Video Game Store APP

Made by Tomer Yaish

Introduction

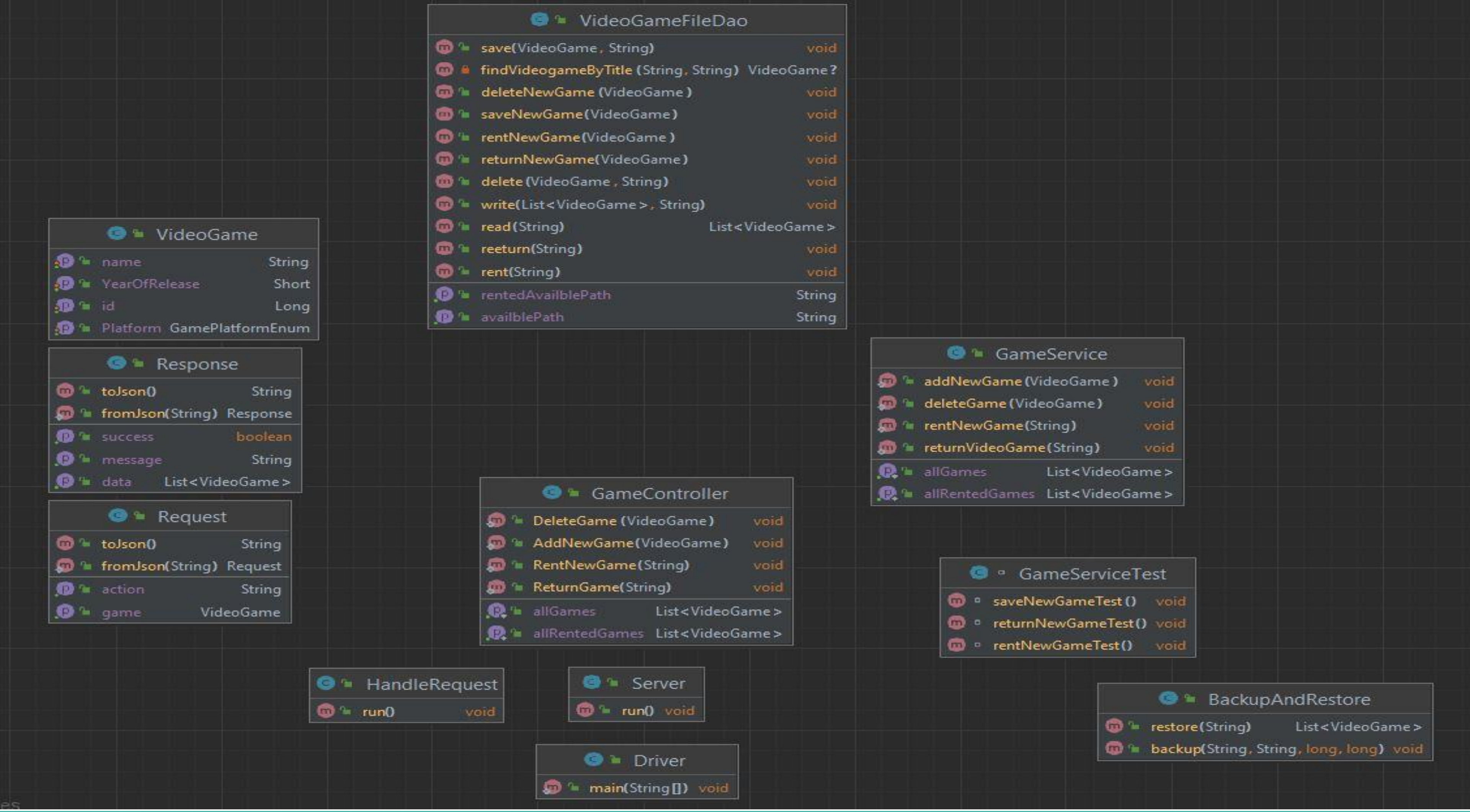
The system is a client-server-based software application that manages a inventory of a video game store.

The system includes a server-side component that uses a handler to process requests from the client, a controller to interact with the game service, and a Dao to store and retrieve data from a local text file. Also there is an option to backup and restore the data.

The client-side includes a GUI built using JavaFX and CSS, and follows the MVC (Model-View-Controller) architecture pattern to handle user inputs. The client-side also includes the option for users to log in as an employee using a password.

The system includes unit testing using JUnit.

The system also using pattern searching algorithms like KMP.



VideoGameFileDao	
save(VideoGame, String)	void
findVideogameByTitle (String, String)	VideoGame?
deleteNewGame (VideoGame)	void
saveNewGame(VideoGame)	void
rentNewGame(VideoGame)	void
returnNewGame(VideoGame)	void
delete (VideoGame, String)	void
write(List<VideoGame>, String)	void
read(String)	List<VideoGame>
reeturn(String)	void
rent(String)	void
rentedAvailblePath	String
availblePath	String

VideoGame	
name	String
YearOfRelease	Short
id	Long
Platform	GamePlatformEnum

Response	
toJson()	String
fromJson(String)	Response
success	boolean
message	String
data	List<VideoGame>

Request	
toJson()	String
fromJson(String)	Request
action	String
game	VideoGame

GameService	
addNewGame (VideoGame)	void
deleteGame (VideoGame)	void
rentNewGame(String)	void
returnVideoGame(String)	void
allGames	List<VideoGame>
allRentedGames	List<VideoGame>

GameController	
DeleteGame (VideoGame)	void
AddNewGame(VideoGame)	void
RentNewGame(String)	void
ReturnGame(String)	void
allGames	List<VideoGame>
allRentedGames	List<VideoGame>

GameServiceTest	
saveNewGameTest ()	void
returnNewGameTest()	void
rentNewGameTest()	void

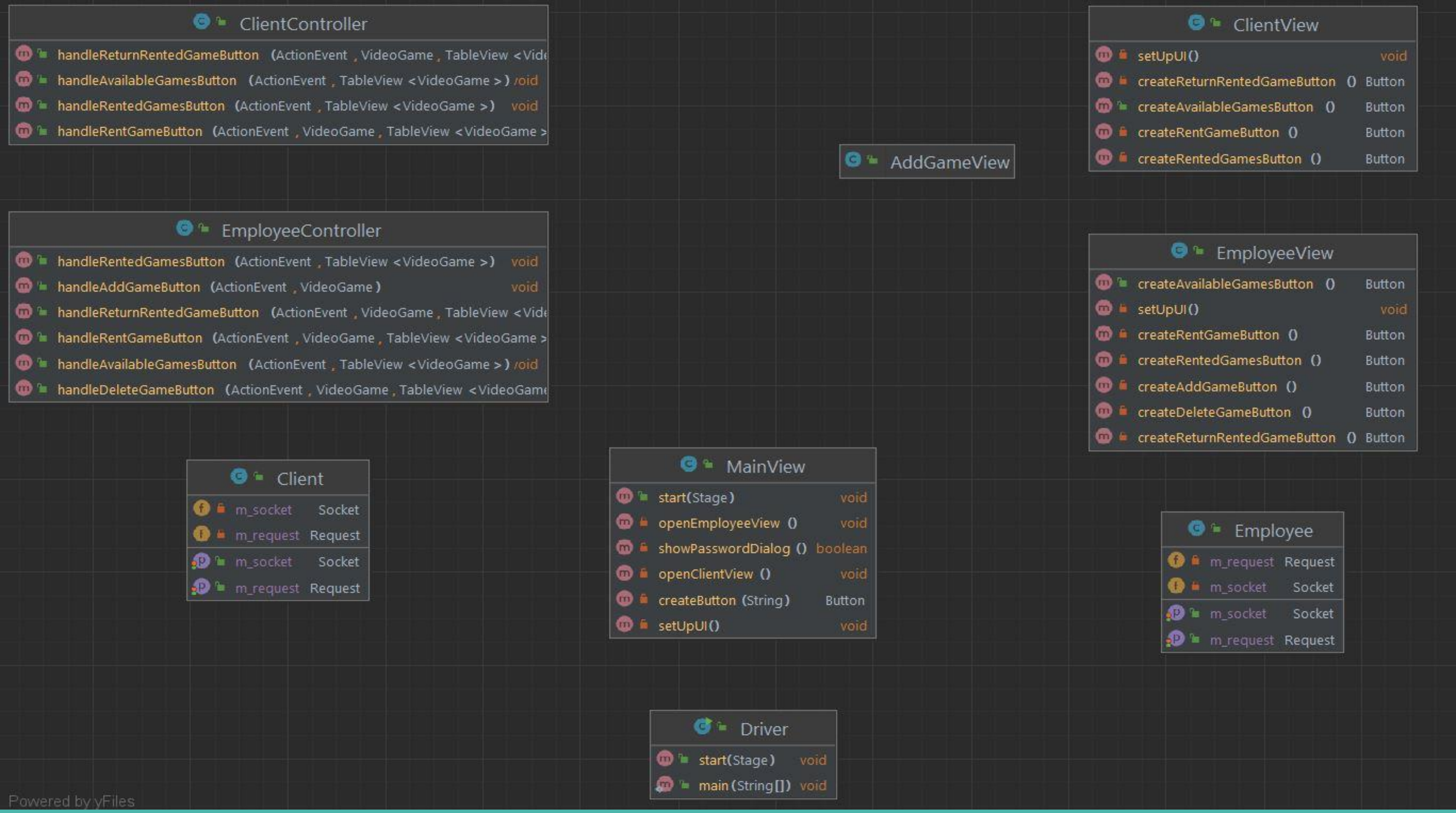
HandleRequest	
run()	void

Server	
run()	void

Driver	
main(String[])	void

BackupAndRestore	
restore(String)	List<VideoGame>
backup(String, String, long, long)	void

es


ClientController
handleReturnRentedGameButton (ActionEvent , VideoGame , TableView <Vid
handleAvailableGamesButton (ActionEvent , TableView <VideoGame >) /oid
handleRentedGamesButton (ActionEvent , TableView <VideoGame >) void
handleRentGameButton (ActionEvent , VideoGame , TableView <VideoGame >
EmployeeController
handleRentedGamesButton (ActionEvent , TableView <VideoGame >) void
handleAddGameButton (ActionEvent , VideoGame) void
handleReturnRentedGameButton (ActionEvent , VideoGame , TableView <Vid
handleRentGameButton (ActionEvent , VideoGame , TableView <VideoGame >
handleAvailableGamesButton (ActionEvent , TableView <VideoGame >) /oid
handleDeleteGameButton (ActionEvent , VideoGame , TableView <VideoGam
Client
m_socket Socket
m_request Request
m_socket Socket
m_request Request
AddGameView
MainView
start(Stage) void
openEmployeeView () void
showPasswordDialog () boolean
openClientView () void
createButton (String) Button
setUpUI() void
Driver
start(Stage) void
main(String[]) void
ClientView
setUpUI() void
createReturnRentedGameButton () Button
createAvailableGamesButton () Button
createRentGameButton () Button
createRentedGamesButton () Button
EmployeeView
createAvailableGamesButton () Button
setUpUI() void
createRentGameButton () Button
createRentedGamesButton () Button
createAddGameButton () Button
createDeleteGameButton () Button
createReturnRentedGameButton () Button
Employee
m_request Request
m_socket Socket
m_socket Socket
m_request Request
Powered by yFiles