February 15, 2019

ECSE-223-001

Model-Based Programming

Group P22

Members:

Guevorkian, Andrei - 260737896

Kim, Younggue-260531176

Kinman, Garrett

Mircic, Michael - 260587925

Nuviadenu, Edem Koshi -260779440

Smith, Sean - 260787775

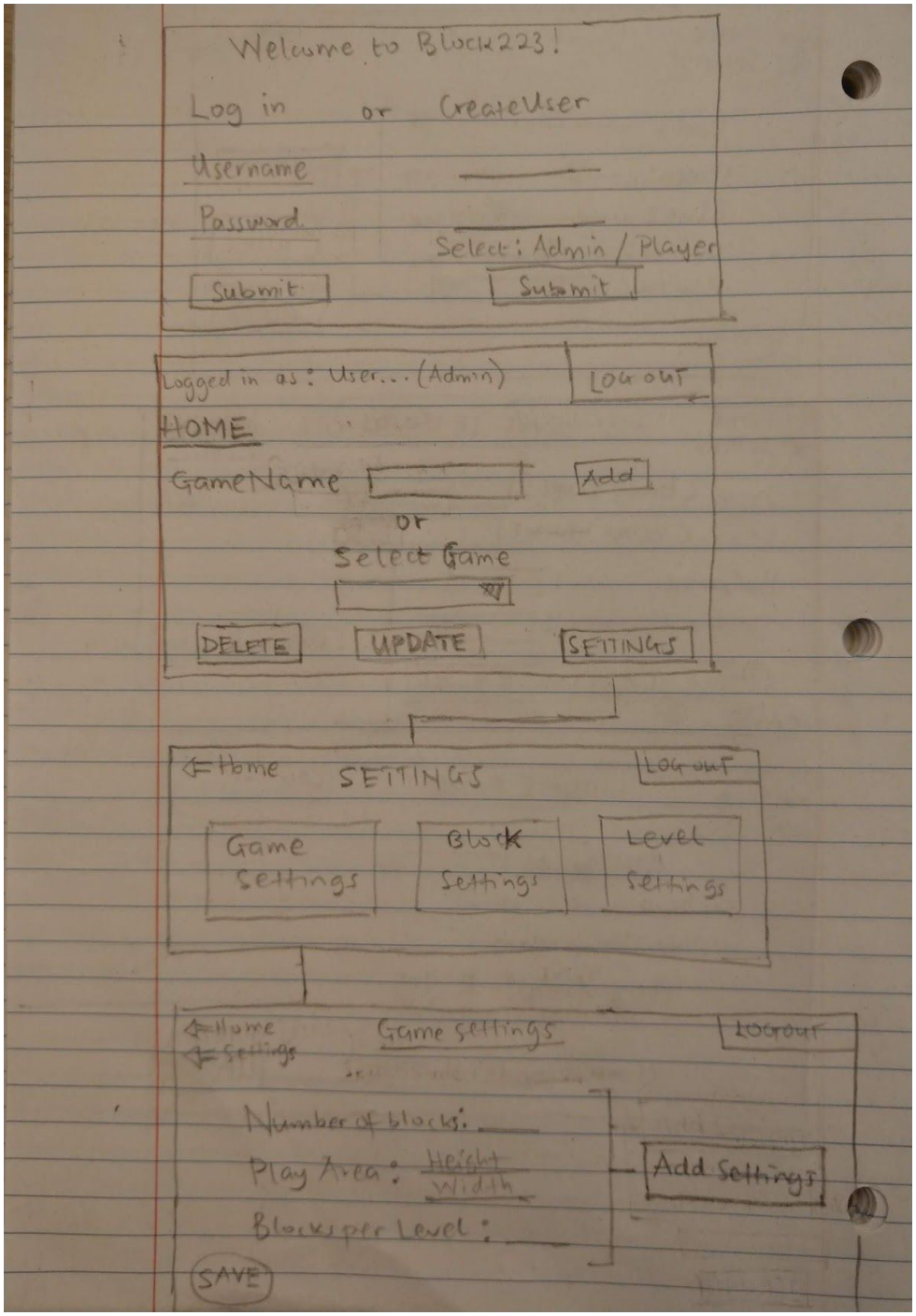
Project Iteration 2

Report

Table of Contents

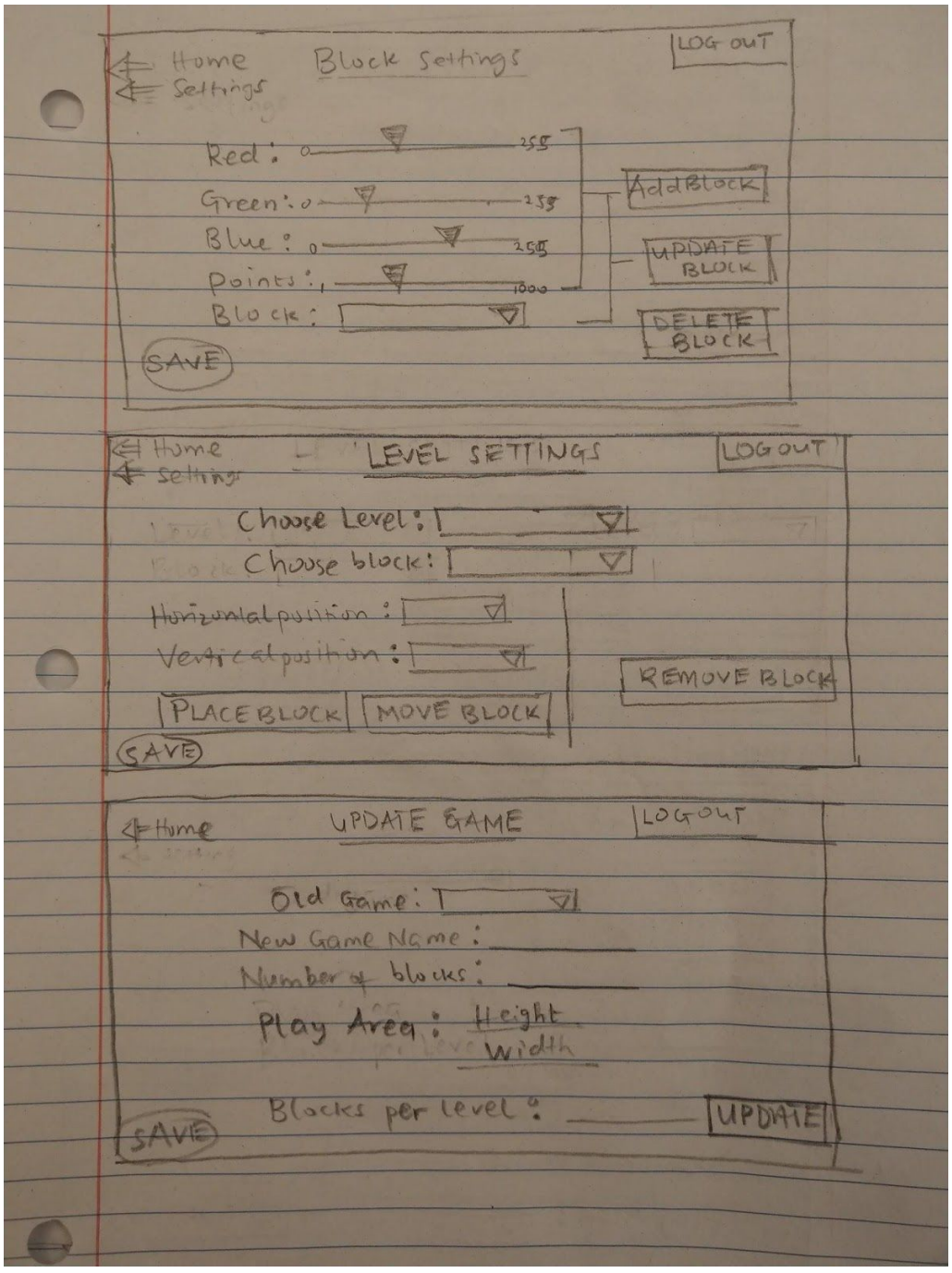
UI Mockup - Team

Figure 1: Mockup Part 1



UI Mockup- Team (Continued)

Figure 2: Mockup Part 2



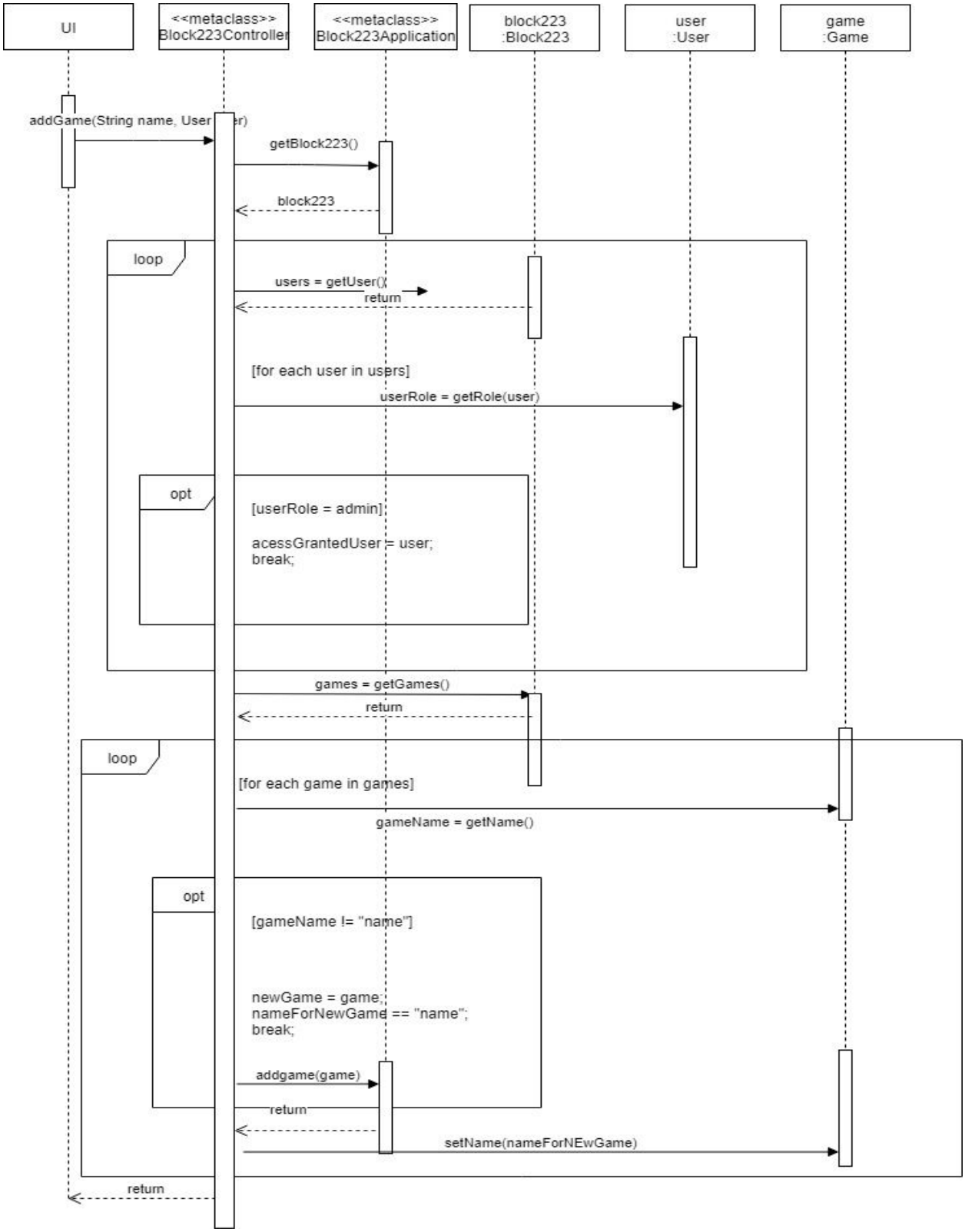
Add a game - Kim, Younggue

Specification of Controller Interface

```
public boolean addGame(String name, User user)
public block223 getBlock223()
public List<User> getUsers()
public UserRole getRole(int index)
Public List<Game> getGames()
Public String getName()
Public boolean addGame(Game aGame)
Public boolean setName(String aName)
```

Adding a Game - Kim, Younggue (Continued)

Figure 3: Sequence Diagram for Adding a Game

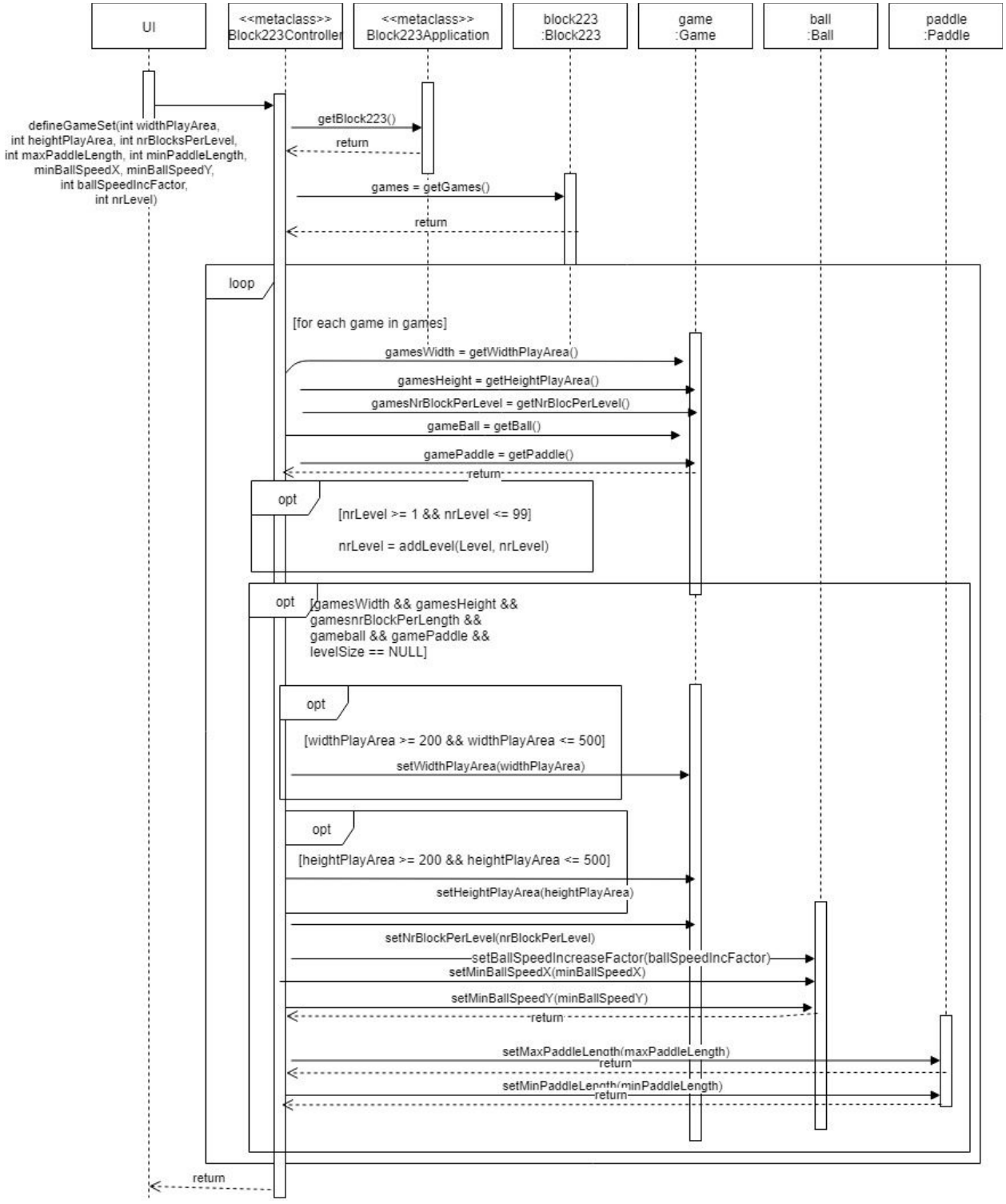


Define game settings - Kim, Younggue
Specification of Controller Interface

public boolean defineGameSetting(int widthPlayArea, int heightPlayArea, int nrBlocksPerLevel, int maxPaddleLength, int minPaddleLength, int minBallSpeedX, int minBallSpeedY, int ballSpeedIncFactor, int nrLevel)
public block223 getBlock223()
public List<Game> getGames()
public int getHeightPlayArea()
public int getWidthPlayArea()
public paddle getPaddle()
public Ball getBall()
public int getNrBlocksPerLevel()
public boolean addLevelAt(Level aLevel)
public boolean setHeightPlayArea(int aHeightPlayArea)
public boolean setNrBlockPerLevel(int aNrBlocksPerLevel)
public boolean setWidthPlayuArea(int aWidthPlayArea)
public boolean setMinBallSpeedX(int aMinBallSpeedX)
public boolean setMinBallSpeedY(int aMinBallSpeedY)
public boolean setBallSpeedIncreaseFactor(double aBallSpeedIncreasedFactor)
public boolean setMaxPaddleLength(int aMaxPaddleLength)
public boolean setMinPaddleLength(int aMinPaddleLength)

Define game settings - Kim, Younggue (Continued)

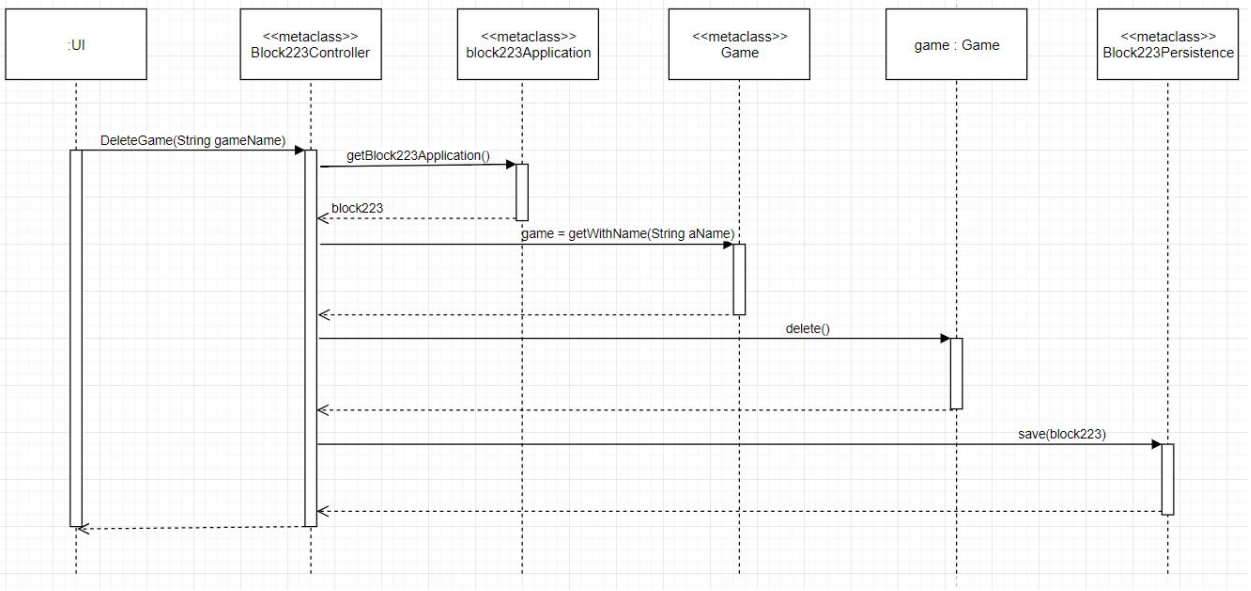
Figure 4: Sequence diagram for Defining Game Settings



Sean Smith

FEATURE	SPECIFICATION OF CONTROLLER INTERFACE
Allows game to be deleted by its' admin	Public void DeleteGame(String gameName) throws invalidInputException

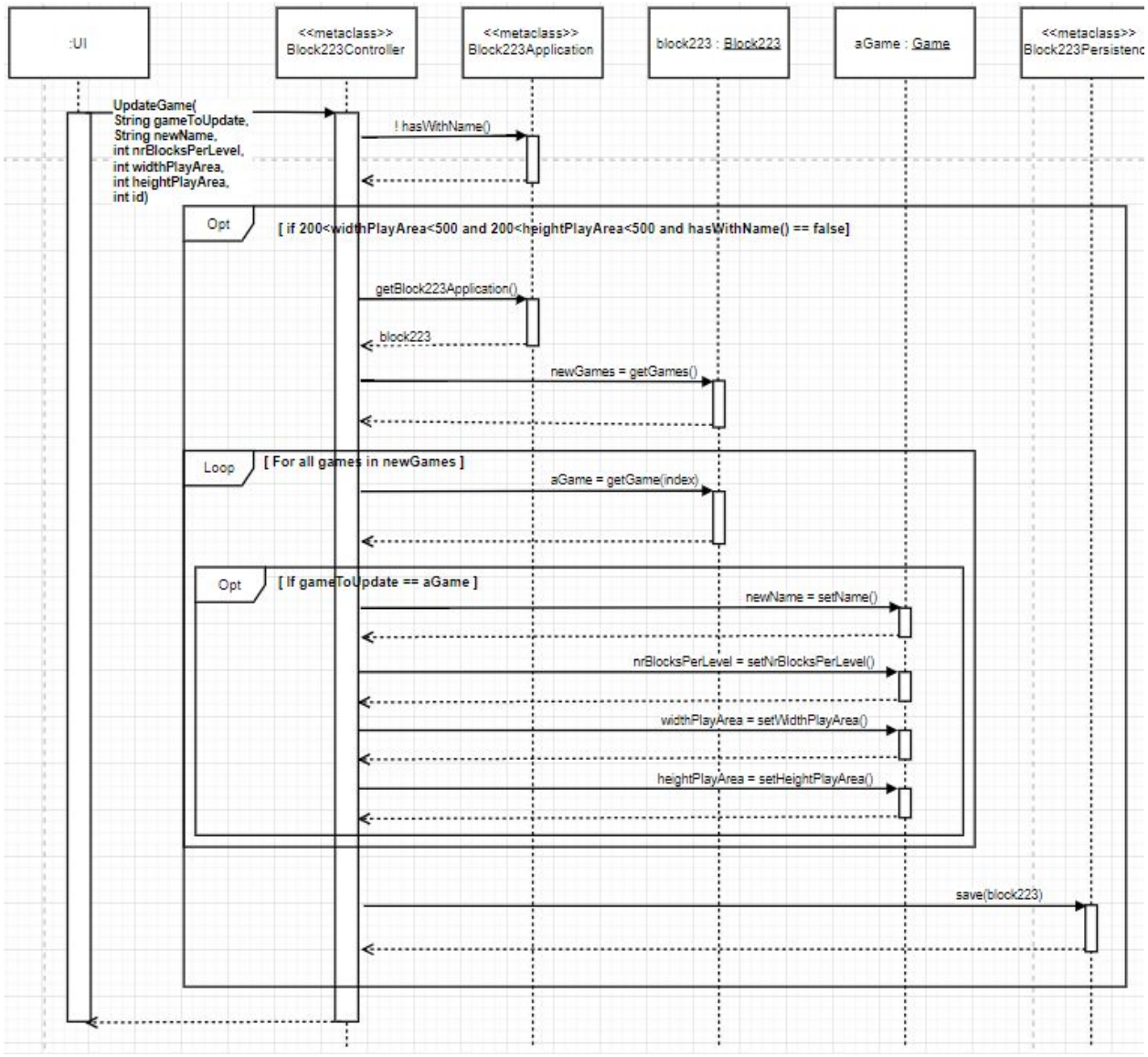
Figure 5: Sequence Diagram for Deleting a Game



Sean Smith

FEATURE	SPECIFICATION OF CONTROLLER INTERFACE
Allows game attributes to be updated by its' admin	Public void UpdateGame(String gameToUpdate, String newName, int nrBlocksPerLevel, int widthPlayArea, int heightPlayArea, int id) throws invalidInputException

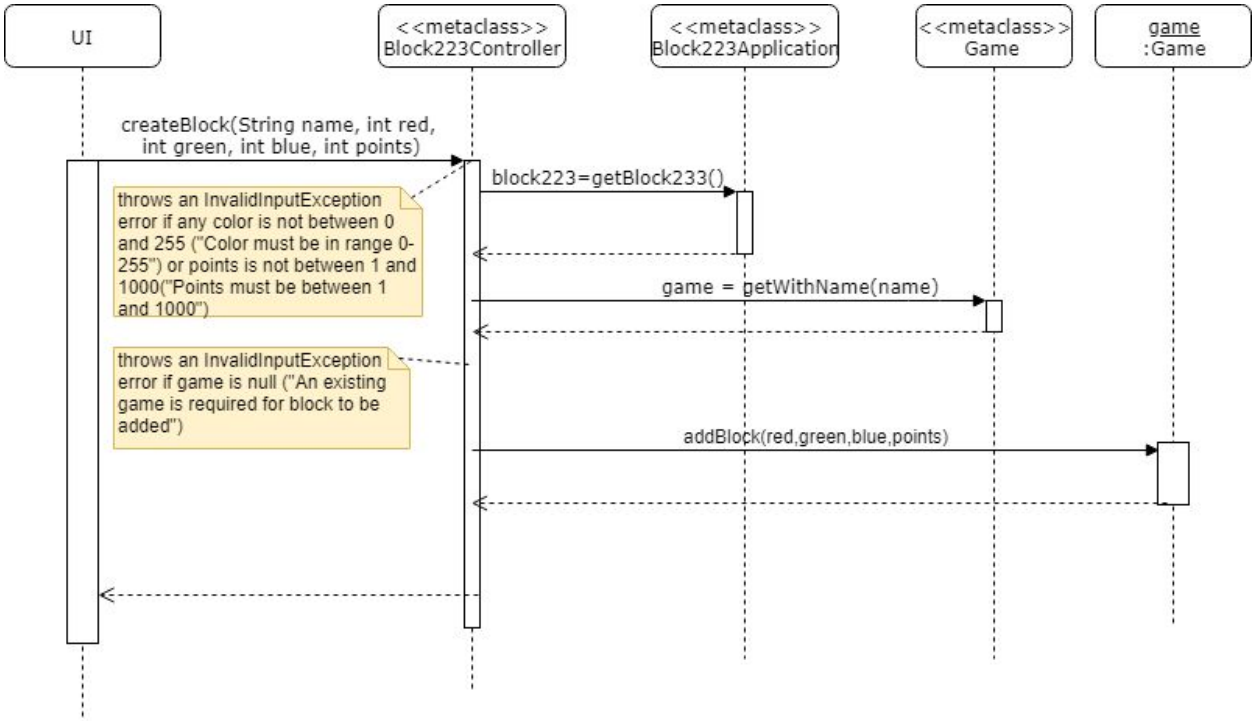
Figure 6: Sequence Diagram for Updating a Game



Nuviadenu, Edem Koshi

Feature	Block223Controller Interface
Add a block to game	public static void createBlock(String name, int red, int green, int blue, int points) throws InvalidInputException

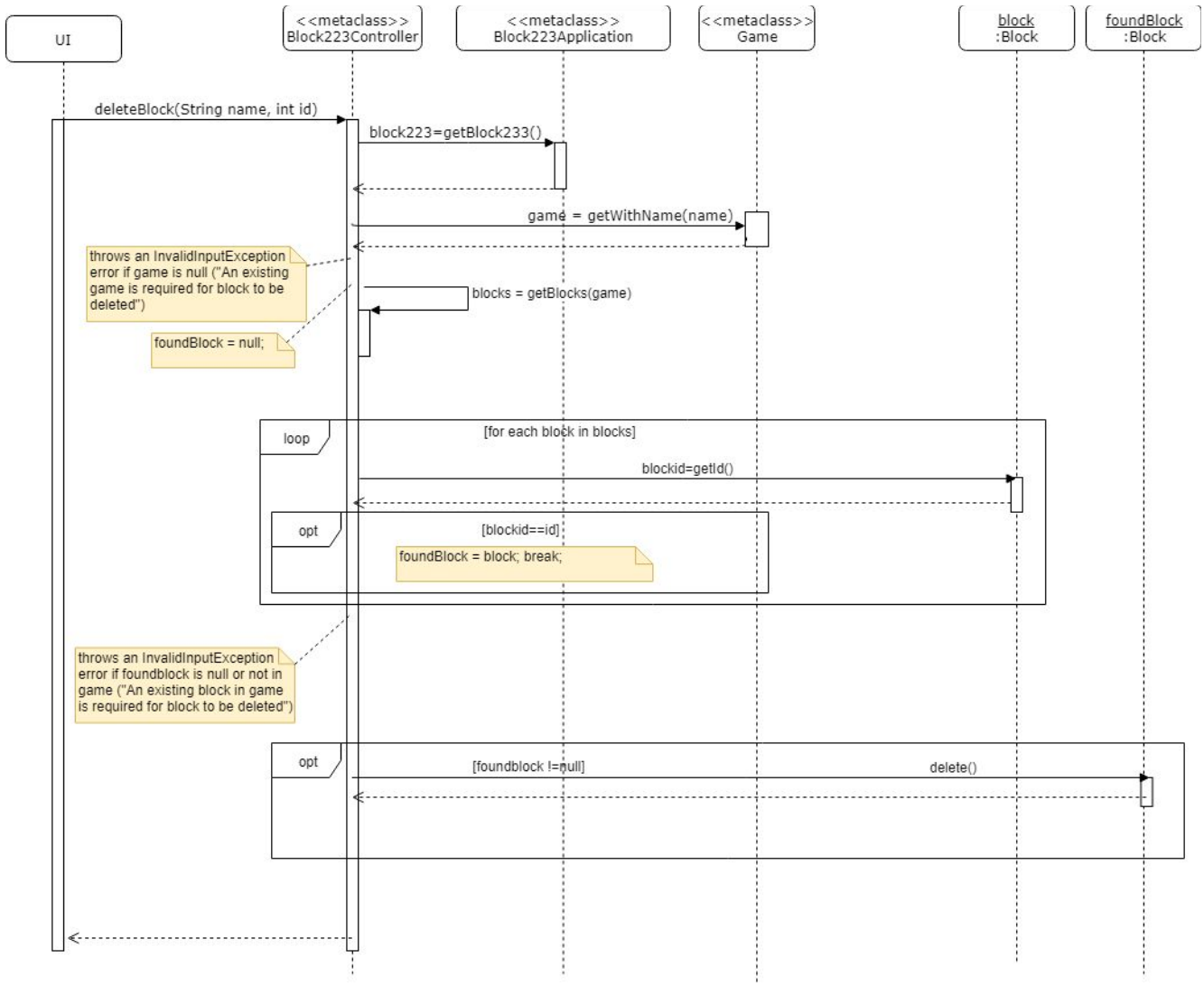
Sequence diagram for adding a block to a game



Nuviadenu, Edem Koshi

Feature	Block223Controller Interface
Delete a block from game	public static void deleteBlock(String name, int id) throws InvalidInputException Public static List<Blocks>findBlock(game)

Sequence diagram for deleting a block from game

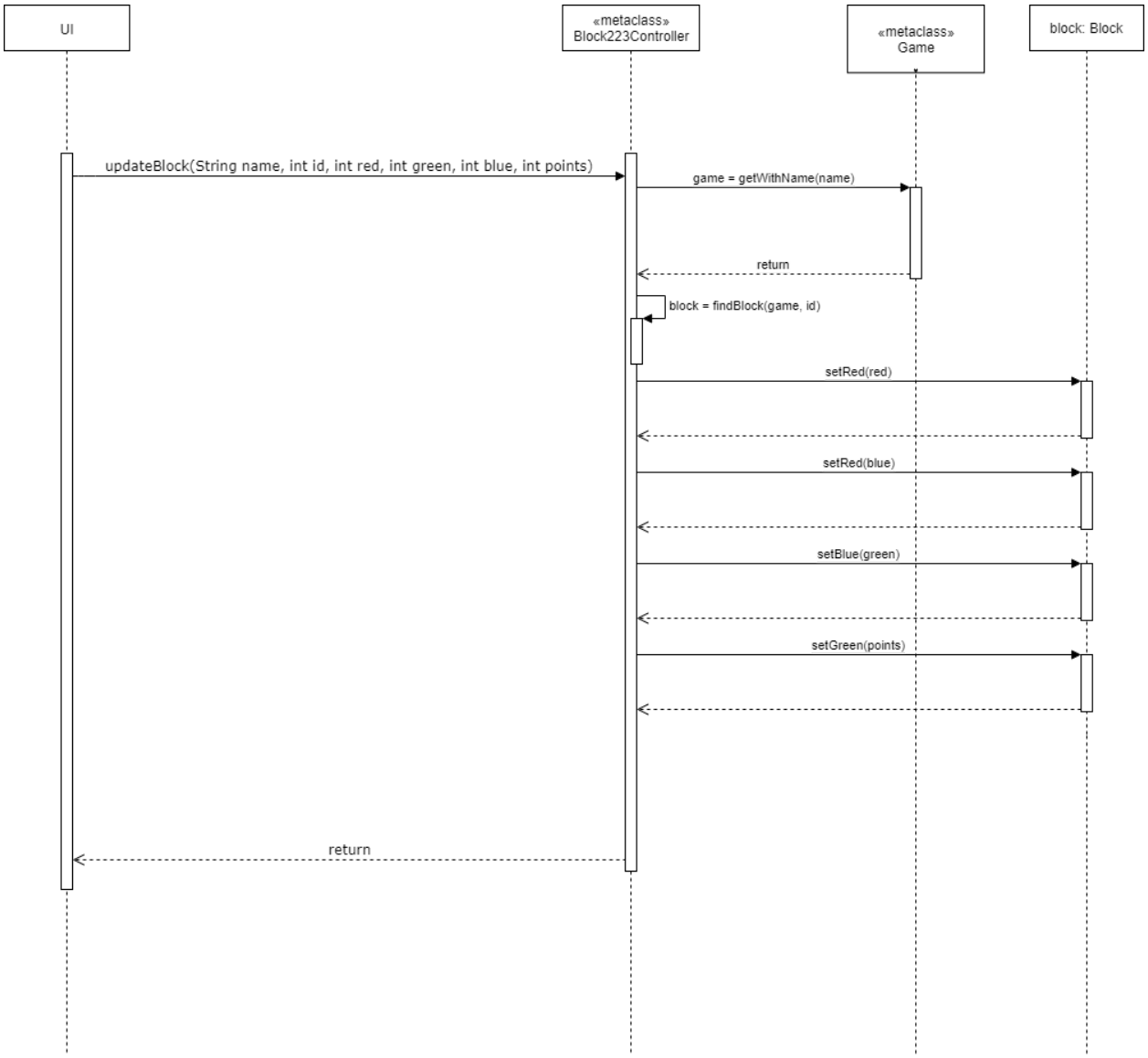


Kinman, Garrett
Query method

1. findBlock() /getBlock()

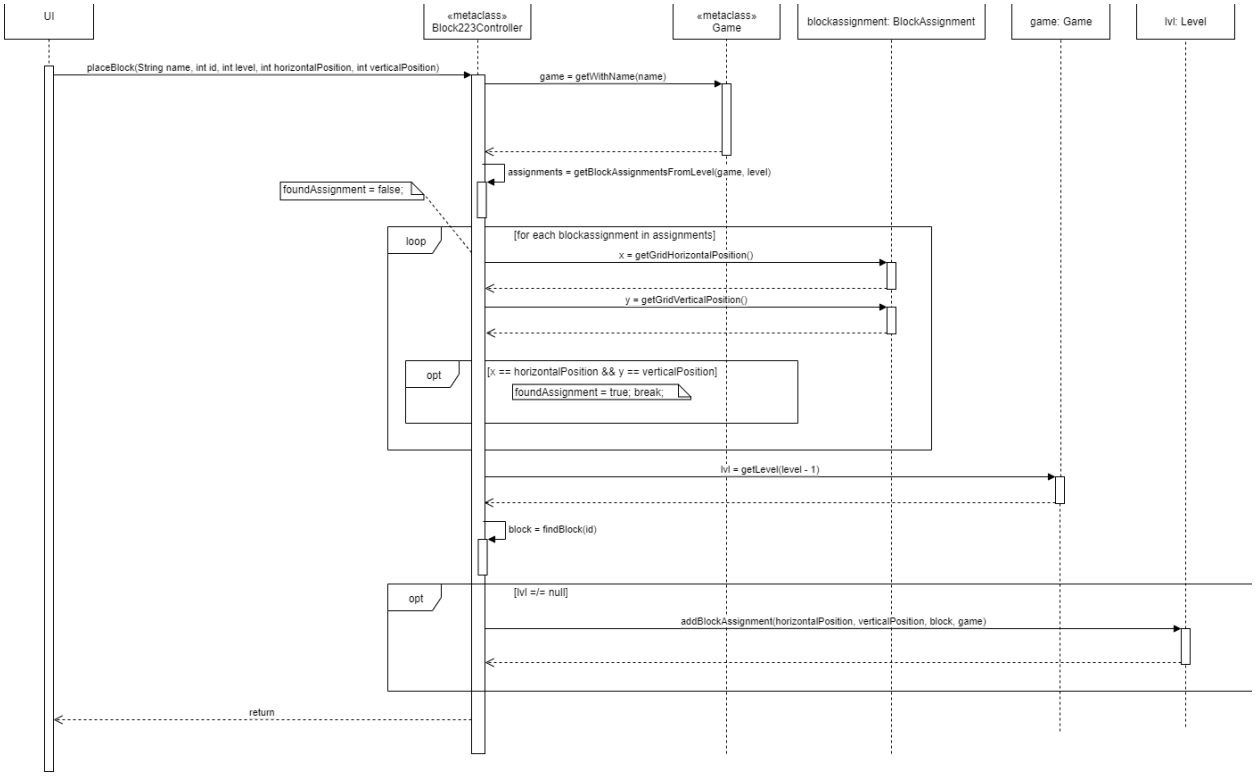
Update a block in a game

public static Block findBlock(Game game, int id);
public static void updateBlock(String name, int id, int red, int green, int blue, int points);



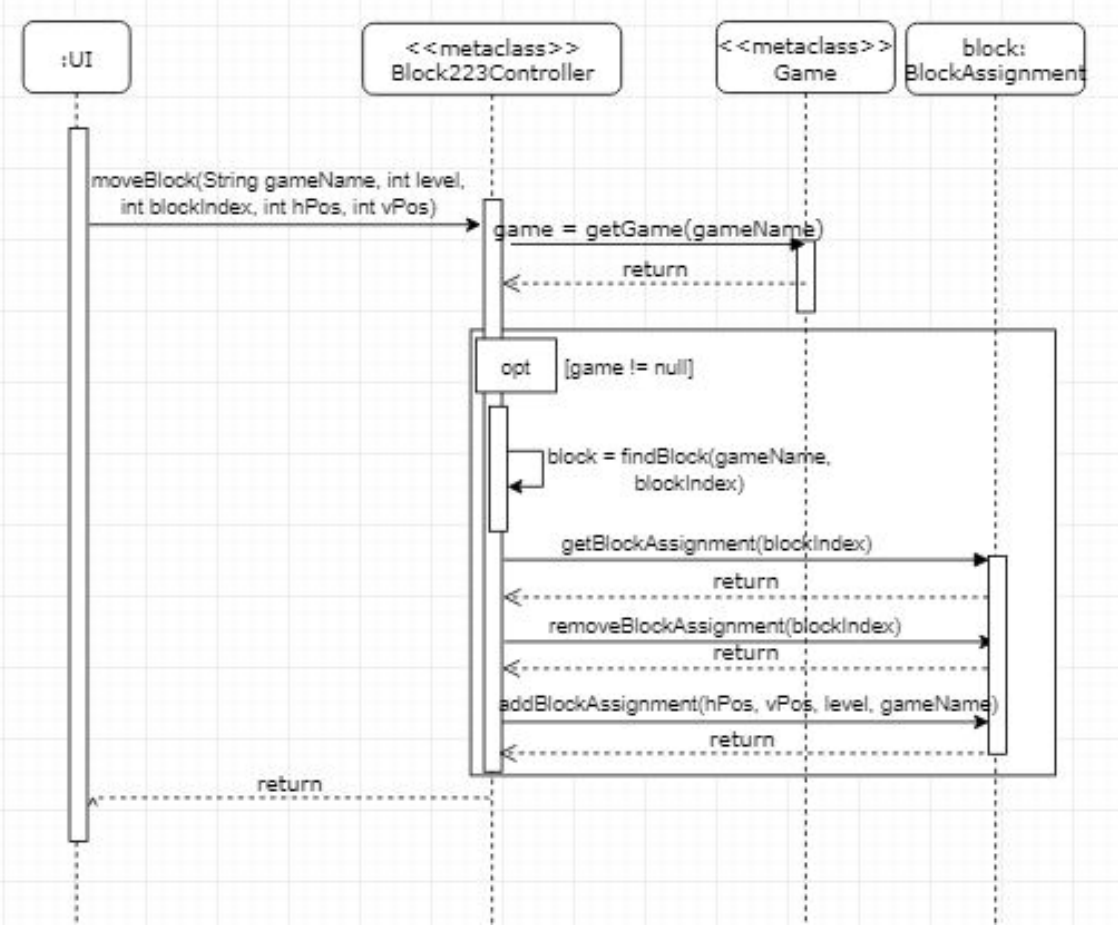
Kinman, Garrett
Position a block at a grid location in level

public static void placeBlock(name, int id, int level, int horizontalPosition, int verticalPosition);



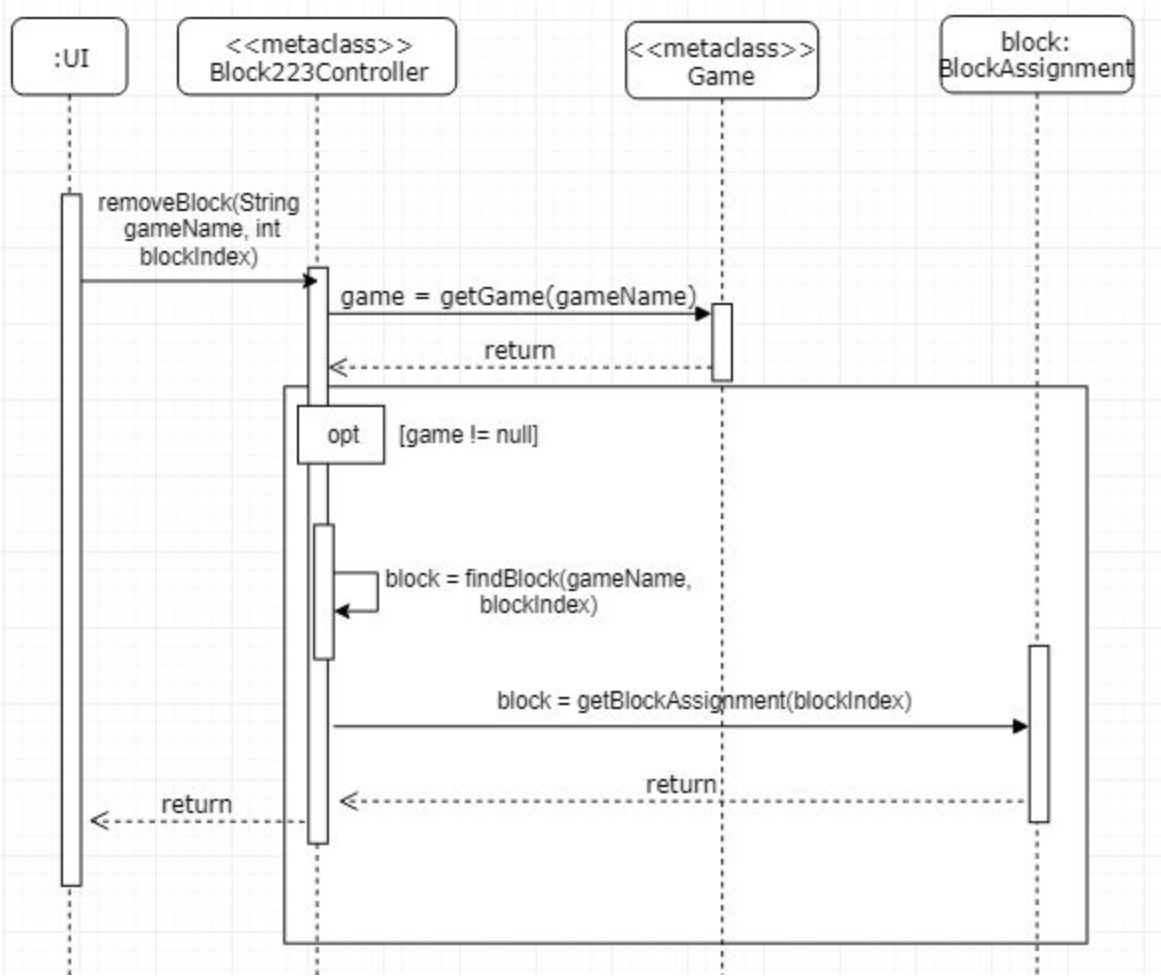
10. Move a block from one grid position to another in a level- Guevorkian, Andrei

Controller Interface: public static void moveBlock(String gameName, int level, int blockIndex, int hPos, int vPos)



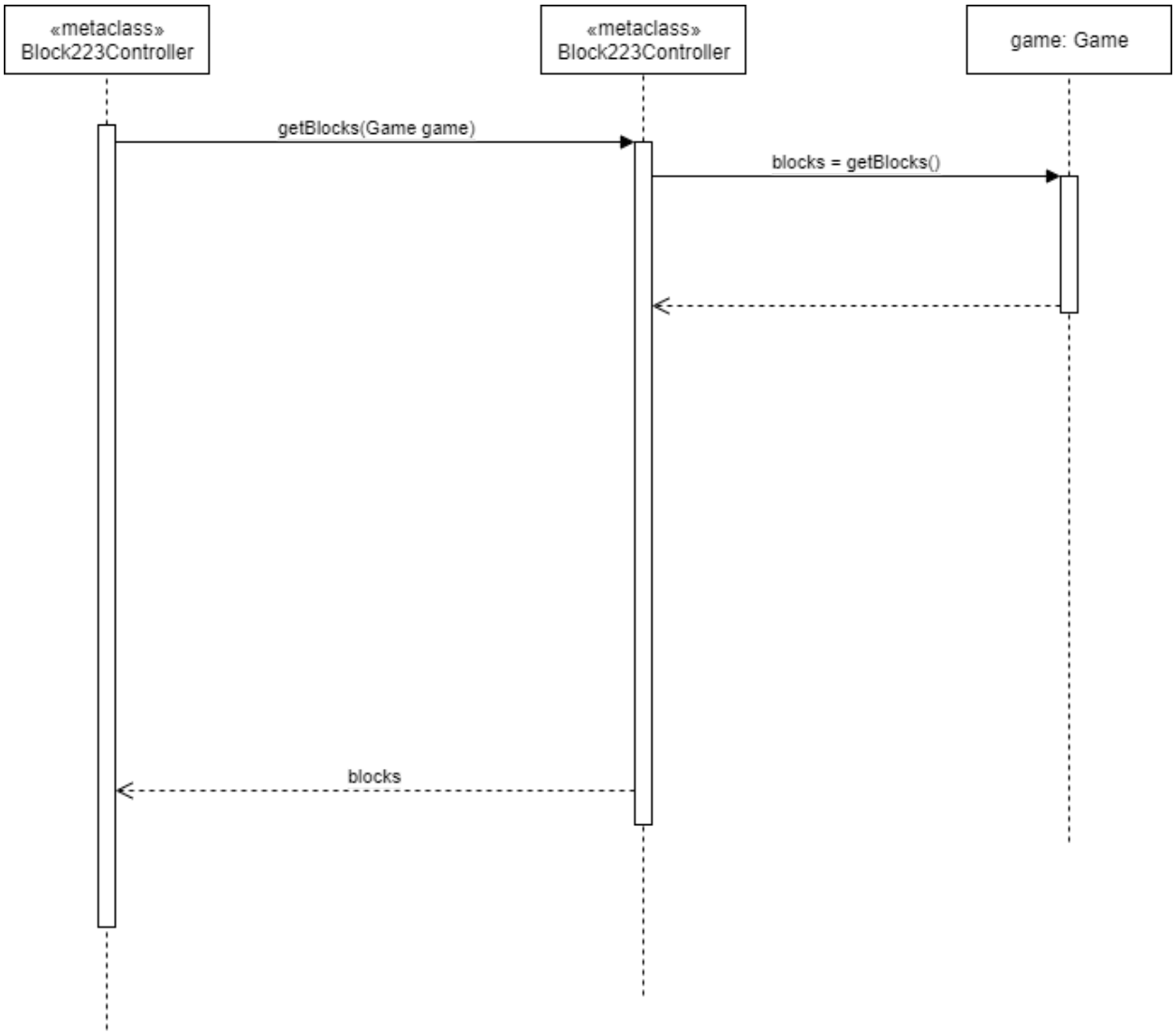
11. Remove a block from level- Guevorkian, Andrei

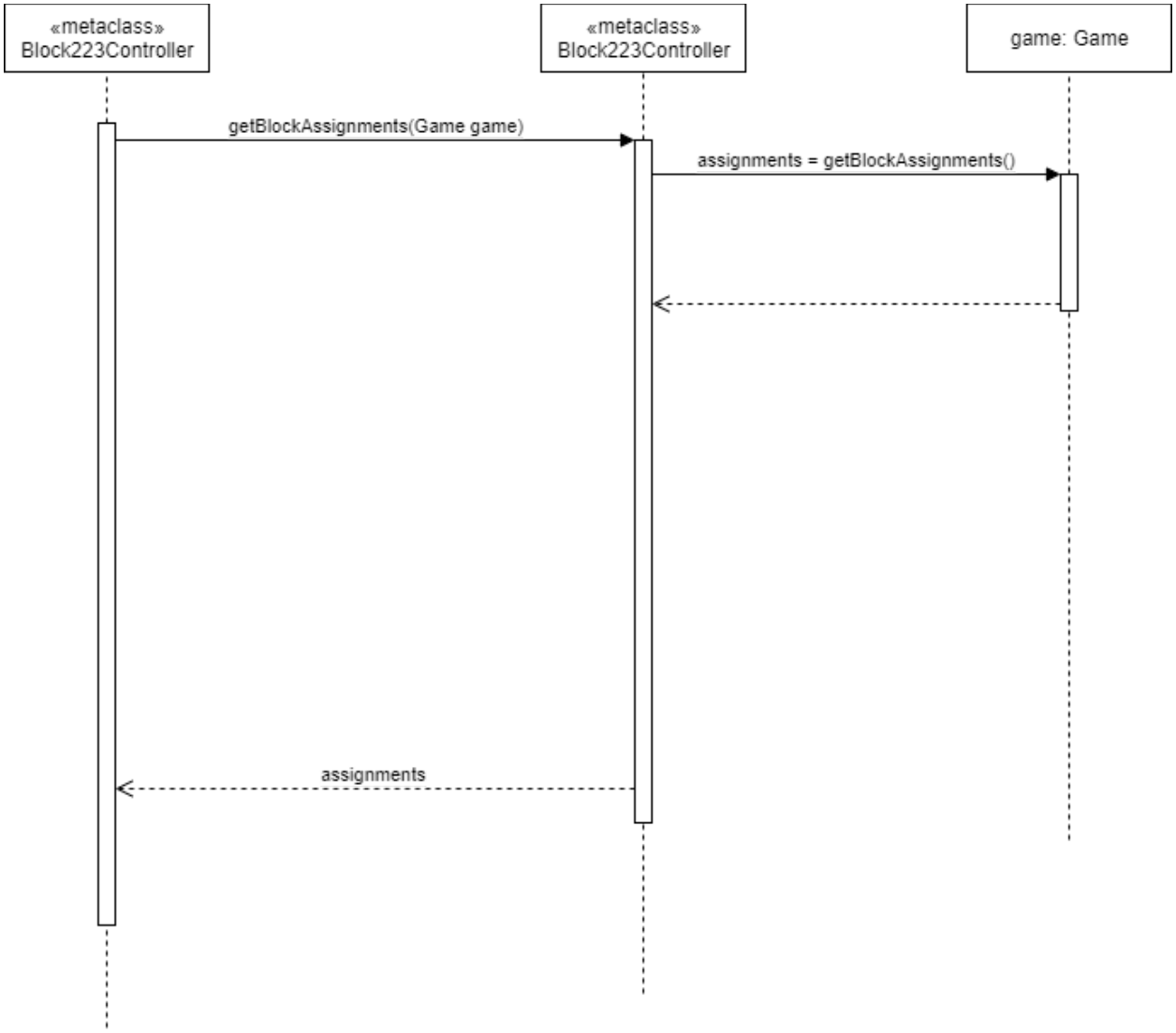
Controller Interface: public static void removeBlock(String gameName, int blockIndex) throws InvalidInputException

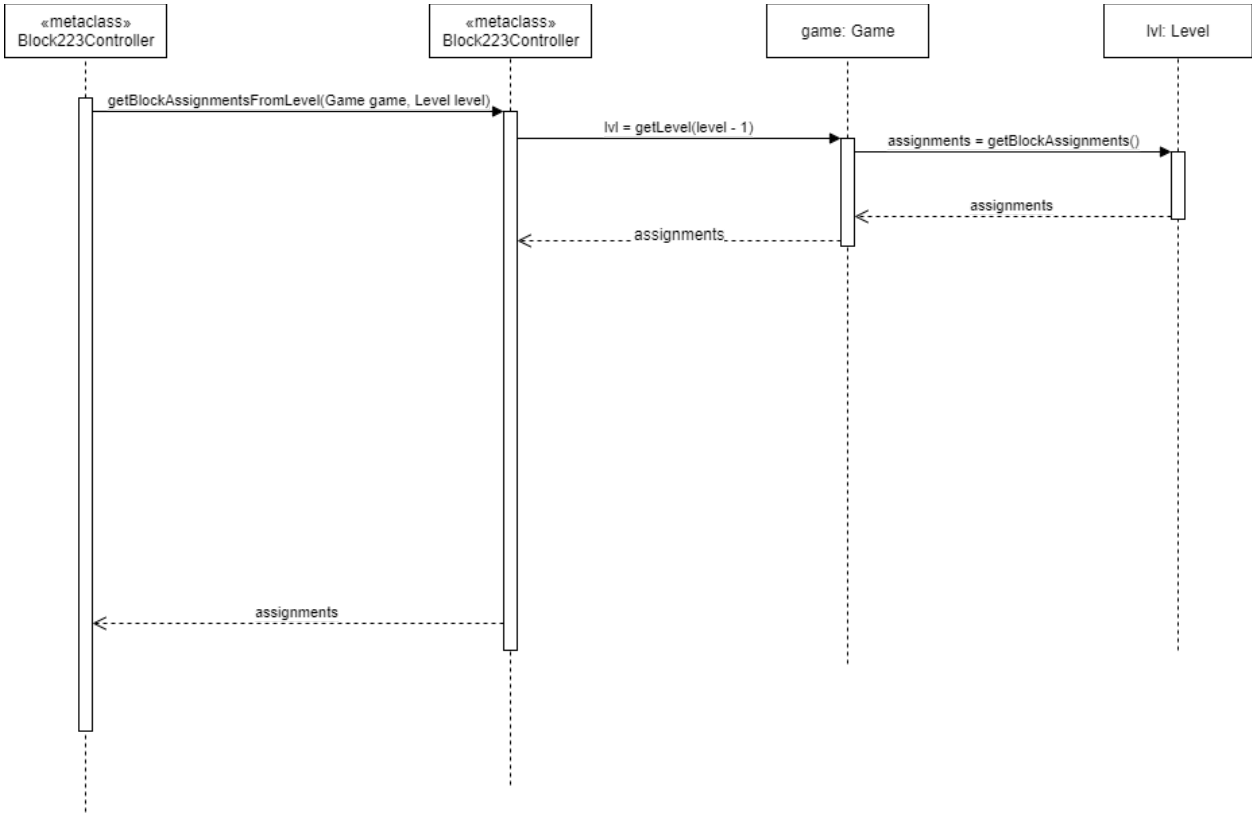


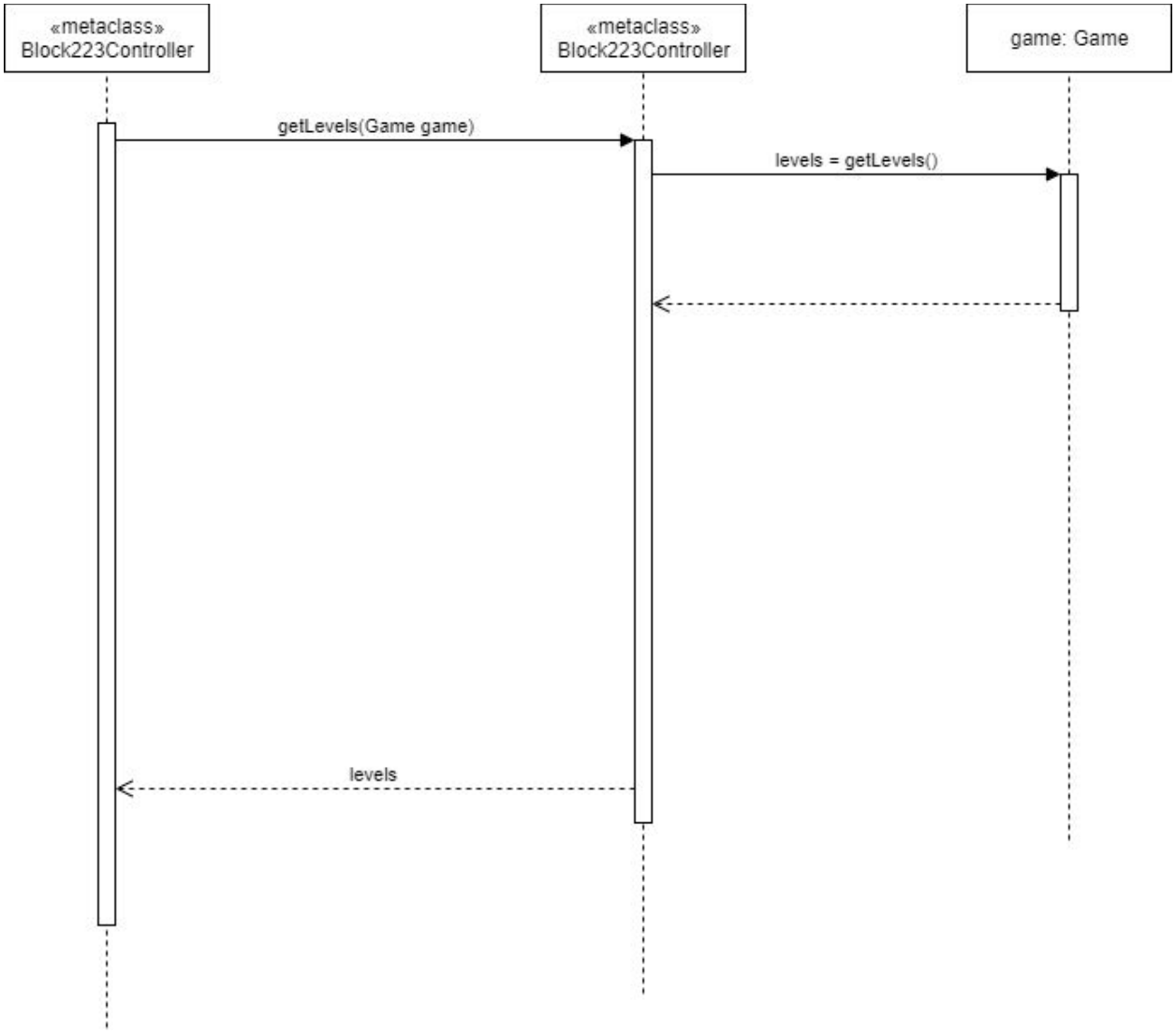
Query Methods

public static Block findBlock(Game game, int id);
public List<Block> getBlocks(Game game);
public List<BlockAssignment> getBlockAssignments(Game game);
public List<Level> getLevels(Game game);
public List<BlockAssignment> getBlockAssignmentsFromLevel(Game game, Level level);




«metaclass»
Block223Controller
«metaclass»
Block223Controller
game: Game
getBlockAssignments(Game game)
assignments = getBlockAssignments()
assignments


«metaclass»
Block223Controller
«metaclass»
Block223Controller
game: Game
lvl: Level
getBlockAssignmentsFromLevel(Game game, Level level)
lvl = getLevel(level - 1)
assignments = getBlockAssignments()
assignments
assignments
assignments


«metaclass»
Block223Controller
«metaclass»
Block223Controller
game: Game
getLevels(Game game)
levels = getLevels()
levels

12-Save game (Michael Mircic)

Method that will save the current game along with all given parameters as a serialized object.

public boolean saveGame ()

The method will return true on a successful save and false otherwise.

Queries:

From Block223

public User getCurrentUser()

This method we will need to create will return the currently logged in user. The assumption is made that the user will be an admin, as this operation will only be allowed from the admin UI.

From Admin

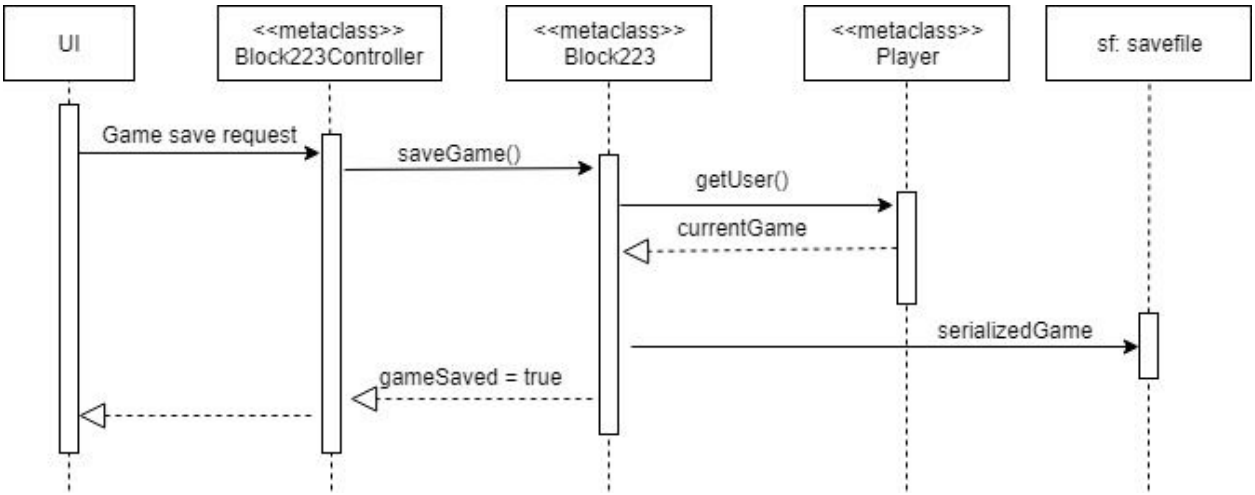
public int indexOfGame(Game aGame)

public Game getGame(int index)

These methods will both find the current game and get the associated information from the database.

Serialization of the game will then occur, with the output stream redirected to a given file.

Sequence Diagram for Saving a Game



13-Log in/out as a player or admin (Michael Mircic)

Method that will sign up a new user:

public boolean signUp (String username, String playerPW, String adminPW)

This method has 2 password variables, with the option to become an admin presented immediately. If an empty string is given for admin password, then it will be assumed that user only wants to create a Player. It can also be used to create a new admin from an existing Player user by leaving the playerPW field with an empty string. If the given username does not exist and no playerPW value is given, false will be returned, and no user will be created.

Queries:

From Block223

public User addUser(String aUsername, UserRole… allRoles)

public boolean addUser(User aUser)

The second addUser method will return false if the user already exists. The new user created in the first step will be deleted.

From User

public boolean addRole(UserRole aRole)

This will be used to add an admin role if that is requested later on.

Next, the method to log in:

public int logIn (String username, String password)

This method will return 0 if given username does not exist, 1 if user logged in as Player and 2 if user logged in as Admin. It will also return -1 if there is already a user logged in (though this should not be allowed to happen through the UI).

Queries:

From Block223

public int indexOfUser(User aUser)

public User getUser(int index)

The first method will both check if the user exists (returns -1 otherwise) and the second will pull out information on the given user.

From User

public int numberOfRoles()

public UserRole getRole(int index)

The first method will check if the user has an admin role (value returned would be 2 as opposed to 1). The second will be used to get one or both roles to check their assigned passwords with the password given.

From UserRole

public String getPassword()

Finally, the method used to log out:

public boolean logOut ()

The method will ask the user if they are sure they want to log out. On reconfirmation, log out will occur and true will be returned. Otherwise, false will be returned. This will also return false if a logout attempt is made while there is no currently logged in users, though this case will be impossible to achieve through the UI. If the logout operation completes successfully, the currently logged in user will also be set to null.

Queries:

From Block223

public User getCurrentUser()

This method we will need to create will return the user that is currently logged in.

Sequence Diagram for Logging In and Out