyam Al-Dawood |
| **Student Id** |  |
| **Email** | Ra1105618@qu.edu.qa |

**Grading Rubric - - In the *Functionality* column please specify either: *Working (completed x%)*, *Not Working (completed x%)* or *Not done*.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | % | Functionality\* | Quality of the implementation | Score |
| * Part A |  |  |  |  |
| * Created the structure described in Questions 1 to 4 | 20 | 100% completed |  |  |
| * Created the following methods   add(account)  getAccount(accountNo)  deleteAccount(accountNo)  avgBalance()  sumBalance()  serialize()  deserialize(accountsJson) | 20 | 100% completed |  |  |
| * Q6. Tested | 10 | done |  |  |
| * Part B |  |  |  |  |
| * Created the structure described in Questions a and b | 10 | 100% completed |  |  |
| * Implemented the following methods in Q.c   addBook  updateBook  deleteBook  getBooks  getDonorBooks | 30 | 90% completed  Except getTopDonor method |  |  |
| * Tested the code | 10 | completed |  |  |
| Total | 100 |  |  |  |
| Copying and/or plagiarism or not being able to explain or answer questions about the implementation | -100 |  |  |  |

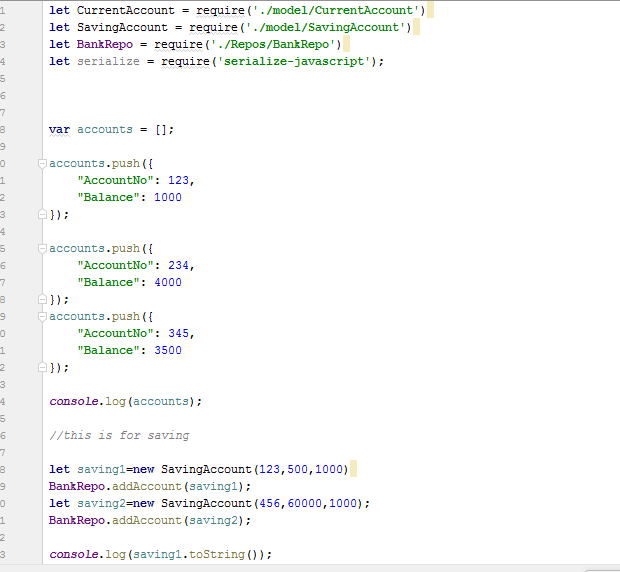
**\* Possible grading for functionality**: ***Complete and*** ***Working*** (get 70% of the assigned grade), ***Complete and*** ***Not*** ***working*** (lose 40% of assigned grade) and ***Not done*** get 0. The remaining grade is assigned to the quality of the implementation. In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation. Quality includes meaningful naming of identifiers, no redundant code, simple and efficient design, clean code without unnecessary files/code, use of comments where necessary, proper white space and indentation. **Marks will be reduced** forcode duplication, poor/inefficient coding practices, poor naming of identifiers and unnecessary complex/poor user interface design.

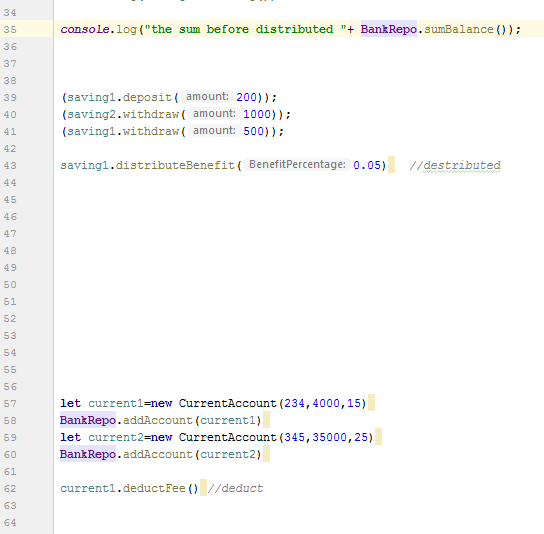
Banking App

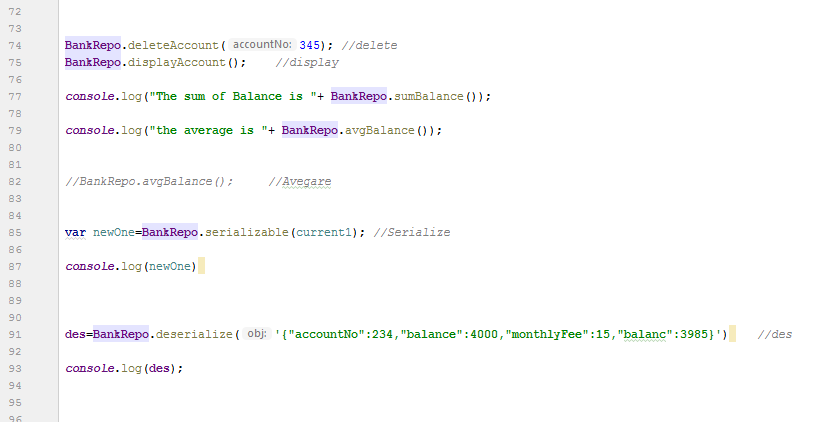




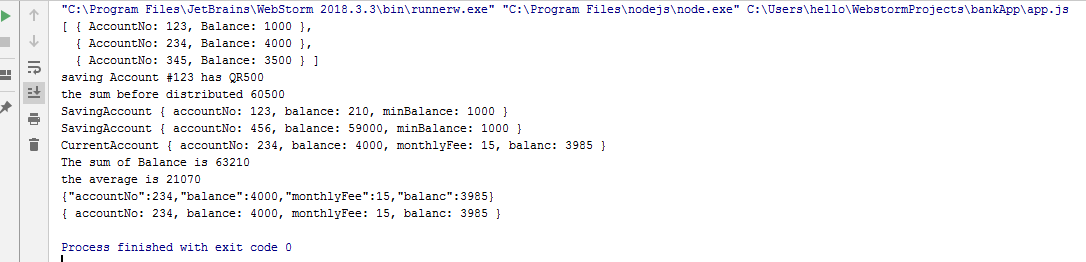
App.js







Run Result



Book Donation App

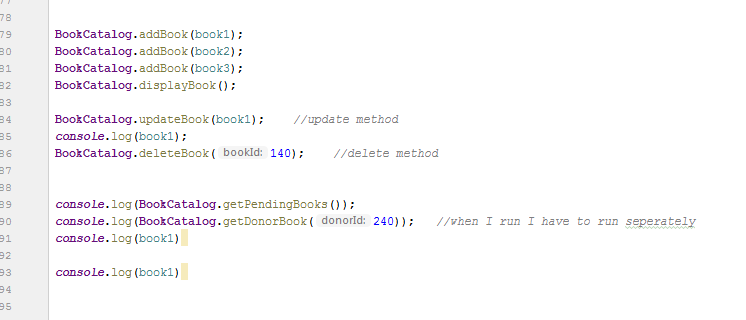




App.js







Run Result:

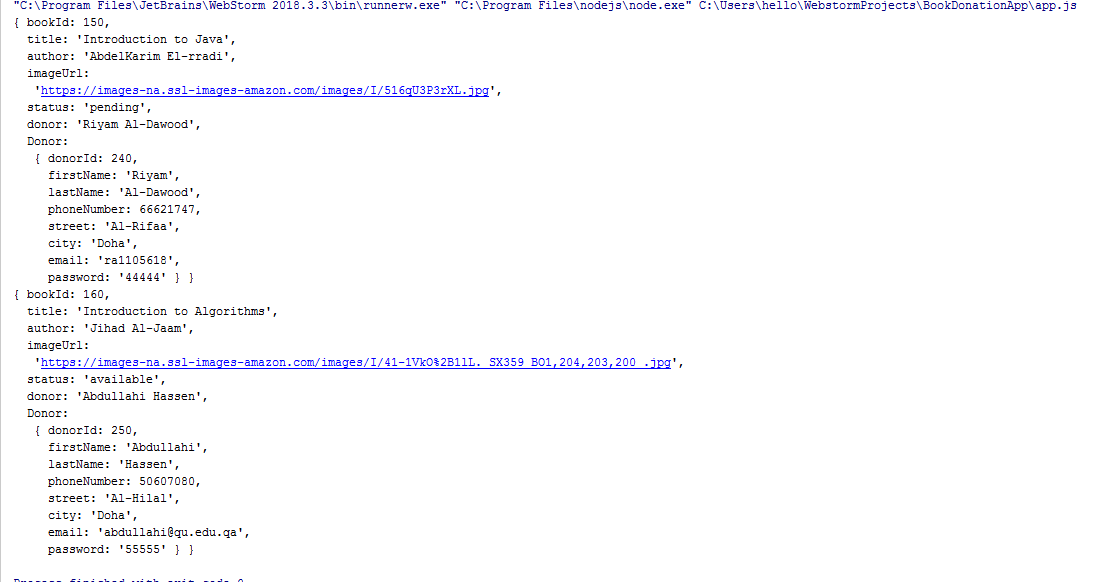
Add:



Update



Delete:



getPendingBook:



getDonorBook



## Testing evidence (You should provide a screenshot of your work)