```java
package Project;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class Test {
    public static void main(String[] args){

        ArrayList<String> wordList = new ArrayList<>();
        try {
            Scanner test = new Scanner(new FileInputStream("D:\\Uni Project\\Final Year Project\\res\\test.txt"));
            while (test.hasNext()){
                while (test.hasNext()){
                    String word = test.next();
                    if(!wordList.contains(word)){
                        wordList.add(word);
                    }
                }
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        System.out.println(wordList);

    }
}

```

```
 1 package Project;
 2
 3
 4 import java.awt.*;
 5 import java.awt.event.KeyEvent;
 6
 7 public class Circle {
 8     private int dx;
 9     private int dy;
10     private int x = 100;
11     private int y = 100;
12     private int w;
13     private int h;
14     private Image image;
15
16     public Circle(String team) {
17         loadImage(team);
18     }
19
20     private void loadImage(String team){
21
22
23         w = image.getWidth(null);
24         h = image.getHeight(null);
25     }
26
27     public void move() {
28
29         x += dx;
30         y += dy;
31     }
32
33     public int getX() {
34
35         return x;
36     }
37
38     public int getY() {
39
40         return y;
41     }
42
43     public int getWidth() {
44
45         return w;
```

```
46      }
47
48      public int getHeight() {
49
50          return h;
51      }
52
53      public Image getImage() {
54
55          return image;
56      }
57
58      public void keyPressed(KeyEvent e) {
59
60          int key = e.getKeyCode();
61
62          if (key == KeyEvent.VK_LEFT) {
63              dx = -2;
64          }
65
66          if (key == KeyEvent.VK_RIGHT) {
67              dx = 2;
68          }
69
70          if (key == KeyEvent.VK_UP) {
71              dy = -2;
72          }
73
74          if (key == KeyEvent.VK_DOWN) {
75              dy = 2;
76          }
77      }
78
79      public void keyReleased(KeyEvent e) {
80
81          int key = e.getKeyCode();
82
83          if (key == KeyEvent.VK_LEFT) {
84              dx = 0;
85          }
86
87          if (key == KeyEvent.VK_RIGHT) {
88              dx = 0;
89          }
90
```

```
 91          if (key == KeyEvent.VK_UP) {
 92              dy = 0;
 93          }
 94
 95          if (key == KeyEvent.VK_DOWN) {
 96              dy = 0;
 97          }
 98      }
 99 }
100
```

```java
package Project;

import Project.Base.Game;
import Project.Base.Enums.Team;


/**
 * Created by Leon on 16/01/2019.
 */
public class Tester {

    public static void main(String args[]){
        Game game = new Game(Team.TOR,Team.SFP);
        System.out.println(game.listAllPlayers());
    }
}
```

```java
package Project.GUI;

import Project.Base.Database;
import Project.Base.Game;
import Project.Base.Handler;
import Project.GUI.Assets.Assets;
import Project.Base.Enums.Team;
import Project.States.FaceoffState;
import Project.States.GameStartState;
import Project.States.SimState;
import Project.States.State;

import java.awt.*;
import java.awt.image.BufferStrategy;

public class Sim implements Runnable{

    private GUITest2 display;

    private Thread thread;

    public int width = 1211;
    public int height = 535;

    private boolean running = false;

    private BufferStrategy bs;
    private Graphics g;

    private Game game;

    //States
    public State simState;
    public State startState;
    public State faceoffState;
    private Handler handler;


    //Game Teams

    public Sim(){

    }

    private void init(){
```

```
46          display = new GUITest2();
47          Assets.init();
48          Database.init();
49          //Start Game
50          game = new Game(Team.TOR,Team.SFP);
51
52          handler = new Handler(this);
53          startState = new GameStartState(handler);
54          simState = new SimState(handler);
55          faceoffState = new FaceoffState(handler);
56
57          State.setState(startState);
58          startState.startGame();
59
60
61      }
62
63      public void setFaceoffState(int faceoffDot){
64          State.setState(faceoffState);
65          System.out.println("I'm here!!S");
66          faceoffState.setFaceoffDot(faceoffDot);
67      }
68
69
70
71      private void update(){
72          if(State.getState() != null){
73              State.getState().update();
74          }
75
76      }
77
78
79
80      private void render(){
81          bs = display.getCanvas().getBufferStrategy();
82          if(bs == null){
83              display.getCanvas().createBufferStrategy(3);
84              return;
85          }
86          g = bs.getDrawGraphics();
87          //clear screen
88          g.clearRect(0,0,width,height);
89
90          //draw items
```

```
 91          if(State.getState()!= null){
 92              State.getState().render(g);
 93          }
 94          bs.show();
 95          g.dispose();
 96      }
 97
 98      public Game getGame(){
 99          return game;
100      }
101
102      public void run(){
103          init();
104
105          //Set game frame rate
106          int fps = 15;
107          double timePerTick = 1000000000 / fps;
108          double delta = 0;
109          long now;
110          long lastTime = System.nanoTime();
111
112
113          //GAME LOOP!
114          while (running){
115              now = System.nanoTime();
116              delta += (now-lastTime)/timePerTick;
117              lastTime = now;
118
119              if (delta >= 1) {
120                  update();
121                  render();
122                  delta--;
123              }
124          }
125      }
126
127      public synchronized void start(){
128          if (running) {
129              return;
130          }
131           else{
132              running = true;
133          }
134          thread = new Thread(this);
135          thread.start();
```

```
136     }
137
138     public synchronized void stop(){
139         if (!running){
140             return;
141         }
142         else{
143             running = false;
144         }
145         try {
146             thread.join();
147         } catch (InterruptedException e) {
148             e.printStackTrace();
149         }
150     }
151
152
153 }
154
```

```java
package Project.GUI;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.event.KeyAdapter;

public class GUITest2{

    private JFrame frame;
    private Canvas canvas;

    private String title = "Hockey Sim Engine";
    private int width = 1211;
    private int height = 535;

    public GUITest2(){

        //init Jframe
        frame = new JFrame(title);
        frame.setSize(width,height);
        frame.setDefaultCloseOperation(JFrame.
EXIT_ON_CLOSE);
        frame.setResizable(false);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);


        //init canvas and set size
        canvas = new Canvas();
        canvas.setPreferredSize(new Dimension(width,height
));
        canvas.setMaximumSize(new Dimension(width,height))
;
        canvas.setMinimumSize(new Dimension(width,height))
;

        frame.add(canvas);
        frame.pack();
    }

    public Canvas getCanvas() {
        return canvas;
    }
}
```

```
1 package Project.GUI;
2
3 public class Launcher {
4     public static void main(String[] args){
5         new Sim().start();
6
7     }
8 }
9
```

```
 1 package Project.GUI.Assets;
 2
 3 import Project.Base.Database;
 4 import Project.Base.Enums.Team;
 5
 6 import java.awt.image.BufferedImage;
 7 import java.util.ArrayList;
 8 import java.util.Objects;
 9
10 public class Assets {
11     public static BufferedImage rink;
12     public static SpriteSheet homeSheet,awaySheet;
13     public static SpriteSheet homeNumberSheet,
   awayNumberSheet;
14     public static BufferedImage homeTeam,awayTeam;
15     public static ArrayList<BufferedImage> homeNumbers =
   new ArrayList<BufferedImage>(10);
16     public static ArrayList<BufferedImage> awayNumbers =
   new ArrayList<BufferedImage>(10);
17     public static BufferedImage numberSpace;
18     public static BufferedImage puck;
19
20     private static final int height = 21;
21     private static final int width = 21;
22
23
24     public static void init(){
25
26         rink = ImageLoader.loadImage("/IceRinkV1.png");
27         puck = ImageLoader.loadImage("/Puck.png");
28         homeSheet = new SpriteSheet(ImageLoader.loadImage(
   "/SpritesheetHomeTeams.png"));
29         awaySheet = new SpriteSheet(ImageLoader.loadImage(
   "/SpritesheetAwayTeams.png"));
30         homeNumberSheet = new SpriteSheet(ImageLoader.
   loadImage("/SpritesheetHomeNumbers.png"));
31         awayNumberSheet = new SpriteSheet(ImageLoader.
   loadImage("/SpritesheetAwayNumbers.png"));
32         numberSpace = homeNumberSheet.crop(6,0,1,9);
33     }
34
35
36     public static void loadTeamAssets(Team home, Team away
   ){
37         int[] position;
```

```
38          int teamNo;
39          BufferedImage image;
40
41          //Home Team
42          position = Database.getTeamPosition(home);
43          homeTeam = homeSheet.crop((Objects.requireNonNull(
   position)[0]*width),(position[1]*height),width,height);
44          teamNo = Database.getTeamNumber(home);
45          for (int i=0;i<10;i++){
46              image = homeNumberSheet.crop(i*5,teamNo*9,5,9)
   ;
47              homeNumbers.add(image);
48          }
49
50          //Away team
51          position = Database.getTeamPosition(away);
52          awayTeam = awaySheet.crop((Objects.requireNonNull(
   position)[0]*width),(position[1]*height),width,height);
53          teamNo = Database.getTeamNumber(away);
54          for (int i=0;i<10;i++){
55              image = awayNumberSheet.crop(i*5,teamNo*9,5,9)
   ;
56              awayNumbers.add(image);
57          }
58      }
59  }
60
```

```java
package Project.GUI.Assets;

import java.awt.Image;
import javax.swing.ImageIcon;

public class Sprite {

    protected int x;
    protected int y;
    protected int width;
    protected int height;
    protected boolean visible;
    protected Image image;

    public Sprite(int x, int y) {

        this.x = x;
        this.y = y;
        visible = true;
    }

    public void loadImage(String file) {
        ImageIcon ii = new ImageIcon(file);
        image = ii.getImage();
    }

    protected void getImageDimensions() {

        width = image.getWidth(null);
        height = image.getHeight(null);
    }

    public Image getImage() {
        return image;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public boolean isVisible() {
```

```
46          return visible;
47      }
48
49      public void setVisible(Boolean visible) {
50          this.visible = visible;
51      }
52  }
53
```

```java
package Project.GUI.Assets;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.IOException;


public class ImageLoader {

    BufferedImage red;

    public static BufferedImage loadImage(String path){
        try {
            return ImageIO.read(ImageLoader.class.
  getResource(path));
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(1);
        }
        return null;
    }
}

```

```java
package Project.GUI.Assets;

import java.awt.image.BufferedImage;

public class SpriteSheet {
    private BufferedImage sheet;

    public SpriteSheet(BufferedImage sheet){
        this.sheet = sheet;
    }

    public BufferedImage crop(int x,int y, int width, int height){
        return sheet.getSubimage(x,y,width,height);
    }
}

```

```java
package Project.GUI.Entities;

import Project.Base.Arena;
import Project.Base.Enums.Possession;
import Project.Base.Game;
import Project.GUI.Assets.Assets;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import java.io.Console;
import java.util.Random;

import static java.lang.Math.*;

public class Puck extends Entity {

    private final PropertyChangeSupport support = new PropertyChangeSupport(this);
    private Possession lastTouch;
    private int direction = 1;
    private float angle;
    private int puckTimer =0;

    public BufferedImage puck = Assets.puck;
    Arena arena = Arena.getArena();
    Random rand = new Random();


    public Puck(Game game) {
        super(5, 5,5,5);
        this.game = game;
    }


    //observer design pattern
    public void addPropertyChangeListener(PropertyChangeListener pcl){
        support.addPropertyChangeListener(pcl);
    }

    public void removePropertyChangeListener(PropertyChangeListener pcl){
        support.removePropertyChangeListener(pcl);
```

```
43      }
44
45      //broadcast change
46      public void setLastTouch(Possession touch){
47          support.firePropertyChange("lastTouch",this.
    lastTouch,touch);
48          this.lastTouch = touch;
49      }
50
51      public void setSpeed(int velocity) {
52          this.xVelocity = velocity;
53          this.yVelocity = velocity;
54      }
55
56      public void setAngeleCalculate(){
57          var deltaX = targetX - x;
58          var deltaY = targetY - y;
59          var rad = Math.atan2(deltaY,deltaX);
60          angle = (float) rad;
61      }
62
63      public void setAngle(float angle){
64          this.angle = angle;
65      }
66
67      private void movePuck() {
68
69          //System.out.println(targetX + " " + targetY + "
    PUCK" + xVelocity);
70          float decel = 0.01f;
71          if (xVelocity > 0.1) {
72              if (xVelocity >= 0) {
73                  xVelocity = xVelocity - decel;
74              } else {
75                  xVelocity = xVelocity + decel;
76              }
77          }
78          if (yVelocity > 0.1) {
79              if (yVelocity < 0) {
80                  yVelocity = yVelocity - decel;
81              } else {
82                  yVelocity = yVelocity + decel;
83              }
84          }
85
```

```
 86          float tx1 = (float) (x + xVelocity * cos(angle));
 87          float tx2 = (float) (x + bounds.width + xVelocity
     * cos(angle));
 88          float ty1 = (float) (y + yVelocity * sin(angle));
 89          float ty2 = (float) (y + bounds.height +
    yVelocity * sin(angle));
 90          if(game.getArena().legalMove(tx1,tx2,ty1,ty2)) {
 91              //System.out.println(x + " " + y);
 92              //System.out.println( xVelocity * cos(angle
    ));
 93              //System.out.println( yVelocity * sin(angle
    ));
 94              x += xVelocity * cos(angle);
 95              y += yVelocity * sin(angle);
 96              //System.out.println("MOVING");
 97          }
 98          else {
 99              int bounce = game.getArena().getBounce(x,y);
100              if (bounce == 1){
101                  yVelocity = -yVelocity;
102              }
103              else if (bounce == 2){
104                  xVelocity = -xVelocity;
105              }
106              else {
107                  xVelocity = -xVelocity;
108                  yVelocity = -yVelocity;
109              }
110
111          }
112      }
113
114      public Possession getLastTouch(){
115          return lastTouch;
116      }
117
118      public int getZone(){
119          return arena.getThird(this);
120      }
121
122      public int getSpeed(){
123          return Math.abs((int)xVelocity);
124      }
125
126      public float getVelocityX(){
```

```
127         return xVelocity;
128     }
129 
130     public float getVelocityY(){
131         return yVelocity;
132     }
133 
134     public void resetPuckTimer(){
135         puckTimer = 0;
136     }
137 
138     public int getPuckTimer() {
139         return puckTimer;
140     }
141 
142     private void tickPuckTimer(){
143         puckTimer++;
144     }
145 
146     @Override
147     public void tick() {
148         movePuck();
149         updateBounds(x,y);
150         tickPuckTimer();
151     }
152 
153     @Override
154     public void render(Graphics g) {
155         g.drawImage(puck,(int) x,(int) y,null);
156 //          g.setColor(Color.red);
157 //          g.fillRect(bounds.x,bounds.y,bounds.width,
    bounds.height);
158 
159     }
160 }
161 
```

```java
package Project.GUI.Entities;

import Project.Base.Enums.Position;
import Project.Base.Game;
import Project.GUI.Entities.Player.Player;

import java.awt.*;
import java.util.ArrayList;

import static java.lang.Math.sqrt;

public abstract class Entity {
    protected Game game;

    protected float x,y;
    protected int width,height;
    protected float targetX,targetY;
    protected float xVelocity,yVelocity;
    protected Position currentPosition;
    protected Rectangle bounds;

    public Entity(float x,float y,int width, int height){
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        bounds = new Rectangle((int)x,(int) y,width,height);
    }

    public void setPositionAbsolute(int x,int y){
        this.x = x - width/2;
        this.y = y - height/2;
    }

    protected void updateBounds(float x, float y){
        bounds.x = (int) x;
        bounds.y = (int) y;
    }

    public Rectangle getCollisionBounds(float xOffset, float yOffset){
        return new Rectangle((int)(bounds.x+xOffset),(int)(bounds.y+yOffset));
    }
```

```
43
44      public boolean checkPlayerCollision(float xOffset,
    float yOffset){
45          for (Player p:game.getAllPlayers()){
46              if(p.equals(this)){
47                  continue;
48              }
49  //              System.out.println(p.getCollisionBounds(0f,
    0f));
50  //              System.out.println(getCollisionBounds(
    xOffset,yOffset));
51  //              if (p.getCollisionBounds(0f,0f).intersects(
    getCollisionBounds(xOffset,yOffset))){
52  //                  return true;
53  //              }
54              if(getDistance(this,p)<6){
55                  return true;
56              }
57              else return false;
58          }
59          return false;
60      }
61
62      private float getDistance(Entity a, Entity b){
63          float deltaX = targetX - x;
64          float deltaY = targetY - y;
65          return  (float) sqrt(Math.pow(deltaX, 2) + Math.
    pow(deltaY, 2));
66      }
67
68      public Player getHittingPlayer(Player player,ArrayList
    <Player> playerArrayList,float xOffset, float yOffset){
69          for(Player p:playerArrayList){
70              if(p.getCollisionBounds(0f,0f).intersects(
    getCollisionBounds(xOffset,yOffset))){
71                  return p;
72              }
73          }
74          return null;
75      }
76
77      public boolean playerPuckCollision(Player player){
78          if (player.getCollisionBounds(0f,0f).intersects(
    player.getGame().getPuck().getCollisionBounds(0f,0f))){
79              return true;
```

```
 80          }
 81          else {
 82              return false;
 83          }
 84      }
 85
 86      protected void move(){
 87
 88      }
 89
 90
 91      public void setTargetPositionRelative(float x, float
    y, Boolean home){
 92          if(home){
 93              this.targetX = x+595;
 94              this.targetY = y;
 95          }
 96          else {
 97              this.targetX = 595 - x;
 98              this.targetY = y;
 99          }
100      }
101
102      public void setTargetAbsolute(float x, float y){
103          this.targetX = x;
104          this.targetY = y;
105      }
106
107      public void setPositionRelative(float x, float y,
    Boolean home){
108          if(home){
109              this.x = x+595;
110              this.y = y;
111          }
112          else {
113              this.x = 595 - x;
114              this.y = y;
115          }
116      }
117
118      public float getX() {
119          return x;
120      }
121
122      public float getY() {
```

```
123         return y;
124     }
125
126     public abstract void tick();
127
128     public abstract void render(Graphics g);
129 }
130
```

```java
package Project.GUI.Entities.Player;

import Project.Base.Enums.Position;
import Project.Base.Enums.Team;
import Project.Base.Stats;

import static java.lang.Math.*;

/**
 * Created by Leon on 02/12/2018.
 */
public class Goalie extends Player {

    private Position position = Position.GOALIE;
    private GoalieStats stats;

    public Goalie(PlayerDetails player, GoalieStats stats, Team team, Boolean home) {
        super(player,team,home);
        this.stats = stats;
    }


    private void move(float angle) {
        int speed = ((stats.statSkating * stats.statEndurance) / currentEndurance) / 10;
        float accel = stats.statSkating * 0.00525f; //0.00625 used to get ~0.5 average based on stats found from Leo Culhane, McGill University 2012 lowered due to goalie
        float decel = 0.75f;

        float deltaX = targetX - x;
        float deltaY = targetY - y;
        float distance = (float) sqrt(Math.pow(deltaX, 2) + Math.pow(deltaY, 2));
        float decelDistance = (float) Math.pow(xVelocity,2) / (2 * decel);

        if (distance > decelDistance){//still accelerating if possible
            xVelocity = Math.min(xVelocity+accel,speed);
            yVelocity = Math.min(yVelocity+accel,speed);
        }
        else {
            xVelocity = Math.max(xVelocity - decel, 0);
```

```
39              yVelocity = Math.max(yVelocity - decel, 0);
40          }
41          float tx1 = (float) (x + xVelocity * cos(angle));
42          float tx2 = (float) (x + bounds.width + xVelocity
    * cos(angle));
43          float ty1 = (float) (y + yVelocity * sin(angle));
44          float ty2 = (float) (y + bounds.height + yVelocity
     * sin(angle));
45          if(getGame().getArena().legalMove(tx1,tx2,ty1,ty2)
    ) {
46              x += xVelocity * cos(angle);
47              y += yVelocity * sin(angle);
48          }
49          else {
50              int bounce = getGame().getArena().getBounce(x,
    y);
51              if (bounce == 1){
52                  yVelocity = -yVelocity;
53              }
54              else if (bounce == 2){
55                  xVelocity = -xVelocity;
56              }
57              else {
58                  xVelocity = -xVelocity;
59                  yVelocity = -yVelocity;
60              }
61
62          }
63      }
64
65      @Override
66      public void faceoff(float x, float y) {
67
68      }
69
70      @Override
71      public Stats getStats() {
72          return stats;
73      }
74
75      public String toString(){
76          String string = position.toString() + " - " +
    super.toString();
77          string += " SK:"+stats.getStatSkating()+" EN:"+
    stats.getStatEndurance()+" SI:"+stats.getStatSize()+" AG:"
```

```
77 +stats.getStatAgility()+" RB:"+stats.
   getStatReboundControl();
78         string += " SC"+stats.getStatStyleControl()+" HS
   :"+stats.getStatHandSpeed()+" PH:"+stats.
   getStatPuckHandling()+" PS:"+stats.getStatPenaltyShot();
79         return string;
80     }
81
82     @Override
83     public void tick() {
84         if((int)x!= (int)targetX || (int)y!=(int)targetY)
   {
85             float direction = getDirection();
86             move(direction);
87         }
88         updateBounds(x,y);
89     }
90 }
91
```

```java
package Project.GUI.Entities.Player;

import Project.Base.Enums.Position;
import Project.Base.Enums.Possession;
import Project.Base.Enums.Team;
import Project.Base.Game;
import Project.Base.Stats;
import Project.GUI.Assets.Assets;
import Project.GUI.Entities.Entity;
import Project.GUI.Entities.Player.PlayerStates.*;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.util.ArrayList;

/**
 * Created by Leon on 02/12/2018.
 */
public abstract class Player extends Entity implements PropertyChangeListener {
    private PlayerDetails player;

    private Boolean onIce = false;

    protected Team team;

    protected Boolean homeTeam;
    private int numberOffset;
    protected Boolean hasPuck = false;
    protected Possession lastTouch;

    protected int currentEndurance = 100;

    protected PlayerState playerState = null;

    public BufferedImage circle;
    private BufferedImage circleNumber;

    public Player(PlayerDetails player,Team team,Boolean home) {
        super(400,250,21,21);
        this.player = player;
        this.team = team;
```

```
44          this.homeTeam = home;
45          createCircle();
46      }
47
48      public void setX(int x){
49          this.x = x;
50      }
51      public void setY(int y){this.y = y;}
52
53      public void setHomeTeam(Boolean home){
54          this.homeTeam = home;
55      }
56
57      public void setCurrentPosition(float x, float y){
58          this.x = x;
59          this.y = y;
60      }
61      public void setTargetPositionAbsolute(float x, float y
    ){
62          this.targetX = x;
63          this.targetY = y;
64      }
65
66      public String getPlayerName(){
67          return player.name;
68      }
69
70      public Position getCurrentPosition(){
71          return currentPosition;
72      }
73
74      private void createCircle(){
75          if (homeTeam){
76              circle = Assets.homeTeam;
77              setCircleNumber(Assets.homeNumbers);
78          }
79          else {
80              circle = Assets.awayTeam;
81              setCircleNumber(Assets.awayNumbers);
82          }
83      }
84
85      private void setCircleNumber(ArrayList<BufferedImage>
    numbers){
86          if (player.number.length()==1){
```

```
 87             numberOffset = 7;
 88             circleNumber = numbers.get(Integer.parseInt(
    player.number));
 89         }
 90         else {
 91             numberOffset =5;
 92             String[] array = player.number.split("");
 93             circleNumber = new BufferedImage(11,9,
    BufferedImage.TRANSLUCENT);
 94             circleNumber.createGraphics().drawImage(
    numbers.get((Integer.parseInt(array[0]))),0,0,null);
 95             circleNumber.createGraphics().drawImage(
    Assets.numberSpace,5,0,null);
 96             circleNumber.createGraphics().drawImage(
    numbers.get((Integer.parseInt(array[1]))),6,0,null);
 97         }
 98     }
 99
100     public void setGame(Game game){
101         this.game = game;
102         System.out.println(this.game.toString());
103         this.game.getPuck().addPropertyChangeListener(
    this);
104     }
105
106
107     public ArrayList<Player> getPositions(boolean home){
108         ArrayList<Player> allPlayers = new ArrayList<>();
109         if(home){
110             allPlayers.addAll(game.getHomeTeam().
    getAllOnIce().values());
111         }
112         else {
113             allPlayers.addAll(game.getAwayTeam().
    getAllOnIce().values());
114         }
115         allPlayers.remove(this);
116         return allPlayers;
117     }
118
119     public abstract void faceoff(float x,float y);
120
121
122     protected void setPlayerState(PlayerState state){
123         this.playerState = state;
```

```
124     }
125 
126 
127     public void setLastTouch(Possession touch){
128         this.lastTouch = touch;
129     }
130 
131     //observer pattern listener
132     public void propertyChange(PropertyChangeEvent evt){
133         this.setLastTouch((Possession) evt.getNewValue())
    ;
134         this.updateState();
135     }
136 
137     private void updateState(){
138         if (this instanceof Skater) {
139             if(hasPuck){
140                 this.setPlayerState(new
    HasPuckPlayerState((Skater)this));
141             }
142             else {
143                 if (lastTouch == Possession.FACEOFF) {
144                     this.setPlayerState(new
    FaceoffPlayerState((Skater)this));
145                 } else if (lastTouch == Possession.HOME){
146                     if (homeTeam){
147                         this.setPlayerState(new
    AttackingPlayerState((Skater)this));
148                     }
149                     else {
150                         this.setPlayerState(new
    DefendingPlayerState(this));
151                     }
152                 } else if (lastTouch == Possession.AWAY)
    {
153                     if (homeTeam) {
154                         this.setPlayerState(new
    DefendingPlayerState(this));
155                     } else {
156                         this.setPlayerState(new
    AttackingPlayerState((Skater) this));
157                     }
158                 }
159                 else if (lastTouch == Possession.LOOSE){
160                         this.setPlayerState(new
```

```
160 DefaultPlayerState(this));
161                 }
162 
163             }
164         } else {
165             //is goalie
166             this.setPlayerState(new GoaliePlayerState((
    Goalie)this));
167         }
168     }
169 
170     public void setCurrentPlayingPosition(Position
    position){
171         this.currentPosition = position;
172     }
173 
174     protected float getDirection(){
175         var deltaX = targetX - x;
176         var deltaY = targetY - y;
177         var rad = Math.atan2(deltaY,deltaX);
178         return (float) rad;
179     }
180 
181     public abstract Stats getStats();
182 
183     public Game getGame() {
184         return game;
185     }
186 
187     public boolean isHomeTeam(){
188         return homeTeam;
189     }
190 
191     public int getCurrentEndurance(){
192         return currentEndurance;
193     }
194 
195     public void setHasPuck(boolean hasPuck){
196         this.hasPuck = hasPuck;
197     }
198 
199     public Boolean getHasPuck() {
200         return hasPuck;
201     }
202 
```

```
203     public void pass(Player target){
204
205     }
206
207     public void setFaceoffRoll(int faceoffRoll){
208
209     }
210     public int getFaceoffRoll(){return 0;}
211
212     public void setState(int state){
213        updateState();
214     }
215
216     @Override
217     public void tick() {
218         playerState.think();
219         playerState.act();
220     }
221
222     @Override
223     public void render(Graphics g) {
224         g.drawImage(circle,(int)x,(int)y,null);
225         g.drawImage(circleNumber,(int)x+numberOffset,(int
    )y+6,null);
226 //        g.setColor(Color.red);
227 //        g.fillRect(bounds.x,bounds.y,bounds.width,
    bounds.height);
228     }
229
230     public String toString(){
231         String string;
232         string = player.name + " #" + player.number +
    ": ";
233         return string;
234     }
235 }
```

```java
package Project.GUI.Entities.Player;

import Project.Base.Arena;
import Project.Base.Enums.Position;
import Project.Base.Enums.Possession;
import Project.Base.Enums.Team;
import Project.Base.Stats;
import Project.GUI.Entities.Player.PlayerStates.*;
import Project.States.FaceoffState;

import java.awt.*;

import static java.lang.Math.*;

/**
 * Created by Leon on 02/12/2018.
 */
public class Skater extends Player {

    private Position position;
    private SkaterStats stats;
    private int faceoffRoll = 0;

    private int hitTimer = 0;
    private int faceoffTimer = 0;
    private int speed;

    //PlayerStates
    DefaultPlayerState defaultPlayerState;
    HasPuckPlayerState hasPuckPlayerState;
    AttackingPlayerState attackingPlayerState;
    DefendingPlayerState defendingPlayerState;
    FaceoffPlayerState faceoffPlayerState;



    public Skater(PlayerDetails player, Position position,
   SkaterStats stats, Team team,Boolean home) {
        super(player,team,home);
        this.position = position;
        this.stats = stats;

        defaultPlayerState = new DefaultPlayerState(this);
        hasPuckPlayerState = new HasPuckPlayerState(this);
        attackingPlayerState = new AttackingPlayerState(
```

```
44 this);
45         defendingPlayerState = new DefendingPlayerState(
   this);
46         faceoffPlayerState = new FaceoffPlayerState(this);
47 
48         this.setPlayerState(defaultPlayerState);
49 
50 
51     }
52 
53     private void updateState(){
54         if(hasPuck){
55             this.setPlayerState(hasPuckPlayerState);
56         }
57         else {
58             if (lastTouch == Possession.FACEOFF) {
59                 this.setPlayerState(faceoffPlayerState);
60             } else if (lastTouch == Possession.HOME){
61                 if (homeTeam){
62                     this.setPlayerState(
   attackingPlayerState);
63                 }
64                 else {
65                     this.setPlayerState(
   defendingPlayerState);
66                 }
67             } else if (lastTouch == Possession.AWAY) {
68                 if (homeTeam) {
69                     this.setPlayerState(
   defendingPlayerState);
70                 } else {
71                     this.setPlayerState(
   attackingPlayerState);
72                 }
73             }
74             else if (lastTouch == Possession.LOOSE){
75                 this.setPlayerState(defaultPlayerState);
76             }
77         }
78     }
79 
80 
81     public void move(float angle) {
82         speed = ((stats.getStatSkating() * stats.
   getStatEndurance()) / currentEndurance) / 10;
```

```
 83          float accel = stats.getStatSkating() * 0.00625f;
    //0.00625 used to get ~0.5 average based on stats found
    from Leo Culhane, McGill University 2012
 84          float decel = 0.75f;
 85          //get distance to target
 86          float deltaX = targetX - x;
 87          float deltaY = targetY - y;
 88          float distance = (float) sqrt(Math.pow(deltaX, 2)
     + Math.pow(deltaY, 2));
 89          float decelDistance = (float) Math.pow(xVelocity,
    2) / (2 * decel);
 90
 91          if (distance > decelDistance){//still
    accelerating if possible
 92              xVelocity = Math.min(xVelocity+accel,speed);
 93              yVelocity = Math.min(yVelocity+accel,speed);
 94          }
 95          else {//deceleration
 96              xVelocity = Math.max(xVelocity - decel, 0);
 97              yVelocity = Math.max(yVelocity - decel, 0);
 98          }
 99          //movement
100          float xMove = (float) (xVelocity * cos(angle));
101          float yMove = (float) (yVelocity * sin(angle));
102          float tx1 = (float) (x + xVelocity * cos(angle));
103          float tx2 = (float) (x + bounds.width + xMove);
104          float ty1 = (float) (y + yVelocity * sin(angle));
105          float ty2 = (float) (y + bounds.height + yMove);
106          if(getGame().getArena().legalMove(tx1,tx2,ty1,ty2
    )) {
107              if (!(playerState instanceof
    DefaultPlayerState)&& !(playerState instanceof
    FaceoffPlayerState)) {
108                  if(!checkPlayerCollision(xMove,0f)){
109                      moveX(xMove);
110                  }
111                  else {
112                      playerState.hit(xMove,0f,xMove,yMove)
    ;
113                      //System.out.println(toString() +"
    HITx");
114                  }
115                  if(!checkPlayerCollision(0f,yMove)){
116                      moveY(yMove);
117                  }
```

```
118                 else{
119                     playerState.hit(0f,yMove,xMove,yMove)
    ;
120                     //System.out.println(toString() + "
    HITy");
121                 }
122             } else {
123                 moveX(xMove);
124                 moveY(yMove);
125             }
126         }
127         else {
128             xVelocity = 0;
129             yVelocity = 0;
130         }
131     }
132 
133     public int getSpeed() {
134         return speed;
135     }
136 
137     public void moveX(float xMove){
138         if(hitTimer>30) {
139             x += xMove;
140         }
141     }
142 
143     public void moveY(float yMove){
144         if(hitTimer>30) {
145             y += yMove;
146         }
147     }
148 
149     public void beenHit(){
150         xVelocity = 0;
151         yVelocity = 0;
152         resetHitTimer();
153         setHasPuck(false);
154     }
155 
156     public void pass(Player target){
157         float targetsX = target.getX();
158         float targetsY = target.getY();
159         int xnoise = getGame().random(125)-stats.
    getStatPassing();
```

```
160         int ynoise = getGame().random(125)-stats.
    getStatPassing();
161         if(getGame().random(1) == 0){
162             xnoise = -xnoise;
163         }
164         if(getGame().random(1)== 1){
165             ynoise = -ynoise;
166         }
167         float passTargetX = targetsX + xnoise;
168         float passTergetY = targetsY + ynoise;
169 
170         int passStrength = 0;
171         passStrength += stats.getStatStrength()*0.2;
172         passStrength += stats.getStatPassing()*0.5;
173         passStrength += game.random(50);
174         passStrength = passStrength/10;
175         System.out.println(passTargetX + " " +
    passTergetY + " " + passStrength);
176         game.getPuck().setTargetAbsolute(passTargetX,
    passTergetY);
177         game.getPuck().setAngeleCalculate();
178         game.getPuck().setSpeed((passStrength));
179         game.getPuck().resetPuckTimer();
180         game.getPuck().setLastTouch(Possession.LOOSE);
181         setHasPuck(false);
182         updateState();
183     }
184 
185     public void shoot(){
186         float targetsX;
187         float targetsY;
188 
189         if(isHomeTeam()){
190             targetsX = 154;
191             targetsY = 267;
192         }
193         else {
194             targetsX = 1060;
195             targetsY = 267;
196         }
197 
198         int xnoise = getGame().random(10)-stats.
    getStatScoring();
199         int ynoise = getGame().random(150)-stats.
    getStatScoring();
```

```
200         if(getGame().random(1) == 0){
201             xnoise = -xnoise;
202         }
203         if(getGame().random(1)== 1){
204             ynoise = -ynoise;
205         }
206         float shootTargetX = targetsX + xnoise;
207         float shootTergetY = targetsY + ynoise;
208 
209         int shotStrength = 0;
210         shotStrength += stats.getStatStrength()*0.2;
211         shotStrength += stats.getStatScoring()*0.5;
212         shotStrength += game.random(50);
213         shotStrength = shotStrength/7;
214         System.out.println(shootTargetX + " " +
    shootTergetY + " " + shotStrength);
215         game.getPuck().setTargetAbsolute(shootTargetX,
    shootTergetY);
216         game.getPuck().setAngeleCalculate();
217         game.getPuck().setSpeed((shotStrength));
218         game.getPuck().resetPuckTimer();
219         setHasPuck(false);
220         game.getPuck().setLastTouch(Possession.LOOSE);
221         updateState();
222     }
223 
224     public int getIdealOffenceZone(){
225         switch (currentPosition){
226             case CENTER:
227                 return 9;
228             case LWING:
229                 return 4;
230             case RWING:
231                 return 14;
232             case LDEFENCE:
233                 return 2;
234             case RDEFENCE:
235                 return 12;
236         }
237         return 0;
238     }
239 
240     public int getIdealDefenceZone(){
241         switch (currentPosition){
242             case CENTER:
```

```
243                 return 8;
244             case LWING:
245                 return 2;
246             case RWING:
247                 return 12;
248             case LDEFENCE:
249                 return 4;
250             case RDEFENCE:
251                 return 14;
252             case DEFENCE:
253                 return 4;
254         }
255         return 0;
256     }
257 
258     public int getFaceoffRoll() {
259         System.out.println(toString() + " Returning
    faceoff roll of " + faceoffRoll);
260         return faceoffRoll;
261     }
262 
263     public void setFaceoffRoll(int faceoffRoll) {
264         System.out.println(toString() + " Setting Faceoff
     Roll to " + faceoffRoll);
265         this.faceoffRoll = faceoffRoll;
266     }
267 
268     public SkaterStats getStats(){
269         return stats;
270     }
271 
272     public void resetHitTimer(){
273         hitTimer = 0;
274     }
275 
276     public int getHitTimer(){
277         return hitTimer;
278     }
279 
280     private void updateTimers(){
281         hitTimer ++;
282         faceoffTimer ++;
283         //System.out.println(hitTimer);
284         //System.out.println(faceoffTimer);
285     }
```

```
286
287     public void setHasPuck(){
288         if(isHomeTeam()){
289             getGame().getPuck().setLastTouch(Possession.
    HOME);
290         }
291         else {
292             getGame().getPuck().setLastTouch(Possession.
    AWAY);
293         }
294         hasPuck = true;
295         setPlayerState(hasPuckPlayerState);
296     }
297
298     public void faceoff(float x, float y){
299         setPlayerState(faceoffPlayerState);
300         setTargetPositionRelative(x,y,isHomeTeam());
301     }
302
303     public void faceoffGo(){
304         faceoffPlayerState.think();
305     }
306
307     public int getFaceoffTimer(){
308         return faceoffTimer;
309     }
310
311     public void resetFaceoffTimer(){
312         faceoffTimer = 0;
313     }
314
315     public String toString() {
316         String string = position.toString() + " - " +
    super.toString();
317         string += stats.toString();
318         string += playerState.toString();
319         string += "  ----  ";
320         string += targetX;
321         string += " ";
322         string += targetY;
323         return string;
324     }
325
326     public void tick() {
327         updateState();
```

```
328         System.out.println(toString());
329         super.tick();
330         //System.out.println(getPlayerName() +" x:"+ x
     +" y:" + y + " Tx:" + targetX + "Ty: "+targetY);
331 //        if((int)x!= (int)targetX || (int)y!=(int)
    targetY){
332 //            float direction = getDirection();
333 //            move(direction);
334 //        }
335         //System.out.println(toString());
336         move(getDirection());
337         updateBounds(x,y);
338         updateTimers();
339 
340     }
341 
342 
343 
344 }
345 
```

```java
package Project.GUI.Entities.Player;

import Project.Base.Stats;

/**
 * Created by Leon on 04/12/2018.
 */
public class GoalieStats extends Stats {
    protected int statSkating;
    protected int statEndurance;
    protected int statSize;
    protected int statAgility;
    protected int statReboundControl;
    protected int statStyleControl;
    protected int statHandSpeed;
    protected int statPuckHandling;
    protected int statPenaltyShot;

    public GoalieStats(int statSkating, int statEndurance, int statSize, int statAgility, int statReboundControl, int statStyleControl, int statHandSpeed, int statPuckHandling, int statPenaltyShot) {
        this.statSkating = statSkating;
        this.statEndurance = statEndurance;
        this.statSize = statSize;
        this.statAgility = statAgility;
        this.statReboundControl = statReboundControl;
        this.statStyleControl = statStyleControl;
        this.statHandSpeed = statHandSpeed;
        this.statPuckHandling = statPuckHandling;
        this.statPenaltyShot = statPenaltyShot;
    }

    public int getStatSkating() {
        return statSkating;
    }

    public int getStatEndurance() {
        return statEndurance;
    }

    public int getStatSize() {
        return statSize;
    }

```

```
43      public int getStatAgility() {
44          return statAgility;
45      }
46
47      public int getStatReboundControl() {
48          return statReboundControl;
49      }
50
51      public int getStatStyleControl() {
52          return statStyleControl;
53      }
54
55      public int getStatHandSpeed() {
56          return statHandSpeed;
57      }
58
59      public int getStatPuckHandling() {
60          return statPuckHandling;
61      }
62
63      public int getStatPenaltyShot() {
64          return statPenaltyShot;
65      }
66  }
67
```

```
 1 package Project.GUI.Entities.Player;
 2
 3 import Project.Base.Stats;
 4
 5 /**
 6  * Created by Leon on 04/12/2018.
 7  */
 8 public class SkaterStats extends Stats {
 9     private int statChecking;
10     private int statFighting;
11     private int statDiscipline;
12     private int statSkating;
13     private int statStrength;
14     private int statEndurance;
15     private int statPuckHandling;
16     private int statFaceOffs;
17     private int statPassing;
18     private int statScoring;
19     private int statDefence;
20     private int statPenaltyShot;
21
22     public SkaterStats(int statChecking, int statFighting,
    int statDiscipline, int statSkating, int statStrength,
   int statEndurance, int statPuckHandling, int statFaceOffs,
    int statPassing, int statScoring, int statDefence, int
   statPenaltyShot) {
23         this.statChecking = statChecking;
24         this.statFighting = statFighting;
25         this.statDiscipline = statDiscipline;
26         this.statSkating = statSkating;
27         this.statStrength = statStrength;
28         this.statEndurance = statEndurance;
29         this.statPuckHandling = statPuckHandling;
30         this.statFaceOffs = statFaceOffs;
31         this.statPassing = statPassing;
32         this.statScoring = statScoring;
33         this.statDefence = statDefence;
34         this.statPenaltyShot = statPenaltyShot;
35     }
36
37     public int getStatChecking() {
38         return statChecking;
39     }
40
41     public int getStatFighting() {
```

```
42          return statFighting;
43      }
44
45      public int getStatDiscipline() {
46          return statDiscipline;
47      }
48
49      public int getStatSkating() {
50          return statSkating;
51      }
52
53      public int getStatStrength() {
54          return statStrength;
55      }
56
57      public int getStatEndurance() {
58          return statEndurance;
59      }
60
61      public int getStatPuckHandling() {
62          return statPuckHandling;
63      }
64
65      public int getStatFaceOffs() {
66          return statFaceOffs;
67      }
68
69      public int getStatPassing() {
70          return statPassing;
71      }
72
73      public int getStatScoring() {
74          return statScoring;
75      }
76
77      public int getStatDefence() {
78          return statDefence;
79      }
80
81      public int getStatPenaltyShot() {
82          return statPenaltyShot;
83      }
84
85      public String toString(){
86          String string ="";
```

```
87          string += " SK:"+statChecking+" FI:"+statFighting
   +" DI:"+statDiscipline+ " SK:"+statSkating+" ST:"+
   statStrength+" EN:"+statEndurance;
88          string += " PH:"+statPuckHandling+" FO:"+
   statFaceOffs+" PA:"+statPassing+" SC:"+statScoring+" DF:"
   +statDefence+" PS:"+statPenaltyShot;
89          return string;
90      }
91  }
92
```

```java
package Project.GUI.Entities.Player;

/**
 * Created by Leon on 04/12/2018.
 */
public class PlayerDetails {
    protected String name;
    protected String number;
    protected int Height;
    protected int weight;
    protected String birthplace;

    public PlayerDetails(String Name, String number, int height, int weight, String birthplace) {
        this.name = Name;
        this.number = number;
        Height = height;
        this.weight = weight;
        this.birthplace = birthplace;
    }
}

```

```
 1 package Project.GUI.Entities.Player.PlayerStates;
 2
 3 public interface PlayerState {
 4     void hit(float xOffset,float yOffset,float xMove,
   float yMove);
 5
 6     void pass();
 7
 8     void think();
 9
10     void act();
11 }
12
```

```java
package Project.GUI.Entities.Player.PlayerStates;

import Project.GUI.Entities.Player.Player;

import java.util.ArrayList;

public abstract class SkaterState {
//    public ArrayList<Player> getPlayers(){
//
//    }


}

```

```java
package Project.GUI.Entities.Player.PlayerStates;

import Project.GUI.Entities.Player.Goalie;

public class GoaliePlayerState implements PlayerState {
    Goalie player;

    public GoaliePlayerState(Goalie player) {
        this.player = player;
    }

    @Override
    public void hit(float xOffset, float yOffset, float xMove, float yMove) {

    }

    @Override
    public void pass() {

    }

    @Override
    public void think() {

    }

    @Override
    public void act() {

    }
}

```

```java
package Project.GUI.Entities.Player.PlayerStates;

import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Player.Skater;
import Project.GUI.Entities.Player.SkaterStats;
import Project.GUI.Entities.Puck;

import java.awt.geom.Point2D;

import static java.lang.Math.sqrt;

public class DefaultPlayerState implements PlayerState {
    private Player player;
    private SkaterStats playerStats;

    public DefaultPlayerState(Player player) {
        this.player = player;
        this.playerStats = (SkaterStats) player.getStats()
;

    }

    /**
     * Calculates the point of interception for one object
 starting at point
     * <code>a</code> with speed vector <code>v</code> and
 another object
     * starting at point <code>b</code> with a speed of <
code>s</code>.
     *
     * @see <a
     *      href="http://jaran.de/goodbits/2011/07/17/
calculating-an-intercept-course-to-a-target-with-constant-
direction-and-velocity-in-a-2-dimensional-plane/">
Calculating
     *      an intercept course to a target with constant
direction and velocity
     *      (in a 2-dimensional plane)</a>
     *
     * @param a
     *            start vector of the object to be
intercepted
     * @param v
     *            speed vector of the object to be
intercepted
```

```
36      * @param b
37      *            start vector of the intercepting object
38      * @param s
39      *            speed of the intercepting object
40      * @return vector of interception or <code>null</code>
    if object cannot be
41      *         intercepted or calculation fails
42      *
43      * @author Jens Seiler
44      */
45     public static Point2D calculateInterceptionPoint(final
    Point2D a, final Point2D v, final Point2D b, final double
    s) {
46         final double ox = a.getX() - b.getX();
47         final double oy = a.getY() - b.getY();
48         //System.out.println(ox + " " + oy);
49 
50         final double h1 = v.getX() * v.getX() + v.getY() *
    v.getY() - s * s;
51         final double h2 = ox * v.getX() + oy * v.getY();
52         //System.out.println(h1 + " " + h2);
53         double t;
54         if (h1 == 0) { // problem collapses into a simple
   linear equation
55             t = -(ox * ox + oy * oy) / (2*h2);
56         } else { // solve the quadratic equation
57             final double minusPHalf = -h2 / h1;
58 
59             double discriminant = minusPHalf * minusPHalf
   - (ox * ox + oy * oy) / h1; // term in brackets is h3
60             if (discriminant < 0) { // no (real) solution
   then...
61                 System.out.println("Oops2");
62                 discriminant = 0.1;
63                 //return null;
64             }
65 
66             final double root = Math.sqrt(discriminant);
67 
68             final double t1 = minusPHalf + root;
69             final double t2 = minusPHalf - root;
70 
71             final double tMin = Math.min(t1, t2);
72             final double tMax = Math.max(t1, t2);
73 
```

```
74              t = tMin > 0 ? tMin : tMax; // get the
    smaller of the two times, unless it's negative
75              if (t < 0) { // we don't want a solution in
    the past
76                  t=1;
77                  System.out.println("Oops");
78                  //return null;
79              }
80          }
81
82          // calculate the point of interception using the
    found intercept time and return it
83          return new Point2D.Double(a.getX() + t * v.getX()
    , a.getY() + t * v.getY());
84      }
85
86      private float distanceToPuck(){
87          float puckX = player.getGame().getPuck().getX();
88          float puckY = player.getGame().getPuck().getY();
89
90          float deltaX = puckX - player.getX();
91          float deltaY = puckY - player.getY();
92          float distanceToPuck = (float) sqrt(Math.pow(
    deltaX, 2) + Math.pow(deltaY, 2));
93          return distanceToPuck;
94      }
95
96      private void moveTowardsPuck(){
97          float puckX = player.getGame().getPuck().getX();
98          float puckY = player.getGame().getPuck().getY();
99          float distanceToPuck = distanceToPuck();
100
101         float puckVelocityX = -(player.getGame().getPuck(
    ).getVelocityX());
102         float puckVelocityY = -(player.getGame().getPuck(
    ).getVelocityY());
103         //int ticksToPuck = (int) (distanceToPuck/Math.
    min(puckVelocityX,puckVelocityY));
104         //float predictedX = puckX+(puckVelocityX*
    ticksToPuck);
105         //float predictedY = puckY+(puckVelocityY*
    ticksToPuck);
106 //          Point2D puckPoint = new Point2D.Float(player.
    getGame().getPuck().getX(),player.getGame().getPuck().
    getY());
```

```
107 //         Point2D puckVelocity = new Point2D.Float(player
    .getGame().getPuck().getVelocityX(),player.getGame().
    getPuck().getVelocityY());
108 //         Point2D playerPoint = new Point2D.Float(player.
    getX(),player.getY());
109         double playerSpeed = ((Skater)player).getSpeed();
110 //         Point2D interceptPoint =
    calculateInterceptionPoint(puckPoint,puckVelocity,
    playerPoint,playerSpeed);
111 //         player.setTargetPositionAbsolute((float)
    interceptPoint.getX(),(float)interceptPoint.getY());
112         //player.setTargetPositionAbsolute(predictedX,
    predictedY);
113 
114         player.setTargetPositionAbsolute(puckX,puckY);
115 
116 //         float moveAngle = (float) Math.atan2(deltaY,
    deltaX);
117 //
118 //         if(interceptPoint != null){
119 //             player.setTargetPositionAbsolute((float)
    interceptPoint.getX(), (float) interceptPoint.getY());
120 //         }
121 //         else {
122 //             System.out.println("NULL");
123 //             ((Skater) player).move(moveAngle);
124 //         }
125     }
126 
127 //    private void contactWithPuck(){
128 //         if(player.playerPuckCollision(player)){
129 //             //collision with puck. Does player collect
    puck
130 //             //puck handling, skating, randomness, puck
    speed
131 //             int puckCollectionChance = 0;
132 //             puckCollectionChance += playerStats.
    getStatPuckHandling()*0.5; //50% PH
133 //             puckCollectionChance += playerStats.
    getStatSkating() *0.25; //20% SK
134 //             int puckSpeed = player.getGame().getPuck().
    getSpeed();
135 //             //check if collect puck
136 //             if (puckCollectionChance > player.getGame
    ().random(100+puckSpeed*5)){
```

```
137 //                  //collects puck
138 //                  ((Skater)player).setHasPuck();
139 //              }
140 //          }
141 //      }
142 
143     private void pickUpPuck(){
144         if (distanceToPuck()<15){
145             //pick up puck
146             System.out.println("GETTING PUCK");
147             ((Skater)player).setHasPuck();
148         }
149     }
150 
151 
152     @Override
153     public void hit(float xOffset, float yOffset, float
    xMove, float yMove) {
154 
155 
156     }
157 
158     @Override
159     public void pass() {
160 
161     }
162 
163     @Override
164     public void think() {
165         if (((Skater)player).getFaceoffTimer()>10) {
166             if (((Skater)player).getFaceoffTimer()>30) {
167                 moveTowardsPuck();
168             }
169             System.out.println(player.getGame().getPuck()
    .getPuckTimer());
170             if(player.getGame().getPuck().getPuckTimer()>
    10) {
171                 pickUpPuck();
172             }
173         }
174 
175     }
176 
177     @Override
178     public void act() {
```

```
179
180      }
181 }
182
```

```java
package Project.GUI.Entities.Player.PlayerStates;

import Project.Base.Enums.Position;
import Project.Base.Enums.Possession;
import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Player.Skater;
import Project.GUI.Entities.Player.SkaterStats;

import java.util.ArrayList;
import java.util.Random;
import java.util.concurrent.TimeUnit;

public class FaceoffPlayerState implements PlayerState{
    private Player player;
    private SkaterStats playerStats;
    private ArrayList<Player> teamPlayers = new ArrayList<>();
    private ArrayList<Player> defenders = new ArrayList<>();

    private int faceoffRoll;

    public FaceoffPlayerState(Skater player) {
        this.player = player;
        this.playerStats = player.getStats();
    }

    private void faceoffRoll(){
        faceoffRoll = playerStats.getStatFaceOffs()+player.getGame().random(50);
        player.setFaceoffRoll(faceoffRoll);
    }

    private void faceoff(){
        //get positions
        teamPlayers = player.getPositions(player.isHomeTeam());
        //get only defensemen
        for (Player p:teamPlayers){
            if (p.getCurrentPosition() == Position.LDEFENCE || p.getCurrentPosition() == Position.RDEFENCE || p.getCurrentPosition() == Position.DEFENCE ){
                defenders.add(p);
            }
        }
```

```
40
41          //pick player to pass to
42          Random rand = new Random();
43          int rint = rand.nextInt(defenders.size());
44          Player playerToPass = defenders.get(rint);
45
46          //Faceoff
47          System.out.println("FACEOFF GO");
48          player.getGame().getPuck().setPositionAbsolute(600
    ,255);
49          try {
50              TimeUnit.MILLISECONDS.sleep(100);
51          } catch (InterruptedException e) {
52              e.printStackTrace();
53          }
54          int oppFaceoffRoll = player.getGame().
    getOtherCenter(player.isHomeTeam()).getFaceoffRoll();
55          //System.out.println("Sleep!");
56          //TimeUnit.MILLISECONDS.sleep();
57          System.out.println(faceoffRoll + oppFaceoffRoll);
58          if(faceoffRoll>oppFaceoffRoll) {
59              System.out.println("Passing to " +playerToPass
    .toString() );
60              player.pass(playerToPass);
61              player.getGame().getPuck().setLastTouch(
    Possession.LOOSE);
62              player.setState(0);
63          }
64      }
65
66      @Override
67      public void hit(float xOffset, float yOffset, float
    xMove, float yMove) {
68
69      }
70
71
72      @Override
73      public void pass() {
74
75      }
76
77      @Override
78      public void think() {
79          if(player.getGame().getFaceoffTimer() > 225){
```

```
80              if (player.getGame().getFaceoffTimer()> 226)
   {
81                  if (player.getCurrentPosition()==Position
   .CENTER) {
82                      ((Skater)player).resetFaceoffTimer();
83                      faceoff();
84                  }
85              } else {
86                  faceoffRoll();
87              }
88
89          }
90      }
91
92      @Override
93      public void act() {
94
95      }
96  }
97
```

```java
package Project.GUI.Entities.Player.PlayerStates;

import Project.Base.Enums.Position;
import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Player.Skater;
import Project.GUI.Entities.Player.SkaterStats;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Random;

public class HasPuckPlayerState implements PlayerState {
    private Skater player;
    private SkaterStats playerStats;
    private boolean noAction = true;
    private Integer[] zoneWeight;
    private boolean pass;
    private boolean shoot;
    Player passTo = null;

    public HasPuckPlayerState(Skater player){
        this.player = player;
        this.playerStats = player.getStats();
    }

    private void skateWithPuck(){
        player.getGame().getPuck().setSpeed(0);
        player.getGame().getPuck().setPositionAbsolute((int)player.getX()+5,(int)player.getY()+5);
    }

    private boolean inOffensiveZone(){
        if (player.getX()<400){
            return true;
        }
        else return false;
    }
    private void moveTowardsOffensiveZone(){
        if (player.isHomeTeam()){
            player.setTargetPositionRelative(350,player.getY(),false);
        }
        else {
            player.setTargetPositionRelative(350,player.getY(),true);
```

```
43          }
44      }
45
46      @Override
47      public void hit(float xOffset, float yOffset, float
   xMove, float yMove) {
48          Player hittingPlayer = player.getHittingPlayer(
   player,player.getGame().getOpponentTeam(player.isHomeTeam(
   )),xOffset,yOffset);
49          //get own stat
50          //combination of skating, strength, puck handling
   , checking, and randomness
51          if (player.getHitTimer() < 30) {
52              int ownStat = 0;
53              ownStat += (playerStats.getStatSkating() * 0.
   35);
54              ownStat += (playerStats.getStatStrength() * 0.
   45);
55              ownStat += (playerStats.getStatPuckHandling()
   * 0.40);
56              ownStat += (playerStats.getStatChecking() * 0.
   15);
57              ownStat += (player.getCurrentEndurance() * 0.
   15);
58              ownStat += (player.getGame().random(50));
59
60              //get opponent stat
61              SkaterStats hittingPlayerStats = (SkaterStats)
    hittingPlayer.getStats();
62              int oppStat = 0;
63              oppStat += (hittingPlayerStats.getStatSkating(
   ) * 0.35);
64              oppStat += (hittingPlayerStats.getStatStrength
   () * 0.40);
65              oppStat += (hittingPlayerStats.
   getStatPuckHandling() * 0.15);
66              oppStat += (hittingPlayerStats.getStatChecking
   () * 0.45);
67              oppStat += (hittingPlayer.getCurrentEndurance(
   ) * 0.15);
68              oppStat += (hittingPlayer.getGame().random(50)
   );
69
70              //determine winner
71              if (ownStat>oppStat){
```

```
72                  //keep puck
73                  player.moveX(xMove);
74                  player.moveY(yMove);
75                  player.resetHitTimer();
76              }
77              else {
78                  //lose puck
79                  player.beenHit();
80              }
81          }
82          else {
83              player.moveX(xMove);
84              player.moveY(yMove);
85          }
86      }
87
88      @Override
89      public void pass() {
90
91      }
92
93      @Override
94      public void think() {
95          if (!inOffensiveZone()) {
96              moveTowardsOffensiveZone();
97          }
98          //get distance to goal
99          int distanceToGoal = player.getGame().
    getGoalieDistance(player, player.isHomeTeam());
100
101         //get distance to closest opponent
102         int distanceToClosestPlayer = player.getGame().
    getClosestPlayerDistance(player);
103
104         //get position of teammate
105         ArrayList<Player> ownTeam = new ArrayList<>();
106         ownTeam = player.getGame().getOwnTeam(player.
    isHomeTeam());
107         Random rand = new Random();
108         HashMap<Player, Integer> passablePlayers = new
    HashMap<>();
109         for (Player p : ownTeam) {
110             if (p.equals(player)) {
111                 continue;
112             }
```

```
113             if (p.getCurrentPosition() == Position.GOALIE
    ) {
114                 continue;
115             }
116             passablePlayers.put(p, player.getGame().
    getDistance(player, p));
117         }
118         int noPassable = passablePlayers.size();
119         int getPasable = rand.nextInt(noPassable);
120         int i = 0;
121         for (Player p:passablePlayers.keySet()){
122             if(i == getPasable){
123                 passTo = p;
124                 break;
125             }
126             i++;
127         }
128         // check not passing to opponent
129 //        HashMap<Player,Integer> clearPasssablePlayers=
    new HashMap<>();
130 //        clearPasssablePlayers.putAll(passablePlayers);
131 //        for (Player p:clearPasssablePlayers.keySet()){
132 //    }
133
134         //get clear shot to goal
135
136         //get zone preference
137         zoneWeight = player.getGame().getArena().
    getOffensiveZoneWeight(player.getGame().getArena().
    getZone(player.getX(),player.getY()));
138
139         //move towards goal, pass, shoot
140
141         int passChance;
142         int shootChance;
143         int skateChance;
144         int PShAttributeDiff = Math.abs(playerStats.
    getStatPassing()-playerStats.getStatScoring());
145         passChance = playerStats.getStatPassing()+
    zoneWeight[0]+player.getGame().random(100);
146         shootChance = playerStats.getStatScoring()+
    zoneWeight[1]+player.getGame().random(50);
147         skateChance = playerStats.getStatSkating()+
    zoneWeight[2]+player.getGame().random(250)+(
    distanceToGoal/10);
```

```
148         if (skateChance< passChance || skateChance<
    shootChance){
149             if (passChance>shootChance){
150                 //pass
151                 pass=true;
152                 shoot=false;
153                 noAction=false;
154             }
155             else {
156                 //shoot
157                 pass=false;
158                 shoot=true;
159                 noAction=false;
160             }
161         }
162         else {
163             //skate
164             pass=false;
165             shoot=false;
166             noAction=true;
167 
168         }
169         //skate
170 
171 
172     }
173 
174     @Override
175     public void act() {
176         if(noAction){
177             System.out.println("SKATING");
178             skateWithPuck();
179         }
180         else if (pass){
181             System.out.println("PASSING");
182            player.pass(passTo);
183         }
184         else if (shoot){
185             System.out.println("SHOOTING");
186             player.shoot();
187         }
188     }
189 }
190 
```

```java
package Project.GUI.Entities.Player.PlayerStates;

import Project.Base.*;
import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Player.Skater;
import Project.GUI.Entities.Player.SkaterStats;

import static java.lang.Math.sqrt;

public class AttackingPlayerState implements PlayerState {
    private Skater player;
    private SkaterStats playerStats;
    private boolean canBeOffside;
    boolean closestToPuck = false;
    boolean goForPuck = false;
    boolean moveTowardsZone = true;
    int idealZone;

    public AttackingPlayerState(Skater player) {
        this.player = player;
        this.playerStats = player.getStats();
    }

    @Override
    public void hit(float xOffset, float yOffset,float xMove,float yMove){
        Player hittingPlayer = player.getHittingPlayer(player,player.getGame().getOpponentTeam(player.isHomeTeam()),xOffset,yOffset);
        //get own stat
        //combination of skating, strength, puck handling, checking, and randomness
        if (player.getHitTimer() < 30) {
            int ownStat = 0;
            ownStat += (playerStats.getStatSkating() * 0.35);
            ownStat += (playerStats.getStatStrength() * 0.45);
            ownStat += (playerStats.getStatPuckHandling() * 0.40);
            ownStat += (playerStats.getStatChecking() * 0.15);
            ownStat += (player.getCurrentEndurance() * 0.15);
            ownStat += (player.getGame().random(50));
```

```
37
38              //get opponent stat
39              SkaterStats hittingPlayerStats = (SkaterStats)
    hittingPlayer.getStats();
40              int oppStat = 0;
41              oppStat += (hittingPlayerStats.getStatSkating(
   ) * 0.35);
42              oppStat += (hittingPlayerStats.getStatStrength
   () * 0.40);
43              oppStat += (hittingPlayerStats.
   getStatPuckHandling() * 0.15);
44              oppStat += (hittingPlayerStats.getStatChecking
   () * 0.45);
45              oppStat += (hittingPlayer.getCurrentEndurance(
   ) * 0.15);
46              oppStat += (hittingPlayer.getGame().random(50)
   );
47
48              //determine winner
49              if (ownStat>oppStat){
50                  //keep puck
51                  player.moveX(xMove);
52                  player.moveY(yMove);
53                  player.resetHitTimer();
54              }
55              else {
56                  //lose puck
57                  player.beenHit();
58              }
59          }
60          else {
61              player.moveX(xMove);
62              player.moveY(yMove);
63          }
64      }
65
66      private float[] getIdealZonePossible(){
67          float[] targetPosition;
68          if(canBeOffside){
69              //ensure won't move into offside position
70              targetPosition = player.getGame().getArena().
   getZonePosition(((player.getIdealOffenceZone()-1) %5)));
71          }
72          else {
73              System.out.println("ELSE");
```

```
74              targetPosition = player.getGame().getArena().
   getZonePosition(player.getIdealOffenceZone());
75          }
76          return targetPosition;
77
78      }
79
80      private void moveTowardsIdealZone(){
81          float[] targetPosition = getIdealZonePossible();
82          player.setTargetPositionRelative(targetPosition[0
   ],targetPosition[1],!player.isHomeTeam());
83      }
84
85      private void moveTowardsPuck(){
86          player.setTargetPositionAbsolute(player.getGame()
   .getPuck().getX(),player.getGame().getPuck().getY());
87      }
88
89      private void contactWithPuck(){
90          if(player.playerPuckCollision(player)){
91              //collision with puck. Does player collect
   puck
92              //puck handling, skating, randomness, puck
   speed
93              int puckCollectionChance = 0;
94              puckCollectionChance += playerStats.
   getStatPuckHandling()*0.5; //50% PH
95              puckCollectionChance += playerStats.
   getStatSkating() *0.25; //20% SK
96              int puckSpeed = player.getGame().getPuck().
   getSpeed();
97              //check if collect puck
98              if (puckCollectionChance > player.getGame().
   random(100+puckSpeed*5)){
99                  //collects puck
100                 player.setHasPuck();
101             }
102         }
103     }
104
105     @Override
106     public void pass() {
107
108     }
109
```

```
110     @Override
111     public void think() {
112         //does someone have the puck
113         boolean puckPossessed = player.getGame().
    puckPossessed();
114 
115         //how close to the puck are we
116         if (!puckPossessed) {
117             closestToPuck = player.getGame().
    isClosestToPuck(player);
118             int goForPuckChance = 20;
119             if (closestToPuck) {
120                 goForPuckChance += 50;
121             }
122             goForPuckChance += (player.
    getCurrentEndurance() / 10);
123             goForPuckChance += (playerStats.
    getStatPuckHandling() * 0.1);
124             if (goForPuckChance > player.getGame().random
    (100)) {
125                 goForPuck = true;
126                 moveTowardsZone = false;
127             } else {
128                 goForPuck = false;
129                 moveTowardsZone = true;
130             }
131         }
132         else {
133             moveTowardsZone = true;
134         }
135 
136         //is puck in offensive zone yet (offside?)
137         if (player.isHomeTeam()) {
138             if (player.getGame().getPuck().getZone() == 1
    ) {
139                 //in offensive zone
140                 canBeOffside = false;
141             } else {
142                 canBeOffside = true;
143             }
144         } else {
145             if (player.getGame().getPuck().getZone() == 3
    ) {
146                 //in offensive zone
147                 canBeOffside = false;
```

```
148             } else {
149                 canBeOffside = true;
150             }
151         }
152 
153         //go for puck or move to better position
154 
155     }
156 
157     private float distanceToPuck(){
158         float puckX = player.getGame().getPuck().getX();
159         float puckY = player.getGame().getPuck().getY();
160 
161         float deltaX = puckX - player.getX();
162         float deltaY = puckY - player.getY();
163         float distanceToPuck = (float) sqrt(Math.pow(
    deltaX, 2) + Math.pow(deltaY, 2));
164         return distanceToPuck;
165     }
166 
167     private void pickUpPuck(){
168         if (distanceToPuck()<15&&player.getGame().getPuck
    ().getPuckTimer()>10){
169             //pick up puck
170             ((Skater)player).setHasPuck();
171         }
172     }
173 
174 
175 
176     @Override
177     public void act() {
178         if(goForPuck){
179             moveTowardsPuck();
180         }
181         else if (moveTowardsZone){
182             moveTowardsIdealZone();
183         }
184         pickUpPuck();
185         //contactWithPuck();
186     }
187 }
```

```java
package Project.GUI.Entities.Player.PlayerStates;

import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Player.Skater;
import Project.GUI.Entities.Player.SkaterStats;

import static java.lang.Math.sqrt;

public class DefendingPlayerState implements PlayerState {
    private Skater player;
    private SkaterStats playerStats;

    public DefendingPlayerState(Player player) {
        this.player = (Skater)player;
        this.playerStats = (SkaterStats)player.getStats();

    }

    private void contactWithPuck(){
        if(player.playerPuckCollision(player)){
            //collision with puck. Does player collect puck
            //puck handling, skating, randomness, puck speed
            int puckCollectionChance = 0;
            puckCollectionChance += playerStats.getStatPuckHandling()*0.5; //50% PH
            puckCollectionChance += playerStats.getStatSkating() *0.25; //20% SK
            int puckSpeed = player.getGame().getPuck().getSpeed();
            //check if collect puck
            if (puckCollectionChance > player.getGame().random(100+puckSpeed*5)){
                //collects puck
                ((Skater)player).setHasPuck();
            }
        }
    }

    private void moveTowardsPuck(){
        player.setTargetPositionAbsolute(player.getGame().getPuck().getX(),player.getGame().getPuck().getY());
    }

```

```
39      private void moveTowardsIdealZone(){
40          float[] targetPosition;
41          targetPosition = player.getGame().getArena().
    getZonePosition(player.getIdealDefenceZone());
42          player.setTargetPositionRelative(targetPosition[0]
    ,targetPosition[1],player.isHomeTeam());
43      }
44
45      private float distanceToPuck(){
46          float puckX = player.getGame().getPuck().getX();
47          float puckY = player.getGame().getPuck().getY();
48
49          float deltaX = puckX - player.getX();
50          float deltaY = puckY - player.getY();
51          float distanceToPuck = (float) sqrt(Math.pow(
    deltaX, 2) + Math.pow(deltaY, 2));
52          return distanceToPuck;
53      }
54
55      private void pickUpPuck(){
56          if (distanceToPuck()<15&&player.getGame().getPuck(
    ).getPuckTimer()>10){
57              System.out.println("GETTING PUCK");
58              System.out.println(distanceToPuck());
59              //pick up puck
60              (player).setHasPuck();
61          }
62      }
63
64      @Override
65      public void hit(float xOffset, float yOffset, float
    xMove, float yMove) {
66
67      }
68
69      @Override
70      public void pass() {
71
72      }
73
74      @Override
75      public void think() {
76          System.out.println("Tick");
77          //contactWithPuck();
78          pickUpPuck();
```

```
79
80      }
81
82      @Override
83      public void act() {
84          moveTowardsIdealZone();
85      }
86  }
87
```

```java
package Project.Base;

import Project.Base.Enums.Line;
import Project.Base.Enums.Position;
import Project.Base.Enums.Possession;
import Project.Base.Enums.Team;
import Project.GUI.Assets.Assets;
import Project.GUI.Entities.Entity;
import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Player.Skater;
import Project.GUI.Entities.Puck;

import java.util.*;
import java.sql.Connection;

import static java.lang.Math.sqrt;

/**
 * Created by Leon on 16/01/2019.
 */
public class Game {
    //Fields
    private TeamObject homeTeam;
    private TeamObject awayTeam;
    private Arena arena = Arena.getArena();
    private Puck puck;
    private int tickTimer = 0;
    private int homeScore = 0;
    private int awayScore = 0;
    private int faceoffTimer = 0;
    private Random rand = new Random();


    public Game(Team home, Team away) {
        Assets.loadTeamAssets(home, away);
        puck = new Puck(this);
        homeTeam = new TeamObject(home, true);
        awayTeam = new TeamObject(away, false);
        setInitialPosition(homeTeam.getAllOnIce(), true);
        setInitialPosition(awayTeam.getAllOnIce(), false);
        for (Player player : homeTeam.getPlayerList().
  values()) {
            player.setGame(this);
        }
        for (Player player : awayTeam.getPlayerList().
```

```
44 values()) {
45             player.setGame(this);
46         }
47 
48         //puck.setSpeed(10);
49     }
50 
51     public int random(int maxRandom){
52         boolean useRandom = true;
53         if(useRandom){
54             return rand.nextInt(maxRandom+1);
55         }
56         else {
57             return maxRandom;
58         }
59     }
60 
61 
62     public static void setInitialPosition(HashMap<String,
   Player> team, Boolean home) {
63         float[] position;
64         float i = 0;
65         for (Player player : team.values()) {
66             position = Positions.getCenterFaceoff(player);
67             player.setCurrentPosition(600 - (25 - (10 * i)
   ), 100);
68             player.setTargetPositionRelative(position[0],
   position[1], home);
69         }
70     }
71 
72     public ArrayList<Player> getAllPlayers(){
73         ArrayList<Player> playerArrayList = new ArrayList<
   >();
74         playerArrayList.addAll(homeTeam.getAllOnIce().
   values());
75         playerArrayList.addAll(awayTeam.getAllOnIce().
   values());
76         return playerArrayList;
77     }
78 
79     public ArrayList<Player> getOpponentTeam(boolean
   isHome){
80         ArrayList<Player> playerArrayList = new ArrayList<
   >();
```

```
 81          if(isHome){
 82              playerArrayList.addAll(getAwayTeam().
    getAllOnIce().values());
 83          }
 84          else {
 85              playerArrayList.addAll(getHomeTeam().
    getAllOnIce().values());
 86          }
 87          return playerArrayList;
 88      }
 89
 90      public ArrayList<Player> getOwnTeam(boolean isHome){
 91          ArrayList<Player> playerArrayList = new ArrayList
    <>();
 92          if(isHome){
 93              playerArrayList.addAll(getHomeTeam().
    getAllOnIce().values());
 94          }
 95          else {
 96              playerArrayList.addAll(getAwayTeam().
    getAllOnIce().values());
 97          }
 98          return playerArrayList;
 99      }
100
101      private boolean loopTeamPossession(TeamObject team){
102
103          for (Player p:team.getAllOnIce().values()){
104              if (p.getHasPuck()){
105                  return true;
106              }
107          }
108          return false;
109      }
110
111      public boolean puckPossessed(){
112          if(puck.getLastTouch() != Possession.FACEOFF){
113              if (puck.getLastTouch() == Possession.HOME){
114                  return loopTeamPossession(homeTeam);
115              }
116              else {
117                  return loopTeamPossession(awayTeam);
118              }
119          }
120          return false;
```

```
121     }
122
123     private float getDistanceToPuck(Player player){
124         float deltaX = puck.getX() - player.getX();
125         float deltaY = puck.getY() - player.getY();
126         float distance = (float) sqrt(Math.pow(deltaX, 2)
     + Math.pow(deltaY, 2));
127         return distance;
128     }
129
130     public boolean isClosestToPuck(Player player){
131         float ownDistance = getDistanceToPuck(player);
132         ArrayList<Player> playerlist = new ArrayList<>();
133         if(player.isHomeTeam()){
134             playerlist.addAll(homeTeam.getAllOnIce().
    values());
135         }
136         else {
137             playerlist.addAll(awayTeam.getAllOnIce().
    values());
138         }
139         for (Player p:playerlist){
140             if(p != player){
141                 if (getDistanceToPuck(p)> ownDistance){
142                     return false;
143                 }
144             }
145         }
146         return true;
147     }
148
149     public TeamObject getHomeTeam() {
150         return homeTeam;
151     }
152
153     public TeamObject getAwayTeam() {
154         return awayTeam;
155     }
156
157     public ArrayList<Player> getBothTeamOnIce(){
158         ArrayList<Player> allOnIce = new ArrayList<>();
159         allOnIce.addAll(homeTeam.getAllOnIce().values());
160         allOnIce.addAll(awayTeam.getAllOnIce().values());
161         return allOnIce;
162     }
```

```
163
164     public String listAllPlayers() {
165         StringBuilder playerList = new StringBuilder("
    Home Team: " + homeTeam.getTeam().name() + "\n");
166         Iterator<Map.Entry<String, Player>> it1 =
    homeTeam.getPlayerList().entrySet().iterator();
167         while (it1.hasNext()) {
168             Map.Entry<String, Player> pair = it1.next();
169             playerList.append(pair.getValue().toString())
    .append("\n");
170         }
171         playerList.append("\nAway Team: ").append(
    awayTeam.getTeam().name()).append("\n");
172         Iterator<Map.Entry<String, Player>> it2 =
    awayTeam.getPlayerList().entrySet().iterator();
173         while (it2.hasNext()) {
174             Map.Entry<String, Player> pair = it2.next();
175             playerList.append(pair.getValue().toString())
    .append("\n");
176         }
177         return playerList.toString();
178     }
179
180     public Player getOtherCenter(boolean isHome){
181         if(isHome){
182             return getAwayTeam().getCenter();
183         }
184         else {
185             return getHomeTeam().getCenter();
186         }
187     }
188
189     public int getDistance(Player a, Player b){
190         float deltaX = a.getX() - b.getX();
191         float deltaY = a.getY() - b.getY();
192         return (int)(sqrt(Math.pow(deltaX, 2) + Math.pow(
    deltaY, 2)));
193     }
194
195     public int getGoalieDistance(Player a,boolean
    isHomeTeam){
196         for(Player p:getOpponentTeam(isHomeTeam)){
197             if (p.getCurrentPosition() == Position.GOALIE
    ) {
198                 return getDistance(a,p);
```

```
199             }
200         }
201         return 600;
202     }
203 
204     public int getClosestPlayerDistance(Player a){
205         int min = 1000;
206         for(Player p:getOpponentTeam(a.isHomeTeam())){
207             if(a.equals(p)){
208                 continue;
209             }
210             else {
211                 if(getDistance(a,p)<min){
212                     min = getDistance(a,p);
213                 }
214             }
215         }
216         return min;
217     }
218 
219     public int getFaceoffTimer(){
220         return faceoffTimer;
221     }
222 
223     public void tickFaceoffTimer(){
224         faceoffTimer ++;
225     }
226 
227     public Puck getPuck() {
228         return puck;
229     }
230 
231     public Arena getArena() {
232         return arena;
233     }
234 }
235 
```

```java
package Project.Base;

import Project.GUI.Entities.Entity;

import java.util.HashMap;

/**
 * Created by Leon on 16/01/2019.
 */
public class Arena {
    private static Arena instance = null; //Class uses singleton pattern to ensure only one arena instance exists
    private static HashMap<Integer,float[]> zonePositions;
    static {
        zonePositions = new HashMap<>();
        zonePositions.put(1,new float[]{90f,107f});
        zonePositions.put(2,new float[]{190f,107f});
        zonePositions.put(3,new float[]{290f,107f});
        zonePositions.put(4,new float[]{390f,107f});
        zonePositions.put(5,new float[]{490f,107f});
        zonePositions.put(6,new float[]{90f,214f});
        zonePositions.put(7,new float[]{190f,214f});
        zonePositions.put(8,new float[]{290f,214f});
        zonePositions.put(9,new float[]{390f,214f});
        zonePositions.put(10,new float[]{490f,214f});
        zonePositions.put(11,new float[]{90f,321f});
        zonePositions.put(12,new float[]{190f,321f});
        zonePositions.put(13,new float[]{290f,321f});
        zonePositions.put(14,new float[]{390f,321f});
        zonePositions.put(15,new float[]{490f,321f});
    }

    private static HashMap<Integer,Integer[]> offZoneWeights;
    //Passing, Shooting, Skating = 100.
    static {
        offZoneWeights = new HashMap<>();
        offZoneWeights.put(1,new Integer[]{25,5,70});
        offZoneWeights.put(2,new Integer[]{30,30,40});
        offZoneWeights.put(3,new Integer[]{35,45,20});
        offZoneWeights.put(4,new Integer[]{45,35,20});
        offZoneWeights.put(5,new Integer[]{70,5,25});
        offZoneWeights.put(6,new Integer[]{25,5,70});
        offZoneWeights.put(7,new Integer[]{20,50,30});
        offZoneWeights.put(8,new Integer[]{15,80,5});
```

```
44          offZoneWeights.put(9,new Integer[]{10,90,0});
45          offZoneWeights.put(10,new Integer[]{75,0,25});
46          offZoneWeights.put(11,new Integer[]{25,5,70});
47          offZoneWeights.put(12,new Integer[]{30,30,40});
48          offZoneWeights.put(13,new Integer[]{35,45,20});
49          offZoneWeights.put(14,new Integer[]{45,35,20});
50          offZoneWeights.put(15,new Integer[]{70,5,25});
51      }
52
53      //Fields
54      //boarders of rink
55      private static int xFullMin = 104;
56      private static int yFullMin =52;
57      private static int xFullMax = xFullMin + 1003;
58      private static int yFullMax = yFullMin + 428;
59
60      private static int xLimitMax = xFullMax - 64;
61      private static int xLimitMin = xFullMin + 64;
62
63      private static int yLimitMax = yFullMax - 64;
64      private static int yLimitMin = yFullMin + 64;
65
66
67      private Arena() {
68      }
69
70      public static boolean onIce(int x,int y){
71          //calculate x
72          if(x<xFullMin || x>xFullMax ){//outside full rink
    on x
73              //System.out.println(x);
74              System.out.println("outside full rink on x");
75              return false;
76          }
77          if(y>yFullMax || y<yFullMin){//outside full rink
    on y
78              System.out.println("outside full rink on y");
79              return false;
80          }
81          //we know player inside full rink, check for
    corners
82          if(x<xLimitMin || x>xLimitMax){//could be in
    corners
83              if (y<yLimitMin || y>yLimitMax) {//in corner
84                  //calculate corner
```

```
 85                 if(x<xLimitMin && y<yLimitMin){//top left
 86                     if(Math.pow((x-xLimitMin),2) + Math.
    pow((y-yLimitMin),2) <=  62){
 87                         return true;
 88                     }
 89                     return false;
 90                 }
 91                 else if(x>xLimitMax && y<yLimitMin){//top
     Right
 92                     if(Math.pow((x-xLimitMax),2) + Math.
    pow((y-yLimitMin),2) <=  62){
 93                         return true;
 94                     }
 95                     return false;
 96                 }
 97                 else if(x<xLimitMin && y>yLimitMax){//
    Bottom left
 98                     if(Math.pow((x-xLimitMin),2) + Math.
    pow((y-yLimitMax),2) <=  62){
 99                         return true;
100                     }
101                     return false;
102                 }
103                 else if(x>xLimitMax && y>yLimitMax){//
    Bottom right
104                     if(Math.pow((x-xLimitMax),2) + Math.
    pow((y-yLimitMax),2) <=  62){
105                         return true;
106                     }
107                     return false;
108                 }
109             }
110         }
111         //not in corner
112         return true;
113     }
114     public int getHalf(boolean home,float x){
115         if(home){
116             if(x <= 605){
117                 return 1;
118             }
119             else return 2;
120         }
121         else{
122             if (x <= 605){
```

```
123                 return 2;
124             }
125             else return 2;
126         }
127     }
128 
129     private int getXZone(float x){
130         if(x<125){
131             return 1;
132         }
133         else if(x<240){
134             return 2;
135         }
136         else if(x<350){
137             return 3;
138         }
139         else if(x<455){
140             return 4;
141         }
142         else {
143             return 5;
144         }
145     }
146 
147     public int getZone(float x, float y){
148         x = Math.abs((x-104) - 605);
149         y = y-52;
150 
151         int xZone = getXZone(x);
152         if (y<160){//Zone 1-5
153             switch (xZone){
154                 case 1:
155                     return 1;
156                 case 2:
157                     return 2;
158                 case 3:
159                     return 3;
160                 case 4:
161                     return 4;
162                 case 5:
163                     return 5;
164             }
165         }
166         if(y<265){//Zone 6-10
167             switch (xZone){
```

```
168                 case 1:
169                     return 6;
170                 case 2:
171                     return 7;
172                 case 3:
173                     return 8;
174                 case 4:
175                     return 9;
176                 case 5:
177                     return 10;
178             }
179         }
180         else{//Zone 11-15
181             switch (xZone){
182                 case 1:
183                     return 11;
184                 case 2:
185                     return 12;
186                 case 3:
187                     return 13;
188                 case 4:
189                     return 14;
190                 case 5:
191                     return 15;
192             }
193         }
194         return 0;
195     }
196 
197     public Boolean legalMove(float tx1, float tx2, float
    ty1, float ty2){
198         if (onIce((int) tx1, (int)ty1)&& onIce((int) tx2,
     (int)ty1)&&onIce((int) tx1, (int)ty2)&&onIce((int) tx2,
    (int)ty2)){
199             return true;
200         }
201         else return false;
202     }
203 
204     /**
205      *
206      * @param x
207      * @param y
208      * @return 1: y bounce, 2: x bounce, 3: corner bounce
209      */
```

```
210     public int getBounce(float x,float y){
211         if(x<xLimitMin || x>xLimitMax) {//could be in
    corners
212             if (y < yLimitMin || y > yLimitMax) {//in
    corner
213                 return 3;
214             }
215             else {//x bounce
216                 return 2;
217             }
218         }
219         else return 1;
220     }
221 
222     public  float[] getZonePosition(int zone){
223         return zonePositions.get(zone);
224     }
225 
226     public int getThird(Entity entity){
227         if(entity.getX()<378){
228             return 1;
229         }
230         else if(entity.getX()< 628){
231             //neutral zone
232             return 2;
233         }
234         else{
235             return 3;
236         }
237     }
238 
239     public static Arena getArena() {
240         if(instance == null) {
241             instance = new Arena();
242         }
243         return instance;
244     }
245 
246     public Integer[] getOffensiveZoneWeight(int zone){
247         return offZoneWeights.get(zone);
248     }
249 
250 }
251 
```

```java
package Project.Base;

/**
 * Created by Leon on 21/01/2019.
 */
public abstract class Stats {
}

```

```java
package Project.Base;

import Project.GUI.Sim;

public class Handler {
    private Sim sim;

    public Handler(Sim sim){
        this.sim = sim;
    }

    public Sim getSim() {
        return sim;
    }

    public void setSim(Sim sim) {
        this.sim = sim;
    }

}

```

```
 1 package Project.Base;
 2
 3 import Project.Base.Enums.Line;
 4 import Project.Base.Enums.Position;
 5 import Project.Base.Enums.Team;
 6 import Project.GUI.Entities.Player.*;
 7
 8 import java.sql.*;
 9 import java.util.HashMap;
10
11 public class Database {
12     private static Connection conn = null;
13
14     public static void init() {
15
16         //setup database connection
17         try {
18             String url = "jdbc:mysql://127.0.0.1:3306/
   project";
19             String username = "Java";
20             String password = "";
21             conn = DriverManager.getConnection(url,
   username, password);//jdbc:mysql://
22             System.out.println("Connected");
23         } catch (Exception ex) {
24             System.out.println("SQLException: " + ex.
   getMessage());
25         }
26
27     }
28
29     private static ResultSet executeSQL(PreparedStatement
   stmt){
30         ResultSet rs = null;
31         try {
32             rs = stmt.executeQuery();
33         }
34         catch (SQLException ex){
35             // handle any errors
36             System.out.println("SQLException: " + ex.
   getMessage());
37             System.out.println("SQLState: " + ex.
   getSQLState());
38             System.out.println("VendorError: " + ex.
   getErrorCode());
```

```
39          }
40          return rs;
41      }
42
43      public static int[] getTeamPosition(Team team){
44          try {
45              PreparedStatement ps = conn.prepareStatement("
    SELECT XPos,YPos FROM Teams WHERE Team = ?");
46              ps.setString(1,team.name());
47              ResultSet rs = executeSQL(ps);
48              int xPos = 0;
49              int yPos = 0;
50              while (rs.next()) {
51                  xPos = rs.getInt("XPos")-1;
52                  yPos = rs.getInt("YPos")-1;
53              }
54              return new int[]{xPos,yPos};
55
56          } catch (SQLException e) {
57              e.printStackTrace();
58              return null;
59          }
60
61      }
62
63      public static HashMap<String, Player> loadTeam(Team
    load,boolean home){
64          HashMap<String, Player> playerList = new HashMap<>
    ();
65
66          try {
67              PreparedStatement pstm = conn.prepareStatement
    ("SELECT * FROM Players WHERE Team = ?");
68              pstm.setString(1,load.name());
69              ResultSet rs = executeSQL(pstm);
70
71              while (rs.next()) {
72                  //To Player Details object
73                  String name = rs.getString("Name");
74                  String number = rs.getString("Number");
75                  Team team = Team.valueOf(rs.getString("
    Team"));
76                  String country = rs.getString("Country");
77                  String rookie = rs.getString("Rookie");
78                  int height = rs.getInt("Height");
```

```
 79                  int weight = rs.getInt("Weight");
 80
 81                  PlayerDetails newPlayerDetails = new
    PlayerDetails(name, number, height, weight, country);
 82
 83                  int rawposition = rs.getInt("Position");
 84                  Position position = null;
 85                  switch (rawposition) {
 86                      case 1:
 87                          position = Position.CENTER;
 88                          break;
 89                      case 2:
 90                          position = Position.LWING;
 91                          break;
 92                      case 4:
 93                          position = Position.RWING;
 94                          break;
 95                      case 8:
 96                          position = Position.DEFENCE;
 97                          break;
 98                      case 16:
 99                          position = Position.GOALIE;
100                          break;
101                  }
102                  if (position == Position.GOALIE) {
103                      //get goalie stats
104                      int sk = rs.getInt("CK");
105                      int en = rs.getInt("FG");
106                      int si = rs.getInt("DI");
107                      int ag = rs.getInt("SK");
108                      int rc = rs.getInt("ST");
109                      int sc = rs.getInt("EN");
110                      int hs = rs.getInt("PH");
111                      int ph = rs.getInt("FO");
112                      int ps = rs.getInt("PA");
113                      //create player
114                      GoalieStats newStats = new
    GoalieStats(sk, en, si, ag, rc, sc, hs, ph, ps);
115                      Player newPlayer = new Goalie(
    newPlayerDetails, newStats,team,home);
116                      newPlayer.setHomeTeam(home);
117                      playerList.put(newPlayer.
    getPlayerName(), newPlayer);
118                  } else {
119                      //ger skater stats
```

```
120                     int ck = rs.getInt("CK");
121                     int fg = rs.getInt("FG");
122                     int di = rs.getInt("DI");
123                     int sk = rs.getInt("SK");
124                     int st = rs.getInt("ST");
125                     int en = rs.getInt("EN");
126                     int ph = rs.getInt("PH");
127                     int fo = rs.getInt("FO");
128                     int pa = rs.getInt("PA");
129                     int sc = rs.getInt("SC");
130                     int df = rs.getInt("DF");
131                     int ps = rs.getInt("PS");
132                     //create player
133                     SkaterStats newStats = new
    SkaterStats(ck, fg, di, sk, st, en, ph, fo, pa, sc, df,
    ps);
134                     Player newPlayer = new Skater(
    newPlayerDetails, position, newStats,team,home);
135                     newPlayer.setHomeTeam(home);
136                     playerList.put(newPlayer.
    getPlayerName(), newPlayer);
137                 }
138             }
139 
140         } catch (SQLException e) {
141             e.printStackTrace();
142         }
143         return playerList;
144     }
145 
146     public static int getTeamNumber(Team team){
147         try {
148             PreparedStatement pstm = conn.
    prepareStatement("SELECT TeamNo FROM Teams WHERE Team
     = ?");
149             pstm.setString(1,team.name());
150             ResultSet rs = executeSQL(pstm);
151             while (rs.next()){
152                 return rs.getInt("TeamNo");
153             }
154         } catch (SQLException e) {
155             e.printStackTrace();
156         }
157         return 0;
158     }
```

```
159
160     public static HashMap<String, Player> loadLines (
    HashMap<String, Player> TeamPlayers, Team team, Line line
    ){
161         try {
162             PreparedStatement ps = conn.prepareStatement(
    "SELECT * FROM `lines` WHERE Team = ? && Line = ?");
163             ps.setString(1,team.name());
164             ps.setString(2,line.name());
165             ResultSet rs = executeSQL(ps);
166             //System.out.println(rs.toString());
167             HashMap<String, Player> teamLine = new
    HashMap<>(5);
168             while (rs.next()) {
169                 String playerName = rs.getString("
    PlayerName");
170                 if (TeamPlayers.containsKey(playerName))
    {
171                     Player player = TeamPlayers.get(
    playerName);
172                     player.setCurrentPlayingPosition(
    Position.valueOf(rs.getString("Position")));
173                     teamLine.put(player.getPlayerName(),
    player);
174                 }
175             }
176             return teamLine;
177         } catch (SQLException e) {
178             e.printStackTrace();
179             return null;
180         }
181     }
182
183 }
184
185
186
```

```java
package Project.Base;

import Project.GUI.Entities.Player.Player;

import java.util.HashMap;
import java.util.Map;

public class Positions {

    public static float[] getCenterFaceoff(Player player){
        float[] position = new float[2];
        switch (player.getCurrentPosition()){
            case CENTER:
                position[0] = 10;
                position[1] = 255;
                break;
            case LWING:
                position[0] = 15;
                position[1] = 420;
                break;
            case RWING:
                position[0] = 15;
                position[1] = 85;
            case LDEFENCE:
                position[0] = 115;
                position[1] = 385;
                break;
            case RDEFENCE:
                position[0] = 115;
                position[1] = 120;
                break;
            case GOALIE:
                position[0] = 435;
                position[1] = 255;
        }
        return position;
    }

    public static float[] getFaceoffOffset(Player player){
        float[] position = new float[2];
        switch (player.getCurrentPosition()){
            case CENTER:
                position[0] = 10;
                position[1] = 0;
                break;
```

```
46              case LWING:
47                  position[0] = -80;
48                  position[1] = 5;
49                  break;
50              case RWING:
51                  position[0] = 80;
52                  position[1] = 5;
53              case LDEFENCE:
54                  position[0] = -115;
55                  position[1] = 95;
56                  break;
57              case RDEFENCE:
58                  position[0] = 115;
59                  position[1] = 95;
60                  break;
61          }
62          if (!player.isHomeTeam()){
63              position[0] = -position[0];
64              //position[1] = -position[1];
65          }
66          return position;
67      }
68
69      public static float[] puckCenterFaceoff(){
70          float[] position = new float[2];
71          position[0] = 0;
72          position[1] = 255;
73          return position;
74      }
75
76
77  }
78
```

```java
package Project.Base;

import Project.Base.Enums.Line;
import Project.Base.Enums.Position;
import Project.Base.Enums.Team;
import Project.GUI.Entities.Player.Goalie;
import Project.GUI.Entities.Player.Player;

import java.util.HashMap;

public class TeamObject {
    private final HashMap<String, Player> playerList;
    private HashMap<String, Player> currentFLine;
    private HashMap<String, Player> currentDLine;
    private Line fLine;
    private Line dLine;
    private HashMap<String, Player> currentGoalie;
    private final Boolean homeTeam;
    private final Team team;

    public TeamObject(Team team, Boolean homeTeam) {
        this.team = team;
        this.homeTeam = homeTeam;
        playerList = Database.loadTeam(team, homeTeam);
        setLines(Line.FORWARD1, Line.DEFENCE1);
        setCurrentGoalie(Line.GOALIE1);
    }


    private void loadlines() {
        currentFLine = Database.loadLines(playerList, team
, fLine);
        currentDLine = Database.loadLines(playerList, team
, dLine);
    }


    public HashMap<String, Player> getPlayerList() {
        return playerList;
    }

    public HashMap<String, Player> getCurrentFLine() {
        return currentFLine;
    }

```

```
44      public HashMap<String, Player> getCurrentDLine() {
45          return currentDLine;
46      }
47
48      public Player getCenter() {
49          for (Player p : playerList.values()) {
50              if (p.getCurrentPosition() == Position.CENTER)
    {
51                  return p;
52              }
53          }
54          return null;
55      }
56
57
58          public HashMap<String, Player> getAllOnIce () {
59              HashMap<String, Player> allmaps = new HashMap<
   >();
60              allmaps.putAll(currentFLine);
61              allmaps.putAll(currentDLine);
62              allmaps.putAll(currentGoalie);
63              return allmaps;
64          }
65
66          public Boolean getHomeTeam () {
67              return homeTeam;
68          }
69
70          public Team getTeam () {
71              return team;
72          }
73
74          public void setLines (Line fLine, Line dLine){
75              this.fLine = fLine;
76              this.dLine = dLine;
77              loadlines();
78          }
79
80          public void setCurrentGoalie (Line line){
81              currentGoalie = Database.loadLines(playerList,
    team, line);
82          }
83      }
84
85
```

```
 1 package Project.Base.Enums;
 2
 3 /**
 4  * Created by Leon on 19/01/2019.
 5  */
 6 public enum Line {
 7     FORWARD1,FORWARD2,FORWARD3,FORWARD4,DEFENCE1,DEFENCE2,
   DEFENCE3,DEFENCE4,PPFORWARD1,PPFORWARD2,PPDEFENCE1,
   PPDEFENCE2,FORWARD14V4,FORWARD24V4,
 8     DEFENCE14V4,DEFENCE24V4,PK3FORWARD1,PK3FORWARD2,
   PK3DEFENCE1,PK3DEFENCE2,GOALIE1,GOALIE2,PENALTYSHOTS,
   EXTRAFORWARDS,EXTRADEFENCE,
 9     LASTMINOFFENCESIVEFORWARDS,LASTMINOFFENSIVEDEFENCE,
   LASTMINDEFENCIVEFORWAD,LASTMINDEVENCIVEDEFENCE
10 }
11
```

```java
package Project.Base.Enums;

/**
 * Created by Leon on 04/12/2018.
 */
public enum Stat {
SChecking, SFighting,SDiscipline,SSkating,SStrength,
SEndurance,SPuckHandling,SFaceOffs,SPassing,SScoring,
SDefence,SPenaltyShot,
GSkating,GEndurance,GSize,GAgility,GReboundControl,
GStyleControl,GHandSpeed,GPuckHandling,GPenaltyShot
}

```

```java
package Project.Base.Enums;

/**
 * Created by Leon on 02/12/2018.
 */
public enum Team {
    BUF,HAM,MAN,MIN,TOR,WKP,CAL,EDM,LAP,SFP,SEA,TEX,WIN,ANC,COL,DET,HAL,KEL,MON,STL,VAN
}

```

```java
package Project.Base.Enums;

/**
 * Created by Leon on 02/12/2018.
 */
public enum Position {
    LWING ("LW"),
    CENTER ("C"),
    RWING ("RW"),
    DEFENCE("D"),
    LDEFENCE("D"),
    RDEFENCE("D"),
    GOALIE ("G"),
    BENCH ("Bench");

    private String value;

    Position(final String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
    public String toString() {
        return this.getValue();
    }
}

```

```java
package Project.Base.Enums;

public enum Possession {
    HOME,AWAY,FACEOFF,LOOSE
}

```

```java
package Project.States;

import Project.Base.Handler;
import Project.GUI.Assets.Assets;
import Project.GUI.Entities.Player.Player;

import java.awt.*;

public abstract class State {
    protected Handler handler;

    public State(Handler handler){
        this.handler = handler;
    }

    private static State currentState = null;

    public static void setState(State state){
        currentState = state;
    }

    public static State getState(){
        return currentState;
    }

    public abstract void setFaceoffDot(int dot);

    public void update(){
        for (Player player:handler.getSim().getGame().getAwayTeam().getAllOnIce().values()) {
            player.tick();
        }
        for (Player player:handler.getSim().getGame().getHomeTeam().getAllOnIce().values()){
            player.tick();
        }
        handler.getSim().getGame().getPuck().tick();
    }

    public  void render(Graphics g){
        g.drawImage(Assets.rink,0,0,null);
        for (Player player:handler.getSim().getGame().getAwayTeam().getAllOnIce().values()) {
            player.render(g);
        }
```

```
43          for (Player player:handler.getSim().getGame().
   getHomeTeam().getAllOnIce().values()){
44              player.render(g);
45          }
46          handler.getSim().getGame().getPuck().render(g);
47      }
48
49      public abstract void startGame();
50  }
51
```

```java
package Project.States;

import Project.Base.*;
import Project.GUI.Assets.Assets;
import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Puck;

import java.awt.*;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;


public class SimState extends State {
    private HashMap<String, Player> playerList;
    private Player player1;
    private Player player2;
    private Arena a;



    public SimState(Handler handler){
        super(handler);
        Assets.init();
        Database.init();

        a = Arena.getArena();
    }


    @Override
    public void startGame() {

    }

    @Override
    public void setFaceoffDot(int dot) {

    }

}

```

```java
package Project.States;

import Project.Base.Handler;

import java.awt.*;

public class MenuState extends State{
    public MenuState(Handler handler){
        super(handler);
    }

    @Override
    public void startGame() {

    }

    @Override
    public void setFaceoffDot(int dot) {

    }

    @Override
    public void update() {

    }

    @Override
    public void render(Graphics g) {

    }
}

```

```java
package Project.States;

import Project.Base.Enums.Position;
import Project.Base.Handler;
import Project.Base.Positions;
import Project.GUI.Assets.Assets;
import Project.GUI.Entities.Player.Player;
import Project.GUI.Entities.Player.Skater;
import Project.GUI.Entities.Puck;

import java.awt.*;

public class FaceoffState extends State {
    private int faceoffDot;
    private Puck puck;
    private int dotX;
    private int dotY;
    private int timer;

    public FaceoffState(Handler handler) {
        super(handler);
        puck = handler.getSim().getGame().getPuck();
    }

    public void setFaceoffDot(int dot){
        //start faceoff timer (250 ticks)
        System.out.println("I'm here");
        timer = 0;
        //hide puck
        puck.setPositionAbsolute(5,5);
        //puck.setSpeed(0);
        faceoffDot = dot;
        switch (dot){
            case 0:
                dotX = 501;
                dotY=268;
                break;
            case 1:
                dotX =256;
                dotY=153;
                break;
            case 2:
                dotX =512;
                dotY=153;
                break;
```

```
46              case 3:
47                  dotX =700;
48                  dotY=153;
49                  break;
50              case 4:
51                  dotX =956;
52                  dotY=153;
53                  break;
54              case 5:
55                  dotX =256;
56                  dotY=380;
57                  break;
58              case 6:
59                  dotX =512;
60                  dotY=380;
61                  break;
62              case 7:
63                  dotX =700;
64                  dotY=380;
65                  break;
66              case 8:
67                  dotX =956;
68                  dotY=380;
69                  break;
70              }
71              //puck.setPositionAbsolute(dotX,dotY);
72          for (Player p:handler.getSim().getGame().
   getBothTeamOnIce()){
73              if (faceoffDot == 0){
74                  p.faceoff(Positions.getCenterFaceoff(p)[0]
   ,Positions.getCenterFaceoff(p)[1]); //double call for no
   reason.. oh well
75              }
76              else{
77                  float[] offset = Positions.
   getFaceoffOffset(p);
78                  p.faceoff(dotX+offset[0],dotY+offset[1]);
79              }
80              //p.faceoff(dotX,dotY);
81          }
82      }
83
84      @Override
85      public void startGame() {
86
```

```
 87      }
 88
 89      @Override
 90      public void update() {
 91          super.update();
 92          System.out.println(timer);
 93
 94          handler.getSim().getGame().tickFaceoffTimer();
 95
 96          if(handler.getSim().getGame().getFaceoffTimer()>
    220){
 97              //puck.setPositionAbsolute(600,255);
 98              //puck.setSpeed(0);
 99              for(Player p:handler.getSim().getGame().
    getBothTeamOnIce()){
100                  if (p.getCurrentPosition() == Position.
    CENTER) {
101                  }
102              }
103              System.out.println(dotX + " " + dotY);
104              }
105          }
106 }
107
108
```

```
 1 package Project.States;
 2
 3 import Project.Base.Enums.Line;
 4 import Project.Base.Handler;
 5 import Project.Base.Positions;
 6 import Project.GUI.Assets.Assets;
 7 import Project.GUI.Entities.Player.Player;
 8
 9 import java.awt.*;
10 import java.util.HashMap;
11
12 public class GameStartState extends State {
13     private int ticks;
14     private State simState;
15
16     public GameStartState(Handler handler){
17         super(handler);
18         ticks = 0;
19     }
20
21     public void startGame(){
22         handler.getSim().getGame().getHomeTeam().setLines(
   Line.FORWARD1,Line.DEFENCE1);
23         handler.getSim().getGame().getHomeTeam().
   setCurrentGoalie(Line.GOALIE1);
24
25         handler.getSim().getGame().getAwayTeam().setLines(
   Line.FORWARD1,Line.DEFENCE1);
26         handler.getSim().getGame().getAwayTeam().
   setCurrentGoalie(Line.GOALIE1);
27
28         setInitialPosition(handler.getSim().getGame().
   getHomeTeam().getAllOnIce(),true);
29         setInitialPosition(handler.getSim().getGame().
   getAwayTeam().getAllOnIce(),false);
30
31         handler.getSim().setFaceoffState(0);
32     }
33
34     public static void setInitialPosition(HashMap<String,
   Player> team, Boolean home) {
35         float[] position;
36         float i = 0;
37         for(Player player: team.values()){
38             position = Positions.getCenterFaceoff(player);
```

```
39              if(home){
40                  player.setCurrentPosition((600-25-(5*i)),
   60);
41              }
42              else {
43                  player.setCurrentPosition((600+25+(15*i)),
   60);
44              }
45              player.setTargetPositionRelative(position[0],
   position[1],home);
46              i++;
47          }
48      }
49
50      @Override
51      public void setFaceoffDot(int dot) {
52
53      }
54
55      @Override
56      public void update() {
57  //          for (Player player:handler.getSim().getGame().
   getAwayTeam().getAllOnIce().values()) {
58  //              player.tick();
59  //          }
60  //          for (Player player:handler.getSim().getGame().
   getHomeTeam().getAllOnIce().values()){
61  //              player.tick();
62  //          }
63  //          handler.getSim().getGame().getPuck().tick();
64  //          ticks += 1;
65  //          if (ticks>=250){
66  //              float[] position = Positions.
   puckCenterFaceoff();
67  //              handler.getSim().getGame().getPuck().
   setPositionRelative(position[0],position[1],false);
68  //              State.setState(handler.getSim().simState);
69  //          }
70      }
71
72      @Override
73      public void render(Graphics g) {
74
75      }
76  }
```

77
78
79