ECSE 223
Model-Based Programming

Winter 2019
Iteration 4
Project Report

Group P 6
Zheng Yu Cui (260808525)
Xu Hai (260661832)
Murray Kornelsen (260714814)
Helene Ma Yang (260775804)
Haowei Qiu (260762269)
Weijing Zhang (260784623)

March 15, 2019
McGill University

Features

1. **Start/pause/resume game** (Murray Kornelsen)

   - Controller Interface (all members)

     Modifier methods
     public static void startGame(String gameName) throws InvalidInputException;
     public static void openSavedGame(int gameId) throws InvalidInputException;
     public static void pauseGame();
     public static void resumeGame() throws InvalidInputException;
     private static void startGameLoop();

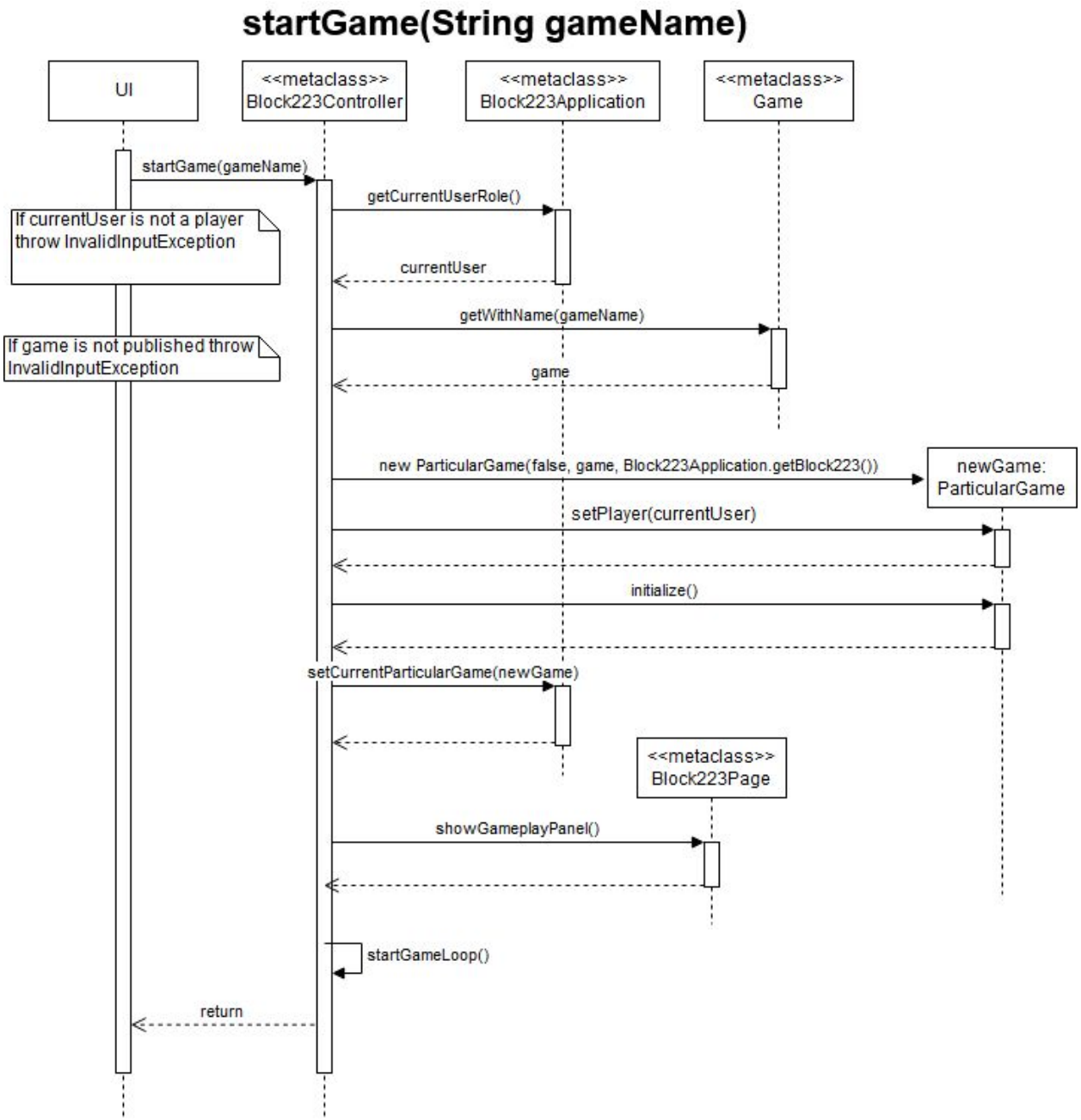
     *Query methods *see sequence diagram(s) at the end of the document**
     public static List<TOGame> getPlayableGames();
     public static List<TOParticularGame> getSavedGames();
     public static TOParticularGame getCurrentPlayingGame();
     public static TOParticularLevel getCurrentPlayingLevel();
     public static List<TOGridCell> getBlocksOfCurrentLevel();

   - View Interface

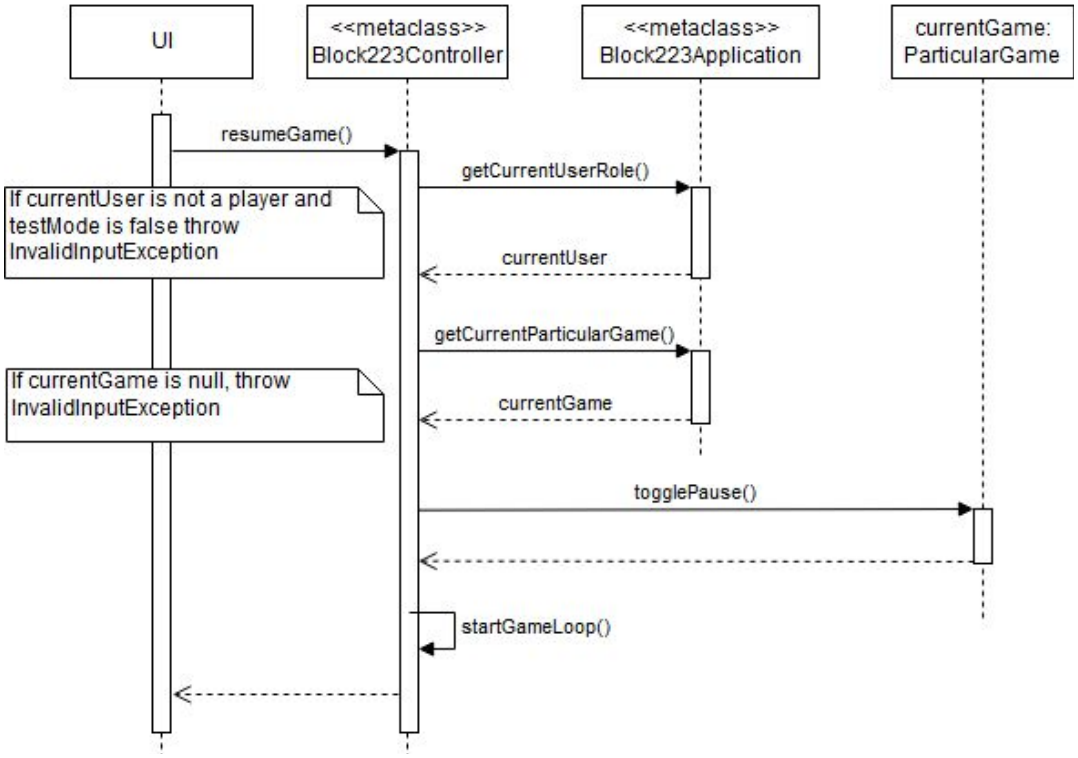
     public static void showGameplayPanel();
     public static int getUserInput();
     public static void redrawGamePanel();

- Sequence Diagrams

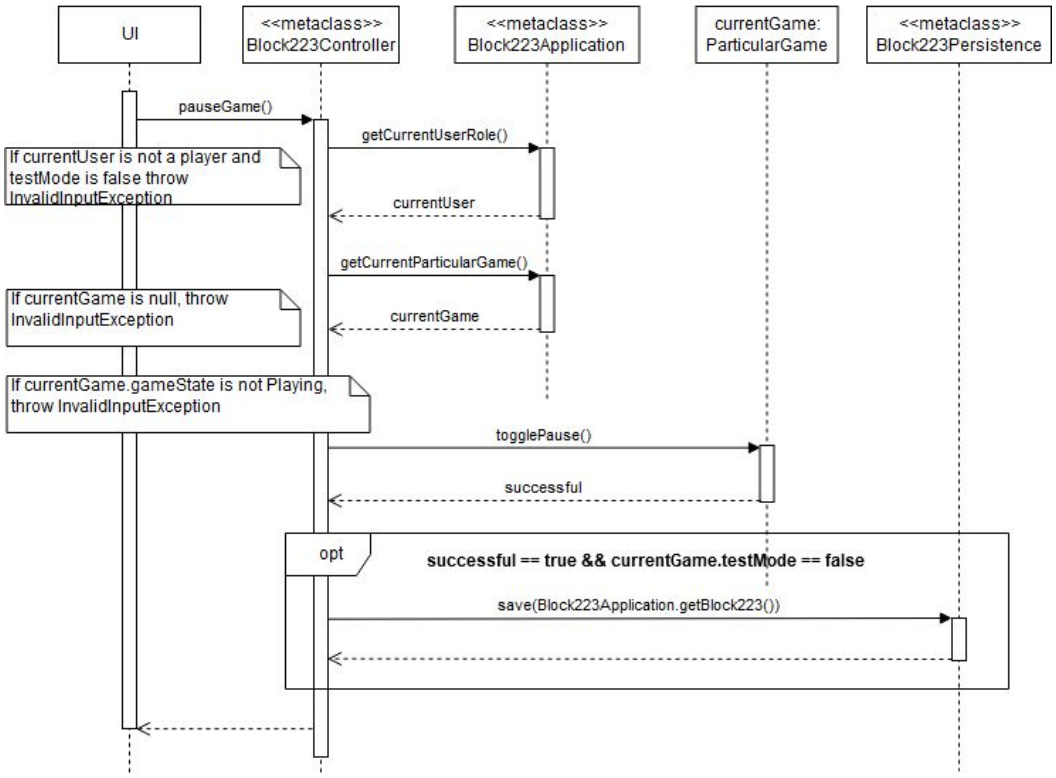
  Controller:
  Modifier methods



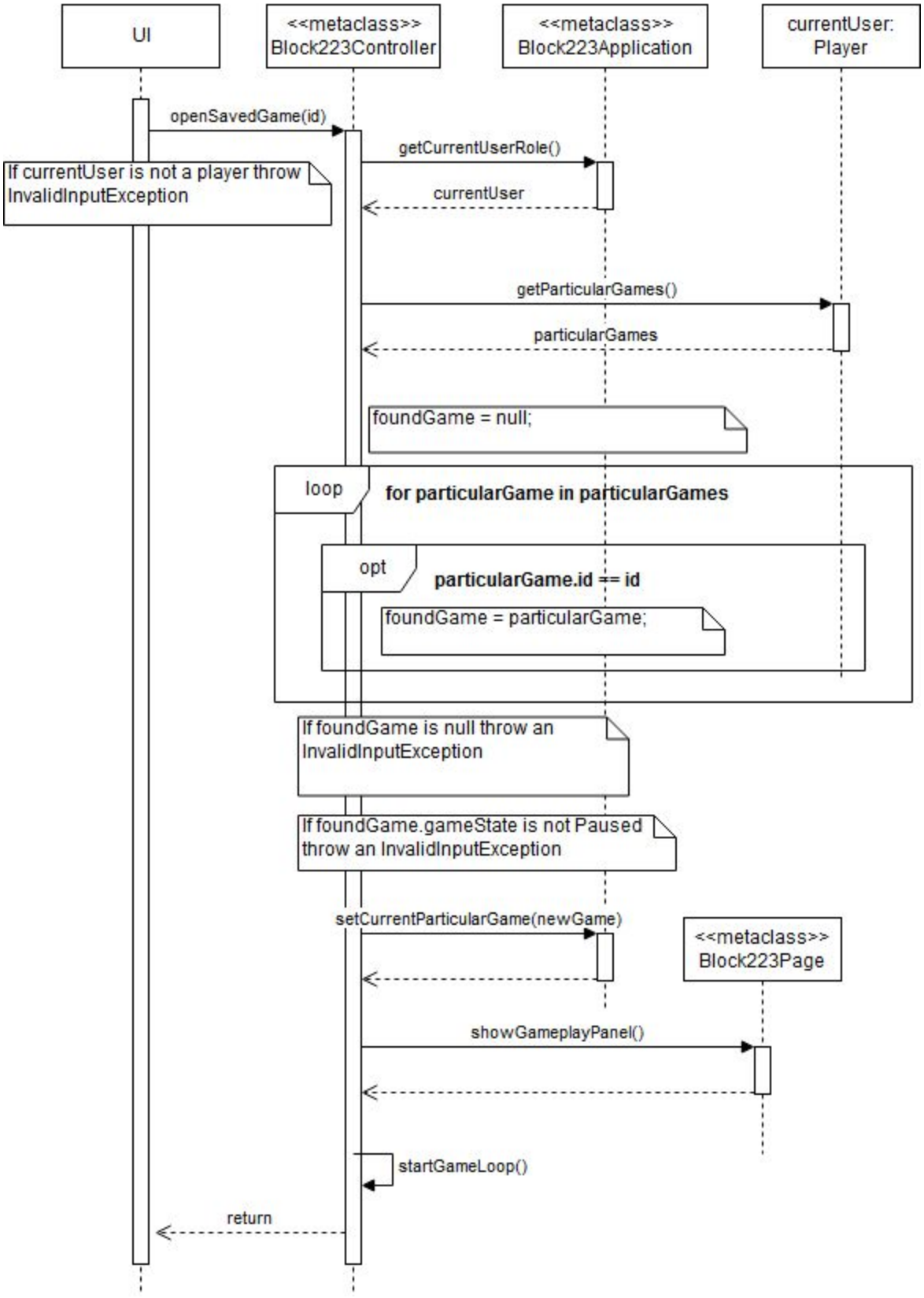
resumeGame()



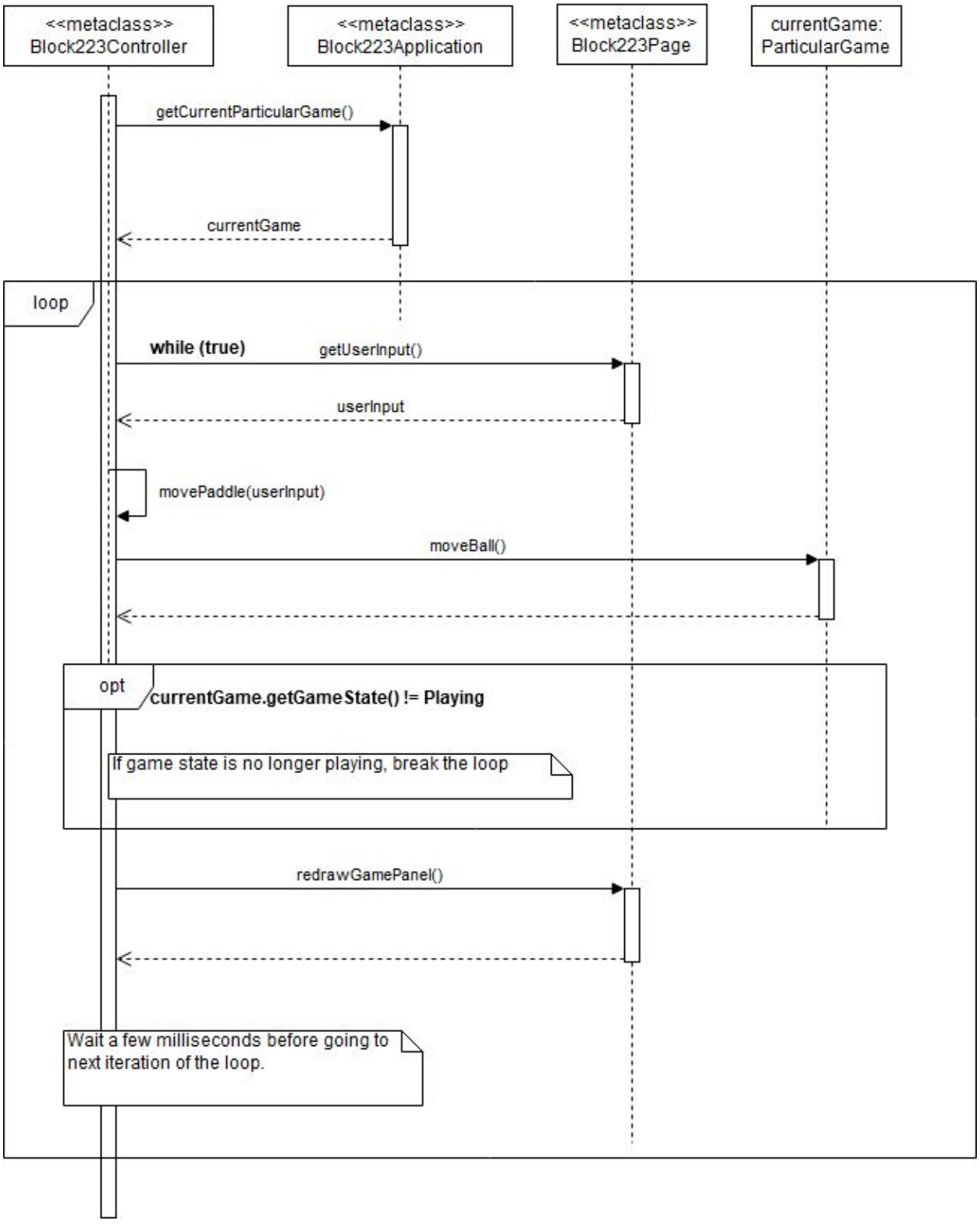
pauseGame()



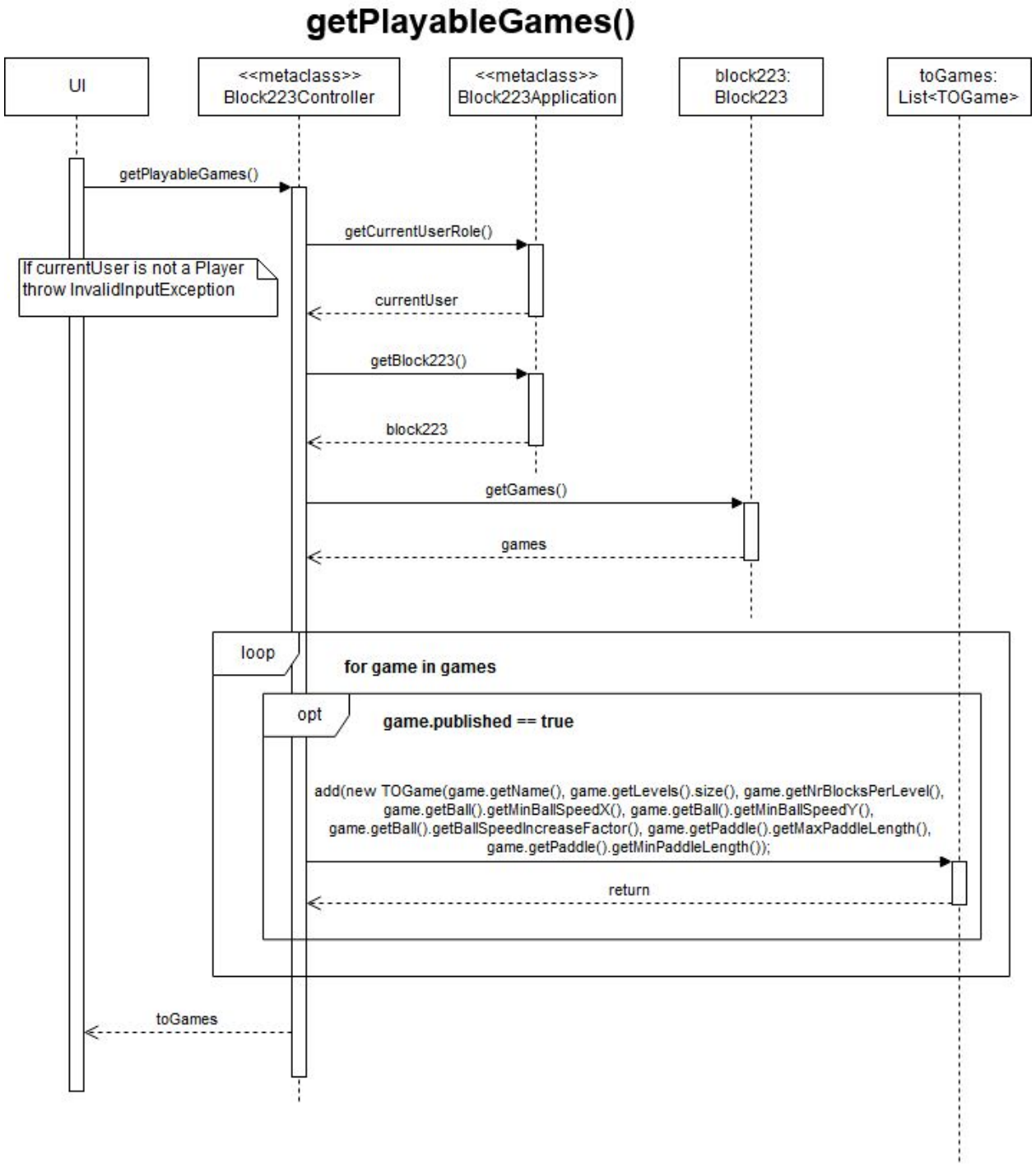
openSavedGame(int id)



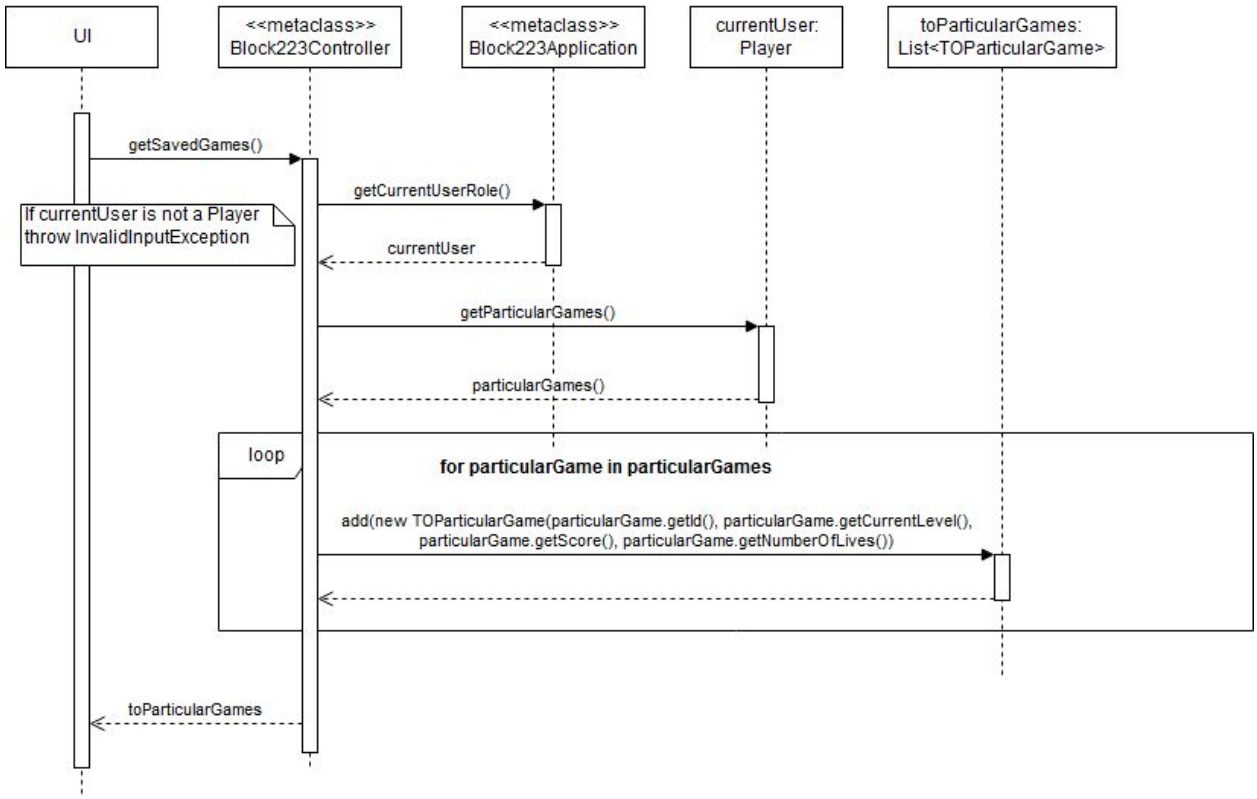
startGameLoop()



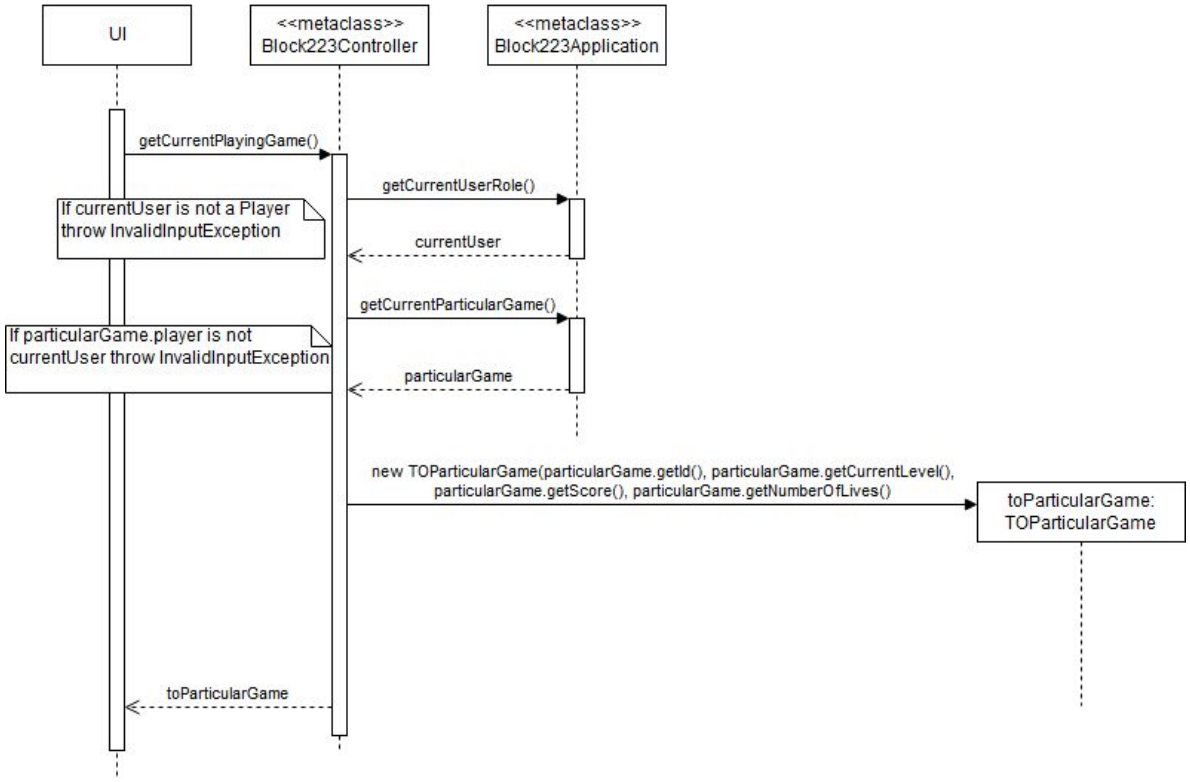
Query methods



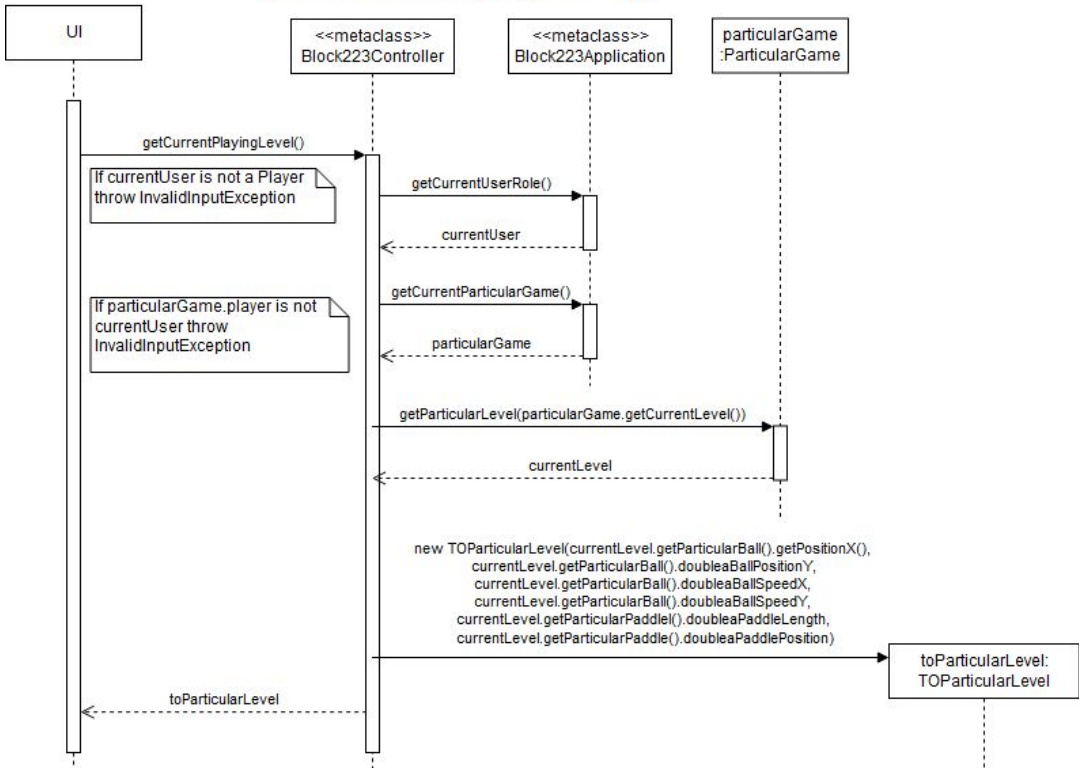
getSavedGames()



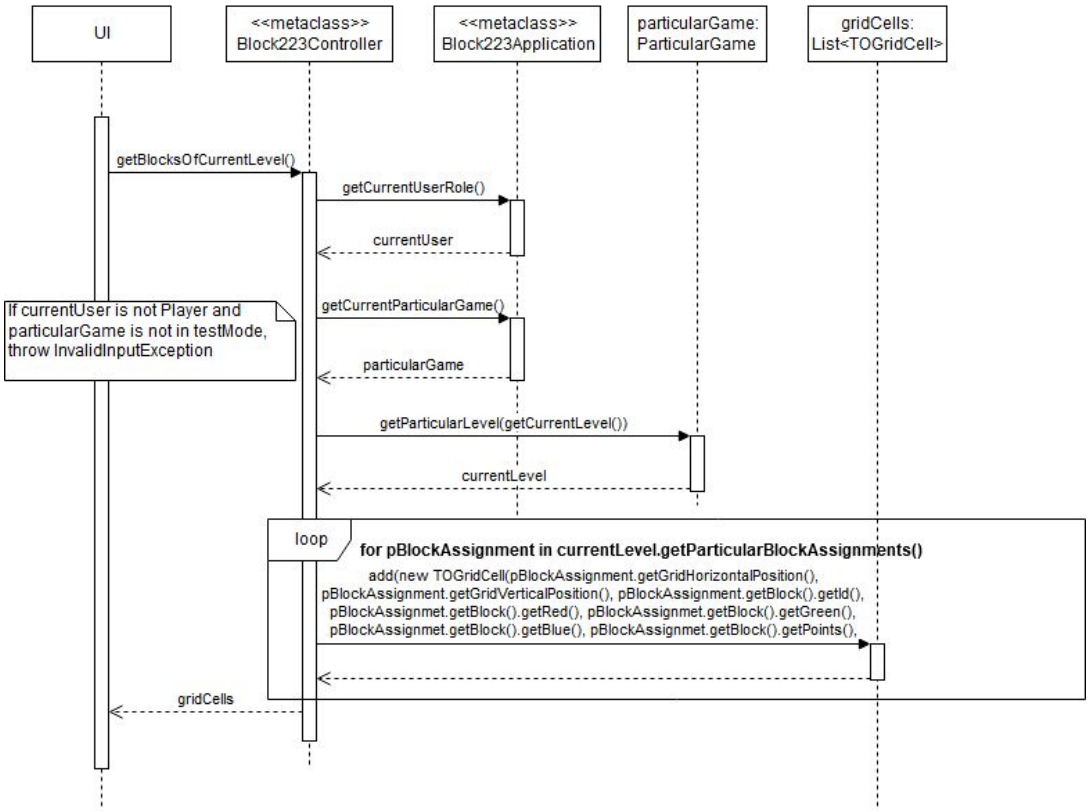
getCurrentPlayingGame()



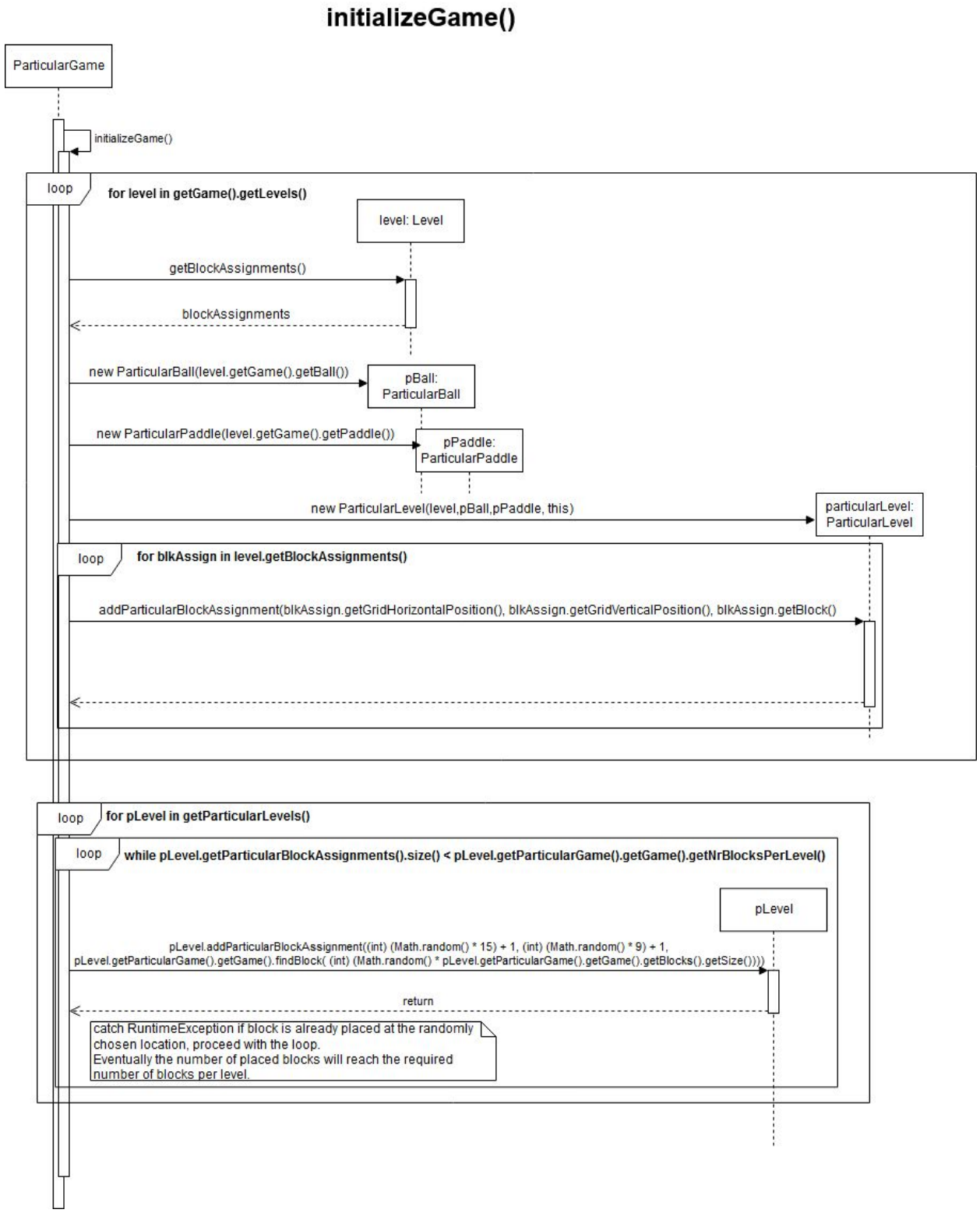
getCurrentPlayingLevel()



getBlocksOfCurrentLevel()



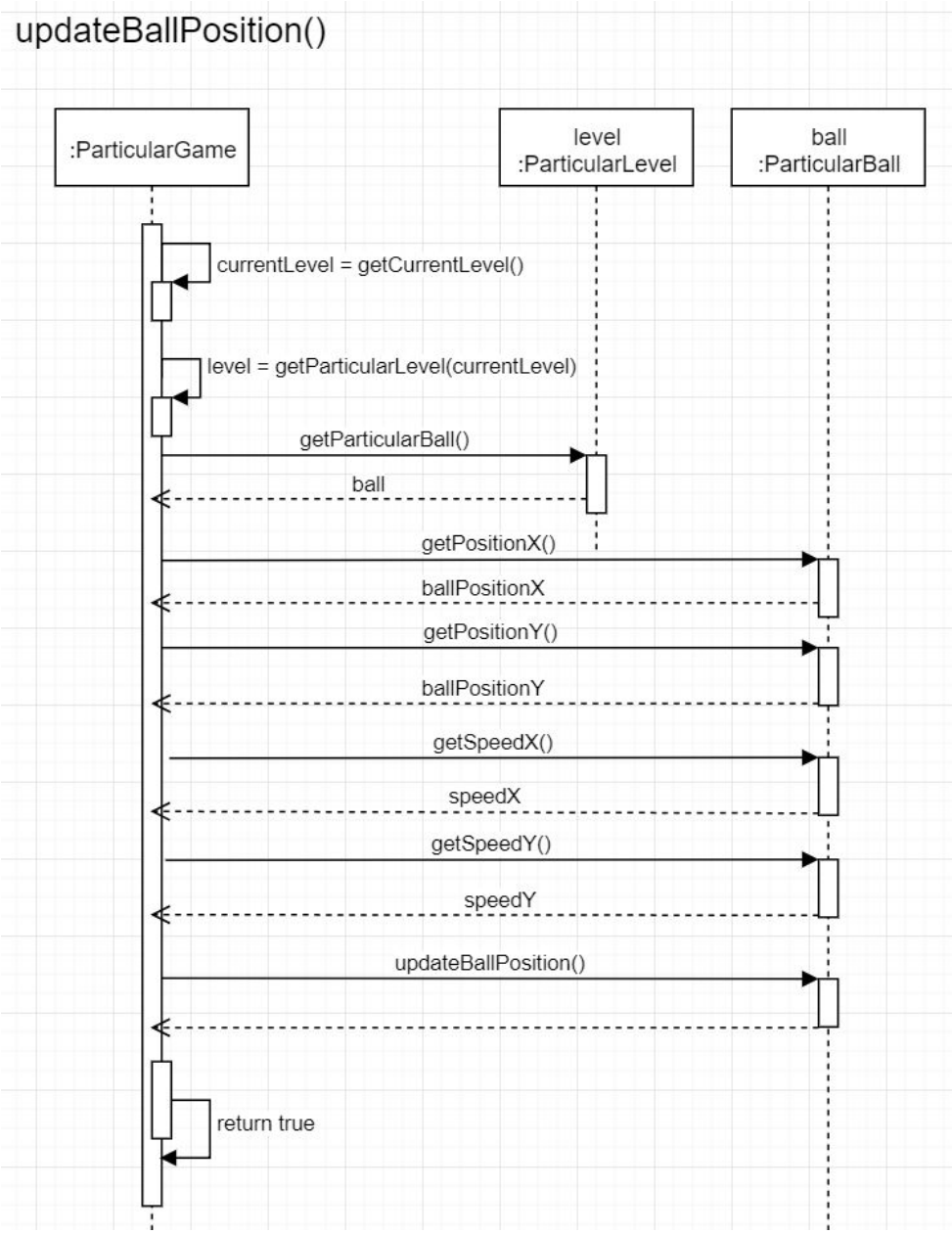
Private method for guard/action (State Machine)



2. **Move ball** (Helene Ma Yang)

- Sequence Diagrams

  Private method for guard/action *(State Machine)*
  private boolean ballHittingNothing();



3. **Ball hits paddle/wall** (Haowei Qiu)

- Sequence Diagrams

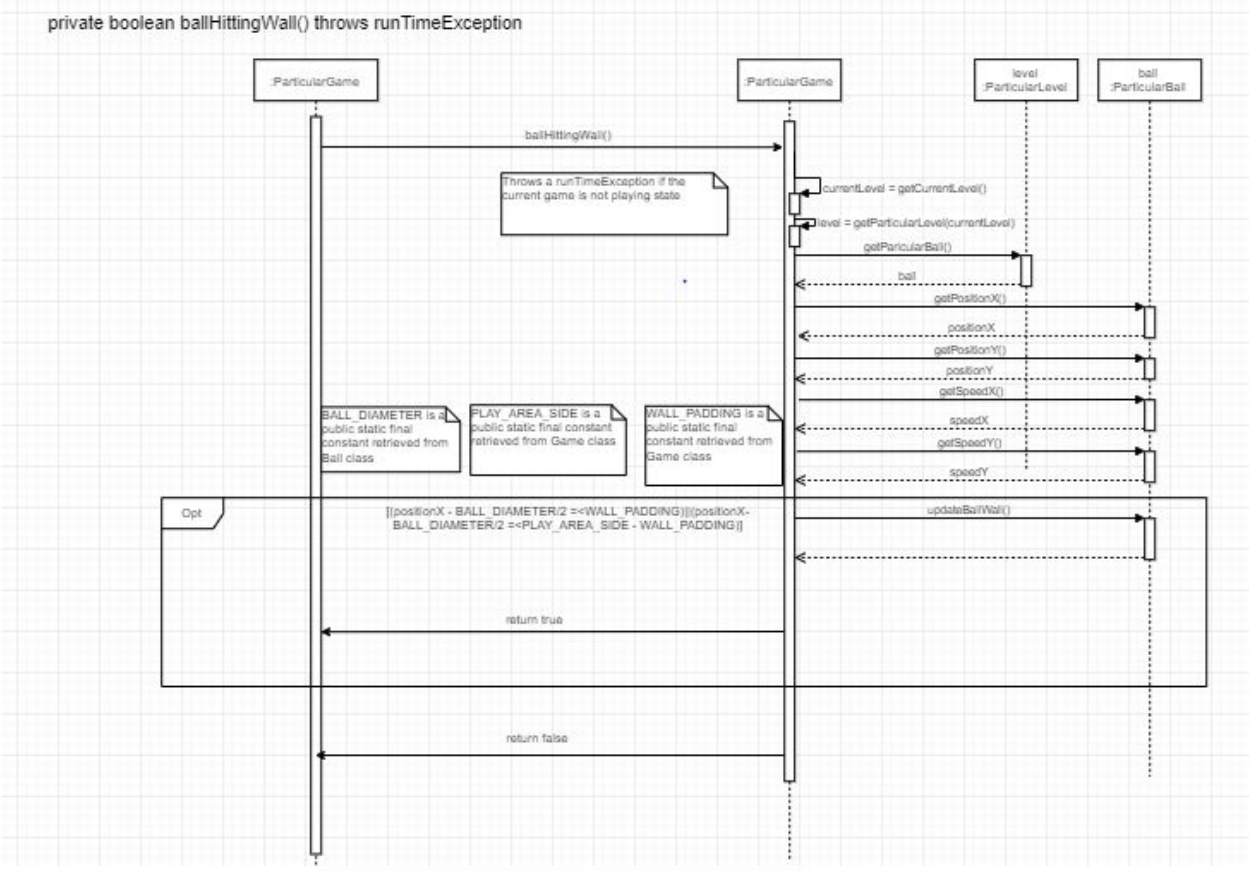
  Private method for guard/action (State Machine)

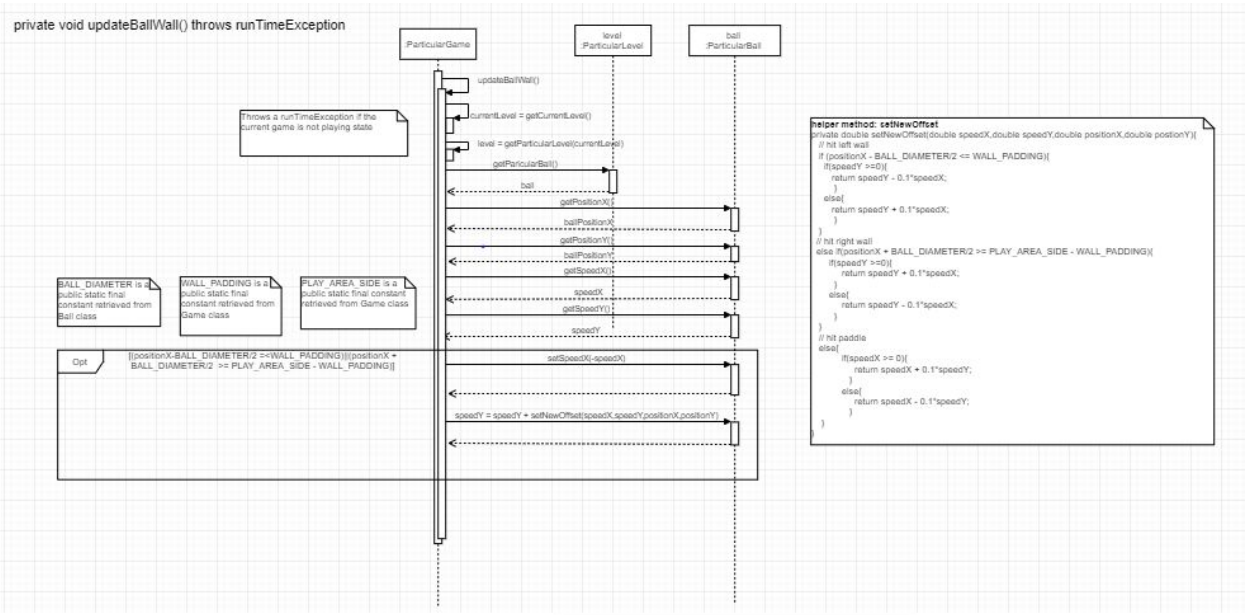
  private boolean ballHittingWall() throws runTimeException{}

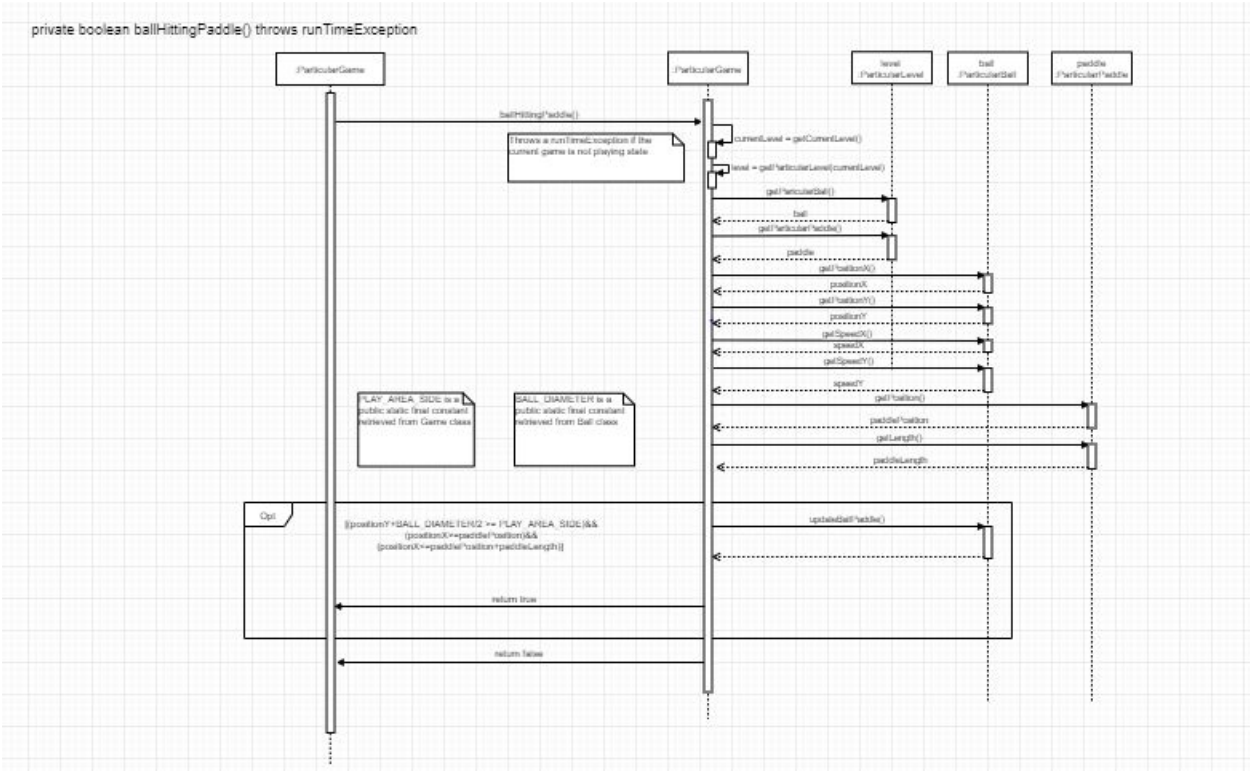
  private boolean ballHittingPaddle() throws runTimeException{}

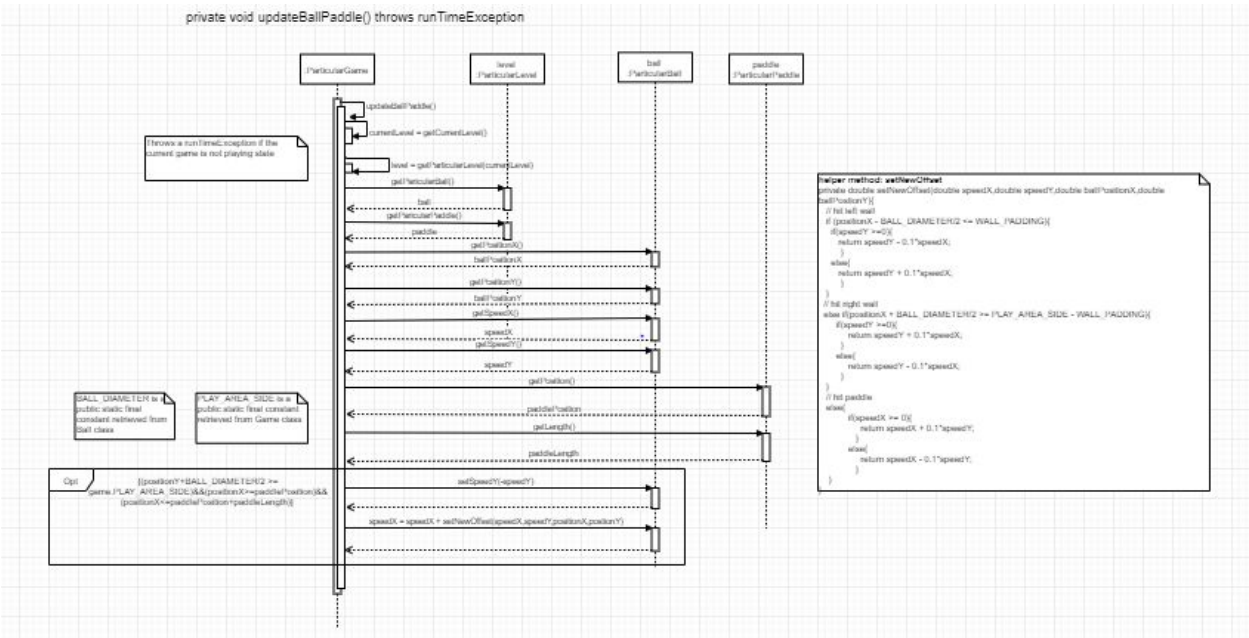
  private void updateBallWall() throws runTimeException{}

  private void updateBallPaddle() throws runTimeException{}




private void updateBallWall() throws runTimeException
:ParticularGame
level
:ParticularLevel
ball
:ParticularBall
updateBallWall()
Throws a runTimeException if the current game is not playing state
currentLevel = getCurrentLevel()
level = getParticularLevel(currentLevel)
getParticularBall()
ball
getPositionX()
ballPositionX
getPositionY()
ballPositionY
getSpeedX()
speedX
getSpeedY()
speedY
BALL_DIAMETER is a public static final constant retrieved from Ball class
WALL_PADDING is a public static final constant retrieved from Game class
PLAY_AREA_SIDE is a public static final constant retrieved from Game class
Opt
[(positionX-BALL_DIAMETER/2 =<WALL_PADDING)||(positionX + BALL_DIAMETER/2 >= PLAY_AREA_SIDE - WALL_PADDING)]
setSpeedX(-speedX)
speedY = speedY + setNewOffset(speedX,speedY,positionX,positionY)
helper method: setNewOffset
private double setNewOffset(double speedX,double speedY,double positionX,double positionY){
// hit left wall
if (positionX - BALL_DIAMETER/2 <= WALL_PADDING){
if(speedY >=0){
return speedY - 0.1*speedX;
}
else{
return speedY + 0.1*speedX;
}
}
// hit right wall
else if(positionX + BALL_DIAMETER/2 >= PLAY_AREA_SIDE - WALL_PADDING){
if(speedY >=0){
return speedY + 0.1*speedX;
}
else{
return speedY - 0.1*speedX;
}
}
// hit paddle
else{
if(speedX >= 0){
return speedX + 0.1*speedY;
}
else{
return speedX - 0.1*speedY;
}
}
}

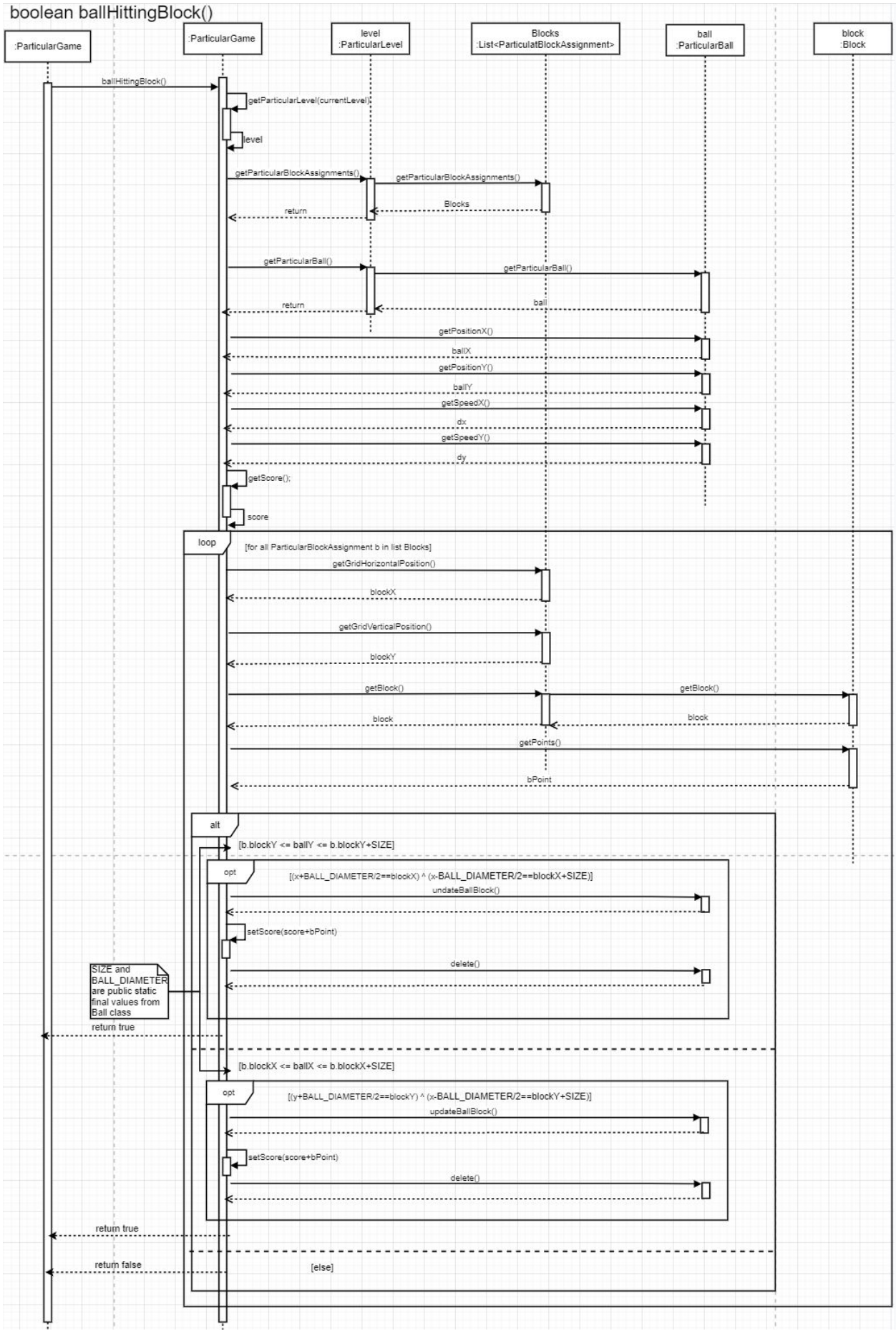

private boolean ballHittingPaddle() throws runTimeException
:ParticularGame
:ParticularGame
level
:ParticularLevel
ball
:ParticularBall
paddle
:ParticularPaddle
ballHittingPaddle()
Throws a runTimeException if the current game is not playing state
currentLevel = getCurrentLevel()
level = getParticularLevel(currentLevel)
getParticularBall()
ball
getParticularPaddle()
paddle
getPositionX()
positionX
getPositionY()
positionY
getSpeedX()
speedX
getSpeedY()
speedY
getPosition()
paddlePosition
getLength()
paddleLength
PLAY_AREA_SIDE is a public static final constant retrieved from Game class
BALL_DIAMETER is a public static final constant retrieved from Ball class
Opt
[(positionY+BALL_DIAMETER/2 >= PLAY_AREA_SIDE)&&
(positionX>=paddlePosition)&&
(positionX<=paddlePosition+paddleLength)]
updateBallPaddle()
return true
return false

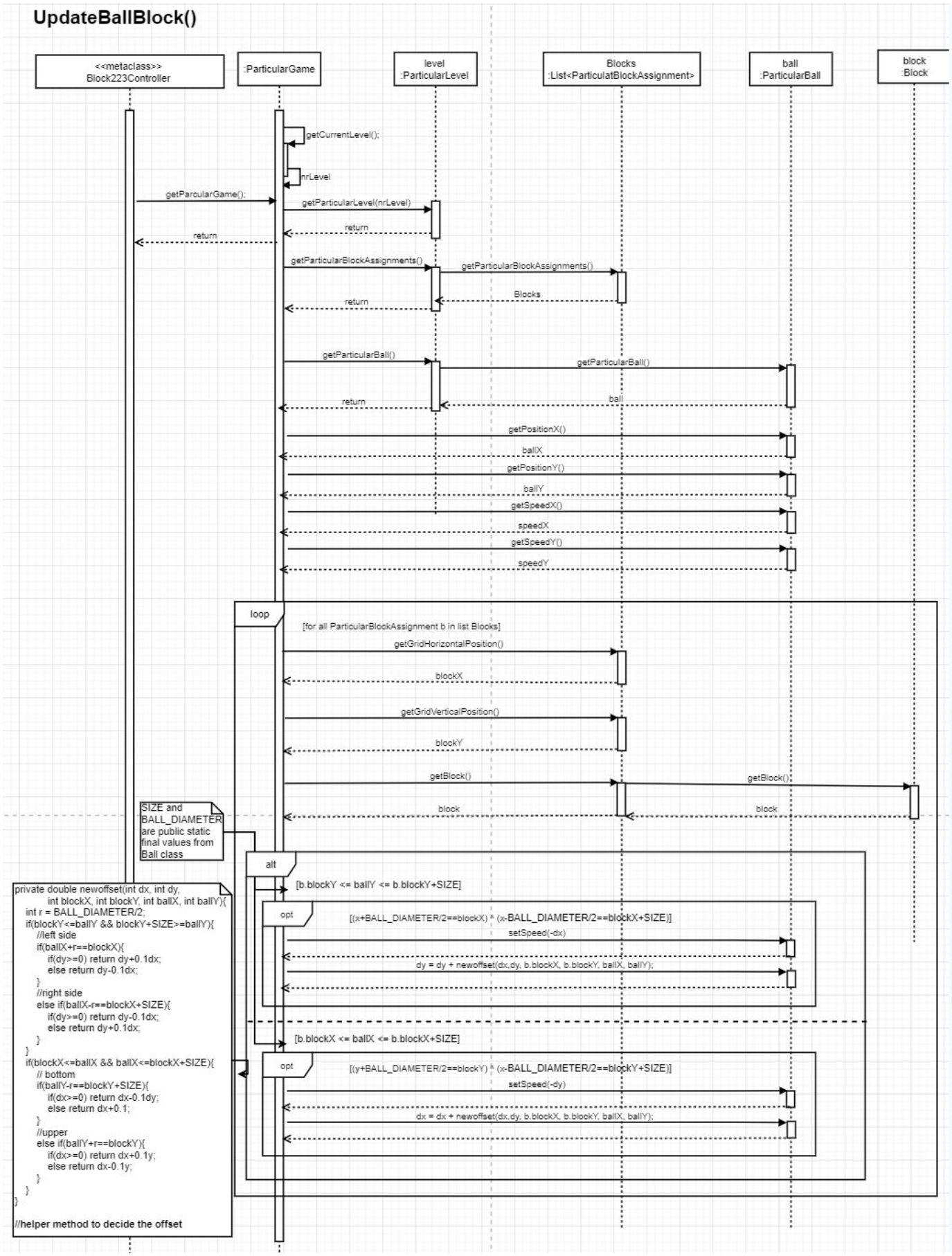


4. **Ball hits block** (Weijing Zhang)

   - Sequence Diagrams

     Private method for guard/action *(State Machine)*

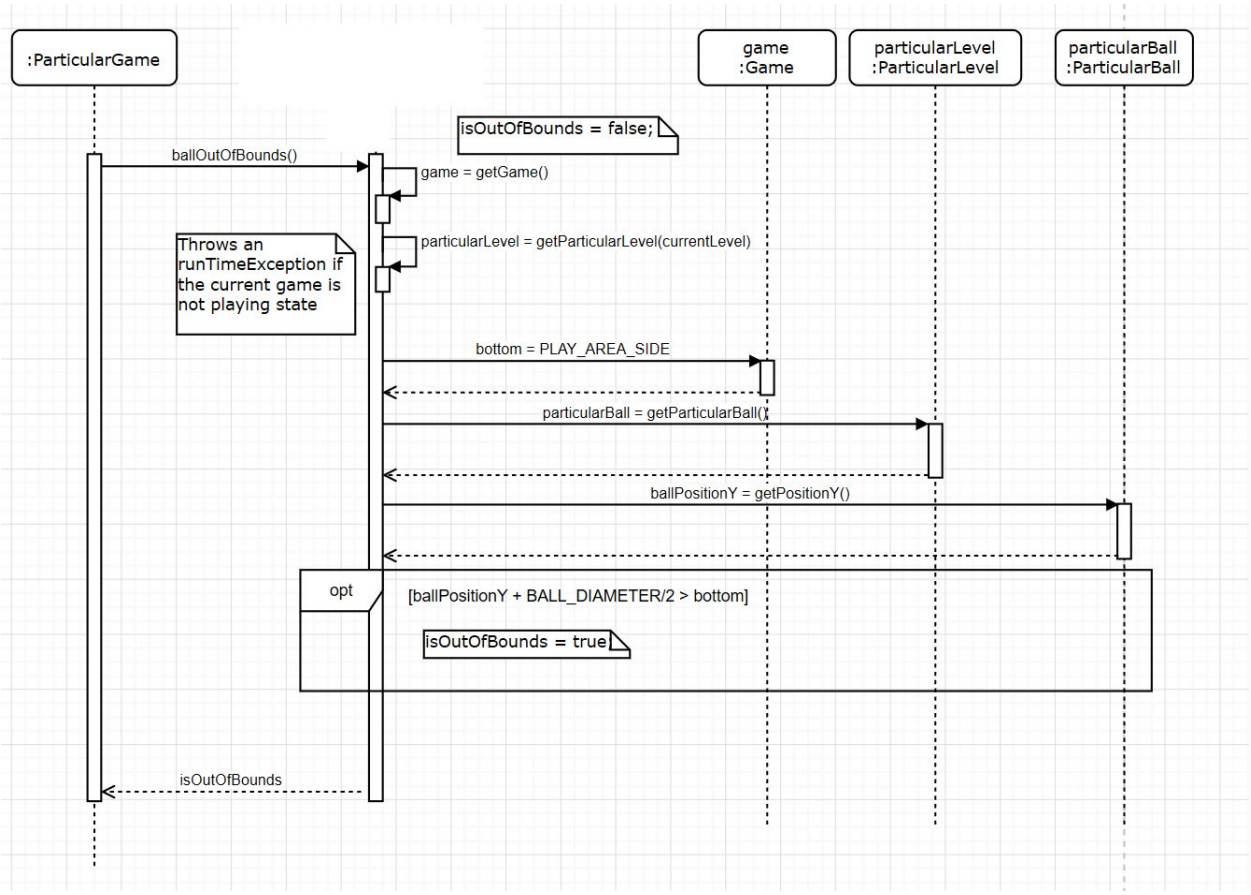

boolean ballHittingBlock()
:ParticularGame
:ParticularGame
level
:ParticularLevel
Blocks
:List<ParticulatBlockAssignment>
ball
:ParticularBall
block
:Block
ballHittingBlock()
getParticularLevel(currentLevel)
level
getParticularBlockAssignments()
getParticularBlockAssignments()
Blocks
return
getParticularBall()
getParticularBall()
ball
return
getPositionX()
ballX
getPositionY()
ballY
getSpeedX()
dx
getSpeedY()
dy
getScore():
score
loop
[for all ParticularBlockAssignment b in list Blocks]
getGridHorizontalPosition()
blockX
getGridVerticalPosition()
blockY
getBlock()
getBlock()
block
block
getPoints()
bPoint
alt
[b.blockY <= ballY <= b.blockY+SIZE]
opt
[(x+BALL_DIAMETER/2==blockX) ^ (x-BALL_DIAMETER/2==blockX+SIZE)]
undateBallBlock()
setScore(score+bPoint)
delete()
SIZE and BALL_DIAMETER are public static final values from Ball class
return true
[b.blockX <= ballX <= b.blockX+SIZE]
opt
[(y+BALL_DIAMETER/2==blockY) ^ (x-BALL_DIAMETER/2==blockY+SIZE)]
updateBallBlock()
setScore(score+bPoint)
delete()
return true
return false
[else]

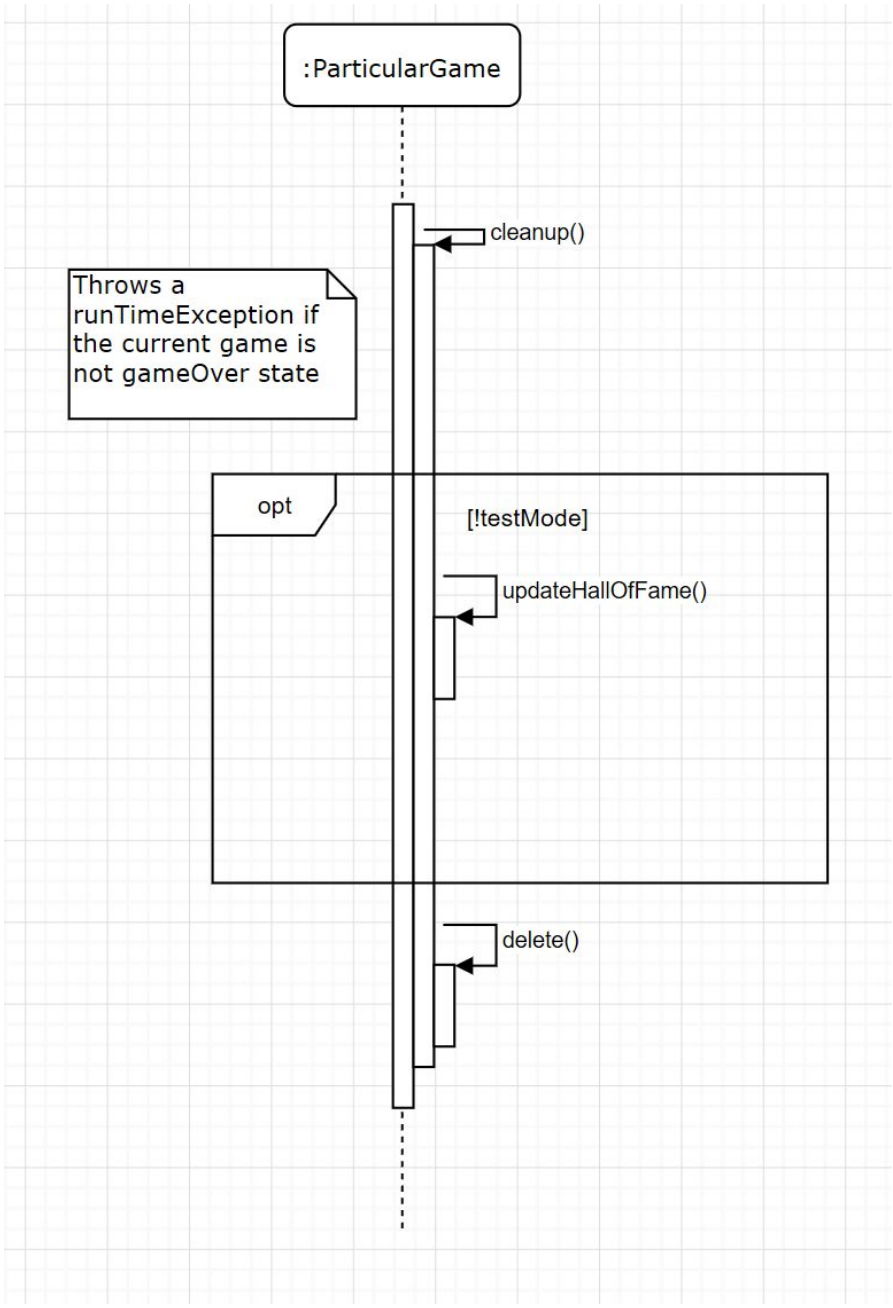

UpdateBallBlock()
<<metaclass>>
Block223Controller
:ParticularGame
level
:ParticularLevel
Blocks
:List<ParticulatBlockAssignment>
ball
:ParticularBall
block
:Block
getCurrentLevel();
nrLevel
getParcularGame();
getParticularLevel(nrLevel)
return
return
getParticularBlockAssignments()
getParticularBlockAssignments()
Blocks
return
getParticularBall()
getParticularBall()
return
ball
getPositionX()
ballX
getPositionY()
ballY
getSpeedX()
speedX
getSpeedY()
speedY
loop
[for all ParticularBlockAssignment b in list Blocks]
getGridHorizontalPosition()
blockX
getGridVerticalPosition()
blockY
getBlock()
getBlock()
block
block
SIZE and BALL_DIAMETER are public static final values from Ball class
alt
[b.blockY <= ballY <= b.blockY+SIZE]
opt
[(x+BALL_DIAMETER/2==blockX) ^ (x-BALL_DIAMETER/2==blockX+SIZE)]
setSpeed(-dx)
dy = dy + newoffset(dx,dy, b.blockX, b.blockY, ballX, ballY);
[b.blockX <= ballX <= b.blockX+SIZE]
opt
[(y+BALL_DIAMETER/2==blockY) ^ (x-BALL_DIAMETER/2==blockY+SIZE)]
setSpeed(-dy)
dx = dx + newoffset(dx,dy, b.blockX, b.blockY, ballX, ballY);
private double newoffset(int dx, int dy,
int blockX, int blockY, int ballX, int ballY){
int r = BALL_DIAMETER/2;
if(blockY<=ballY && blockY+SIZE>=ballY){
//left side
if(ballX+r==blockX){
if(dy>=0) return dy+0.1dx;
else return dy-0.1dx;
}
//right side
else if(ballX-r==blockX+SIZE){
if(dy>=0) return dy-0.1dx;
else return dy+0.1dx;
}
}
if(blockX<=ballX && ballX<=blockX+SIZE){
// bottom
if(ballY-r==blockY+SIZE){
if(dx>=0) return dx-0.1dy;
else return dx+0.1;
}
//upper
else if(ballY+r==blockY){
if(dx>=0) return dx+0.1y;
else return dx-0.1y;
}
}
}
//helper method to decide the offset

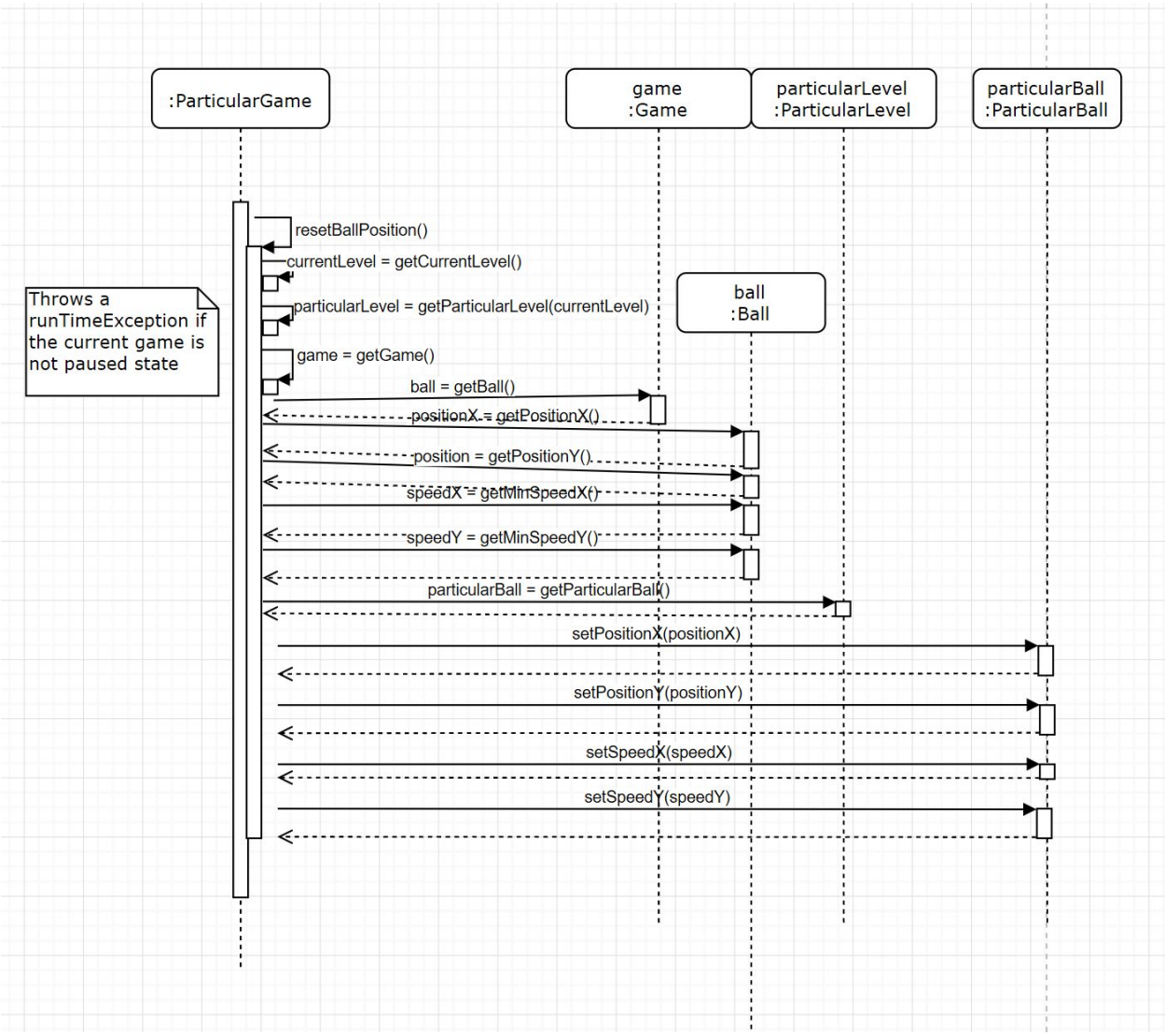
5. **Ball is out of bounds** (Xu Hai)

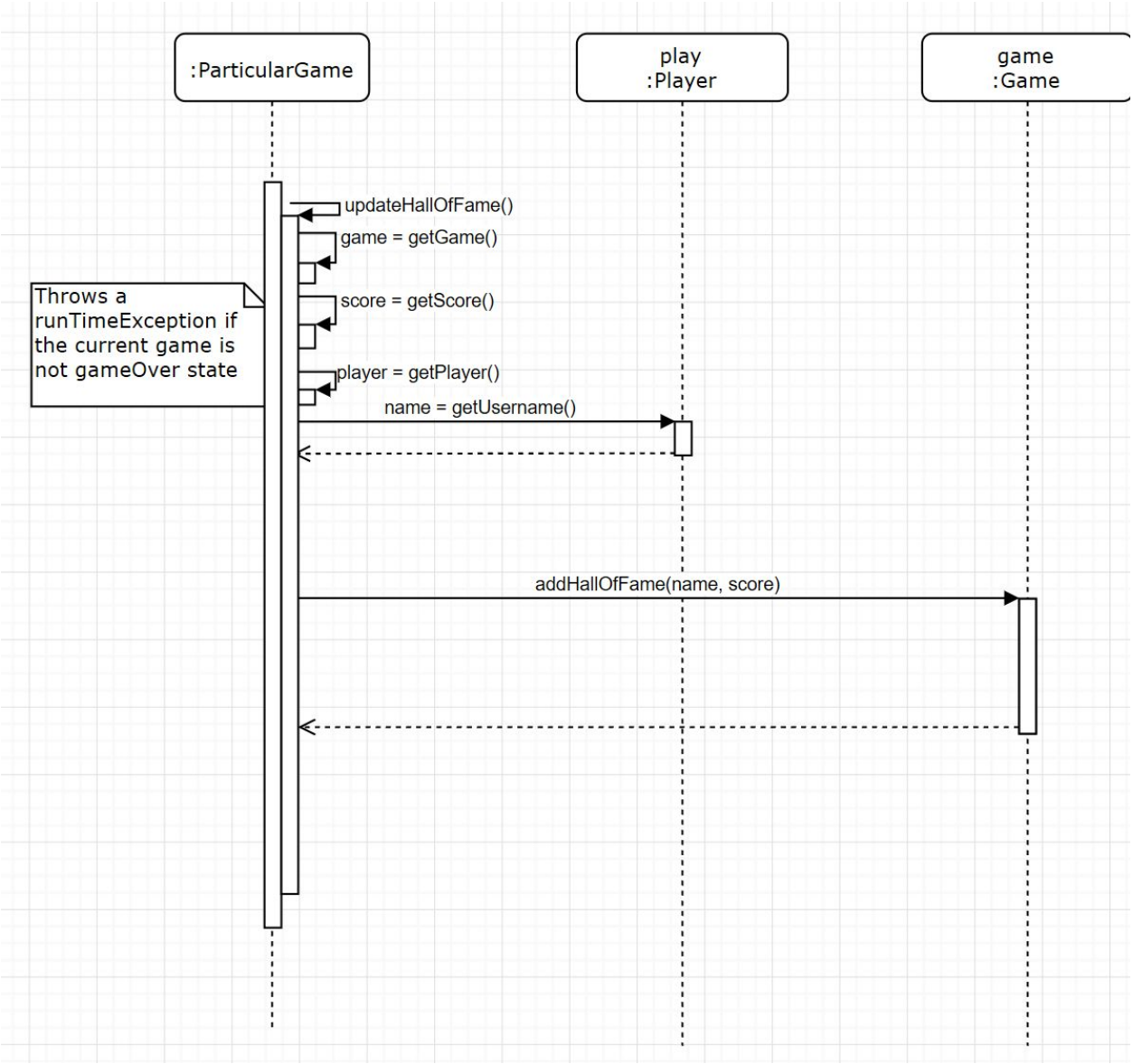
- Sequence Diagrams

  Private method for guard/action (State Machine)
  private boolean ballOutOfBounds() throws runTimeException{}
  private void cleanUp() throws runTimeException{}
  private void resetBall() throws runTimeException{}
  private void updateHallOfFame() throw runTimeException{};




:ParticularGame
cleanup()
Throws a runTimeException if the current game is not gameOver state
opt
[!testMode]
updateHallOfFame()
delete()


:ParticularGame
game
:Game
particularLevel
:ParticularLevel
particularBall
:ParticularBall
ball
:Ball
resetBallPosition()
currentLevel = getCurrentLevel()
particularLevel = getParticularLevel(currentLevel)
game = getGame()
ball = getBall()
positionX = getPositionX()
position = getPositionY()
speedX = getMinSpeedX()
speedY = getMinSpeedY()
particularBall = getParticularBall()
setPositionX(positionX)
setPositionY(positionY)
setSpeedX(speedX)
setSpeedY(speedY)
Throws a runTimeException if the current game is not paused state



6. **View hall of fame** (Zheng Yu Cui)

- Controller Interface (all members)

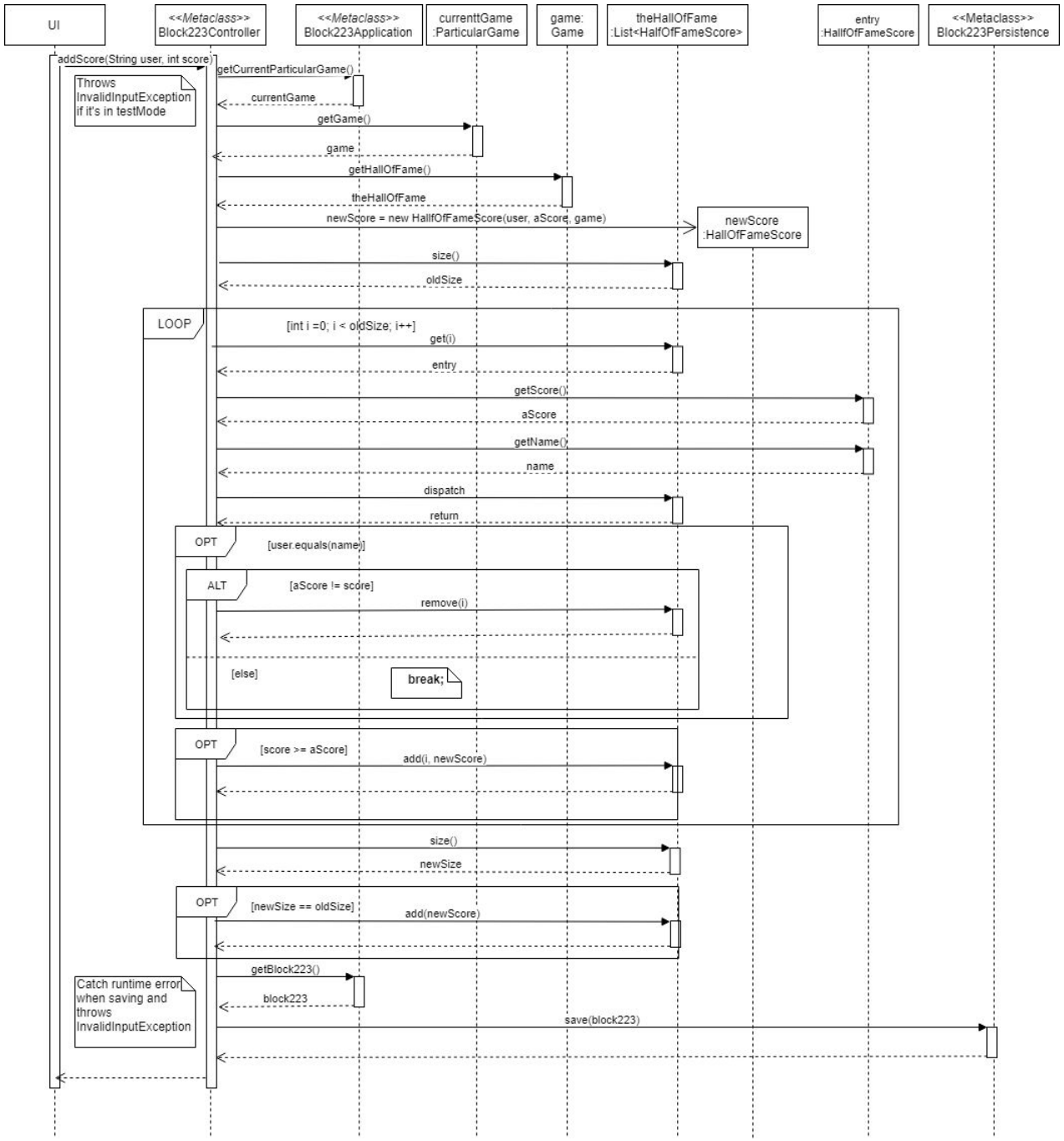
  Modifier methods
  public static void addScore(String user, int score) throw InvalidInputException;

  *Query methods *see sequence diagram(s) at the end of the document**
  public static List<TOHallOfFameScore> getHallOfFame();

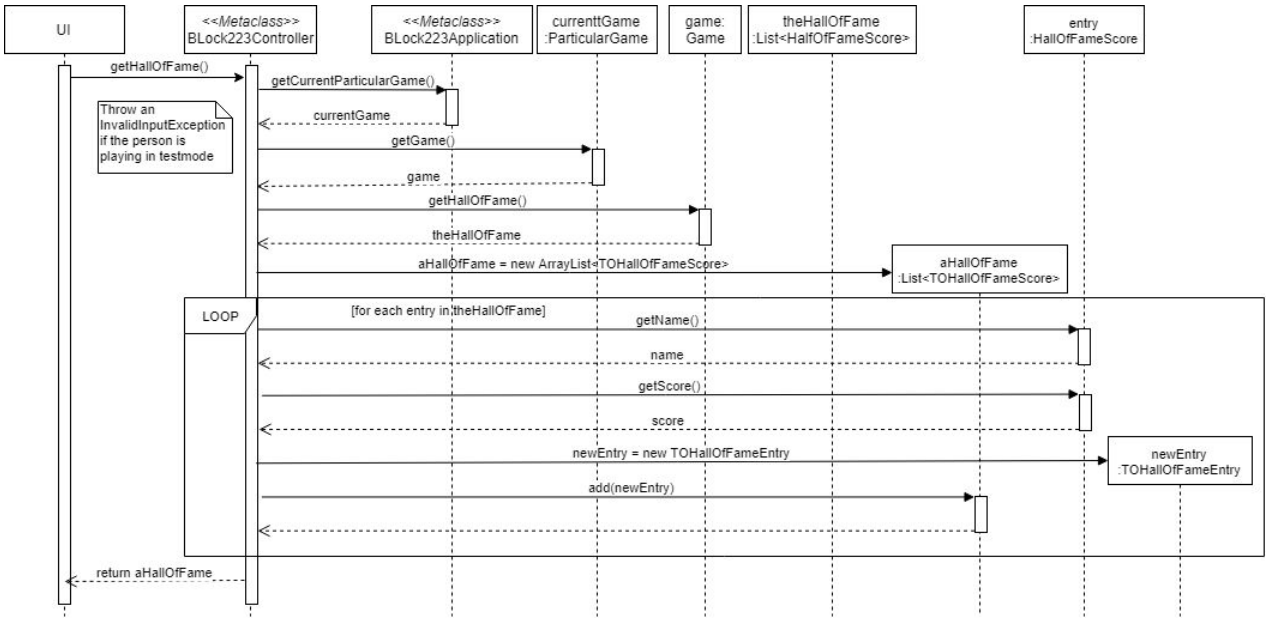
- Sequence Diagrams

Controller:

Modifier methods



Query methods



Private method for guard/action *(State Machine)*
Not applicable

7. **Test game** (Helene Ma Yang)

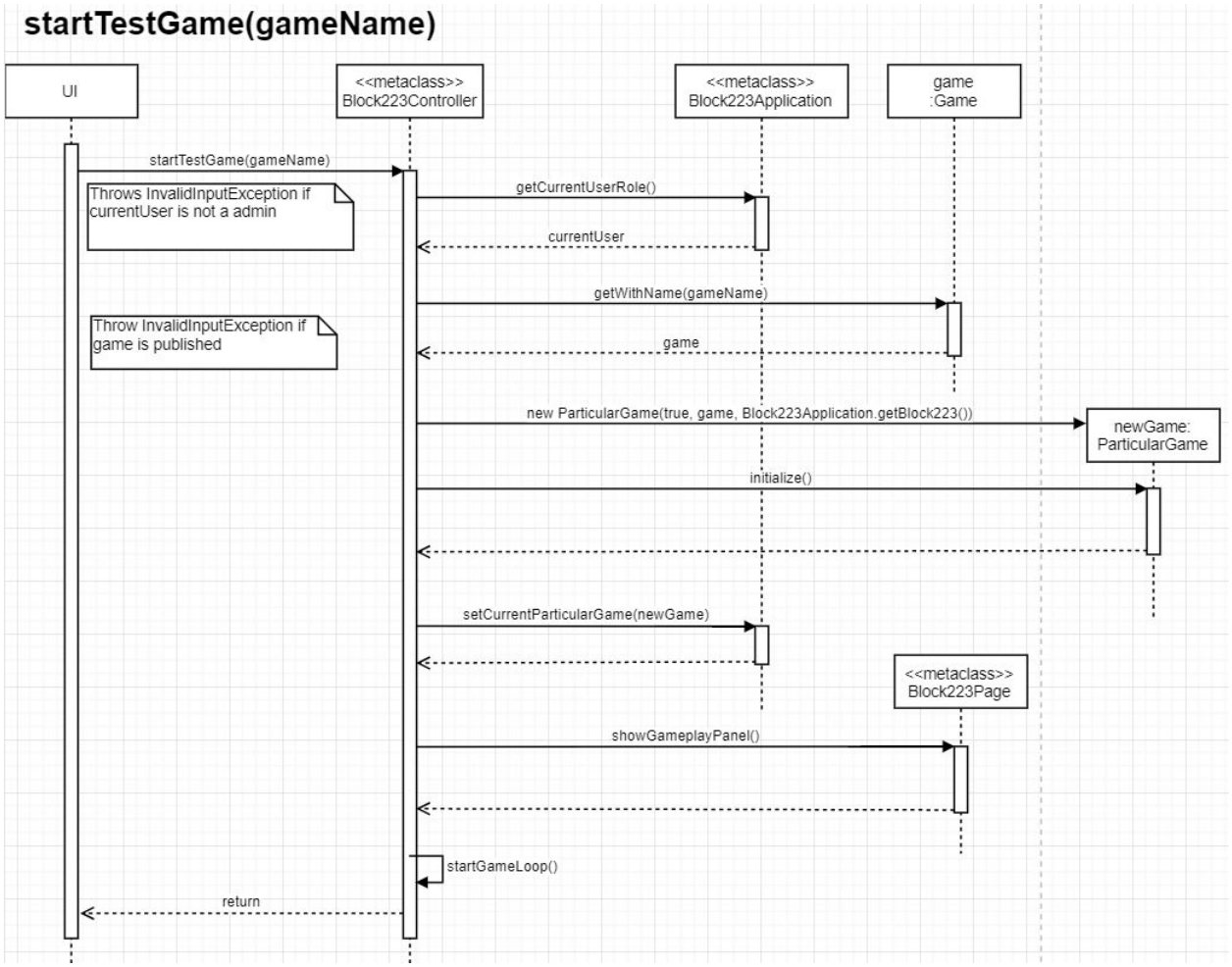
- Controller Interface (all members)

  Modifier methods
  public static void startTestGame(String gameName) throws InvalidInputException;

  Query methods
  public static TOParticularGame getCurrentPlayingGame() throws InvalidInputException;
  public static TOParticularLevel getCurrentPlayingLevel();
  public static List<TOGridCell> getBlocksOfCurrentLevel();

- Sequence Diagrams

  Controller:
  Modifier methods



Query methods
see sequence diagrams in startGame (feature 1)

Private method for guard/action (State Machine)
Not applicable

8. **Publish game** (Helene Ma Yang)

- Controller Interface (all members)

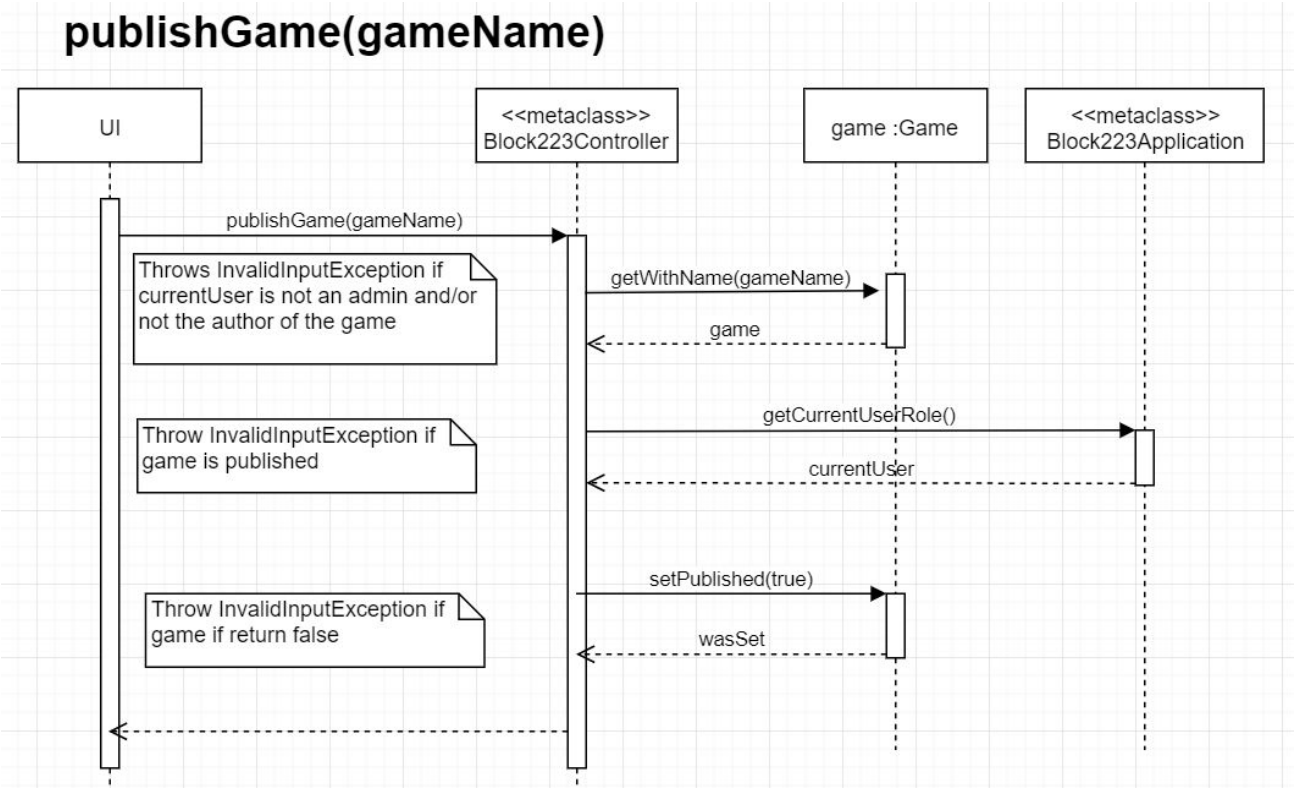
  Modifier methods (Controller)
  public static void publishGame(String gameName) throws InvalidInputException;

  Query methods
  Not applicable

- Sequence Diagrams

Controller:

Modifier methods



Query methods

Not applicable

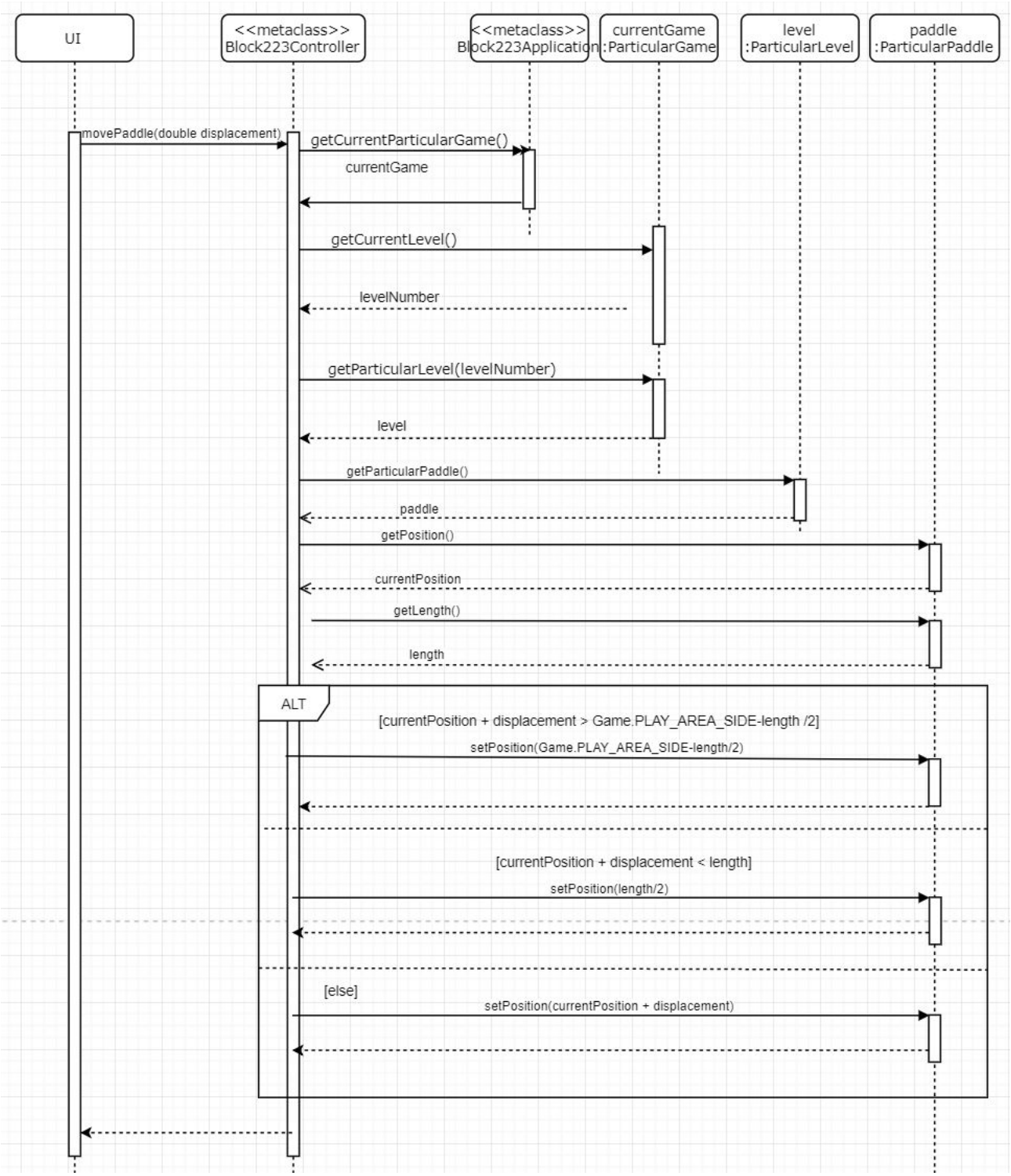
Private method for guard/action (State Machine)

Not applicable

9. **Move paddle** (all members)

- Sequence Diagrams

Private method for guard/action (State Machine)



Additional Validation Checks for Controller Methods from Iteration 2 & 3 (all members)

Controller Method	Condition	Error Message
updateGame	game.isPublished == true	A game cannot be modified after it is published.
setGameDetails	game.isPublished == true	A game's details cannot be modified after it is published.
addLevelToGame	game.isPublished == true	Cannot add a level to a game after it is published.
deleteGame	game.isPublished == true	Cannot delete a game after it is published.
addBlock	game.isPublished == true	Cannot add a block to a game after it is published.
deleteBlock	game.isPublished == true	Cannot delete a block in a game after it is published.
updateBlock	game.isPublished == true	Cannot update a block in a game after it is published.
positionBlock	game.isPublished == true	Cannot position a block in a level of a game that is published.
moveBlock	game.isPublished == true	Cannot move a block in a game after it is published.
removeBlock	game.isPublished == true	Cannot remove a block from a level of a game that is published.