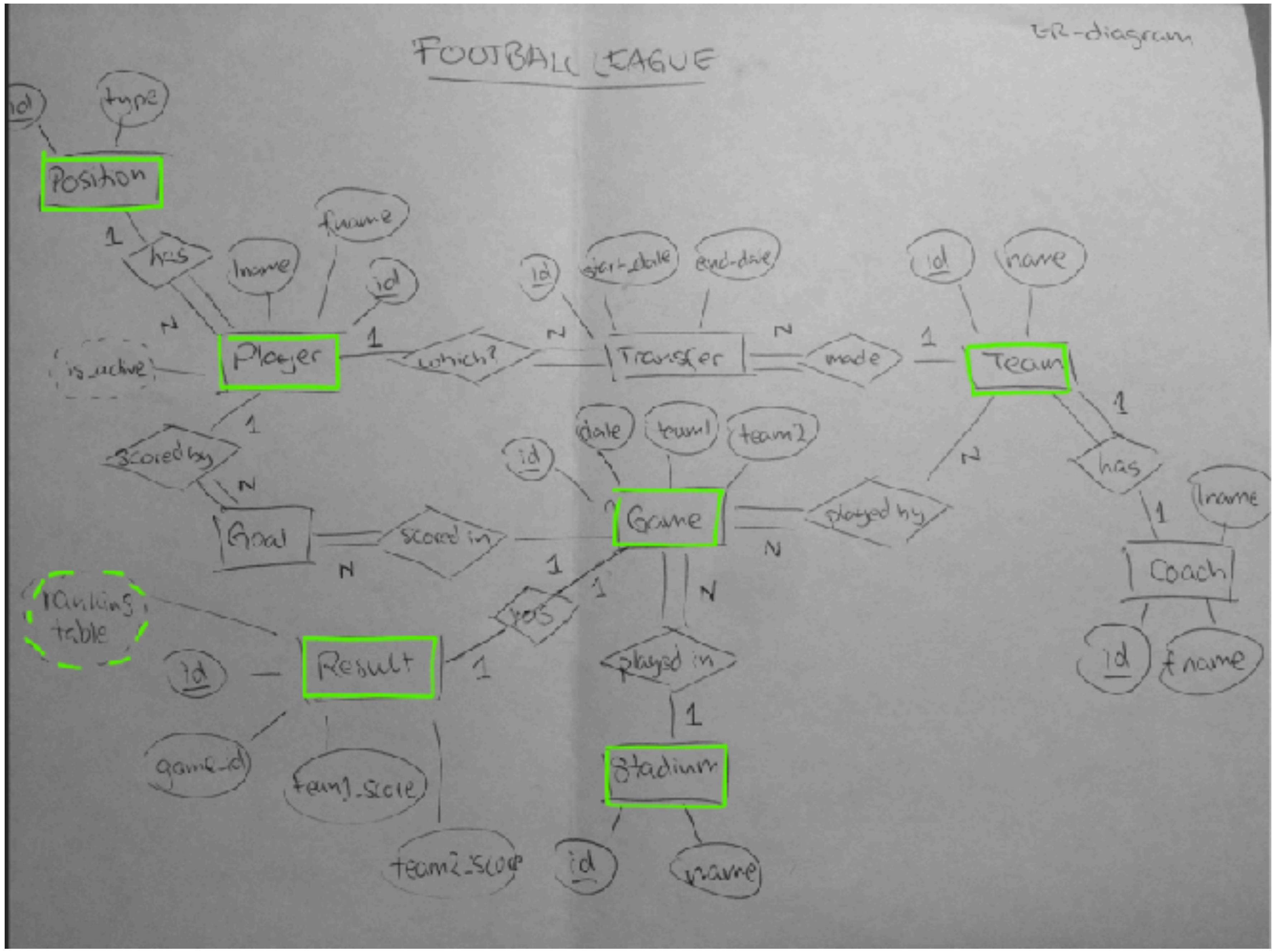
Football League

MySQL/Hibernate/JavaFX

ER-diagram



FOOTBALL LEAGUE
ER-diagram
id
type
Position
1
has
N
fname
lname
id
Player
is_active
1
which?
N
id
start_date
end_date
Transfer
N
made
1
id
name
Team
1
scored by
N
Goal
scored in
N
date
team1
team2
id
Game
played by
N
N
1
has
1
lname
Coach
id
fname
ranking table
id
Result
1
has
1
N
played in
1
game_id
team1_score
team2_score
Stadium
id
name

Uppfyller

Genomtänkt struktur (ER-diagram) ✓

Strukturerade klasser som motsvarar DB (ORM) ✓

Strukturerade kod ✓

Javadoc ✓

Grafiskt gränssnit (JavaFX) ✓

Data Flow
(pre user interaction)

Delete all results (ResultModel)	CRU**D**	✓
Queue all games (GameModel)	Datalogisk klass	✓
Create all results (ResultsModel)	**C**RUD	✓

User options

Show all games
(ResultModel.getAllGames =
Listners.addMenuLinkListener1)

C**R**UD ✓

Show all results
(ResultModel.getResultTeams()
= Listners.addMenuLinkListener2)

C**R**UD ✓

Update results
(ResultModel.updateScores()
=Listners.addUpdateButtonListeners()
)

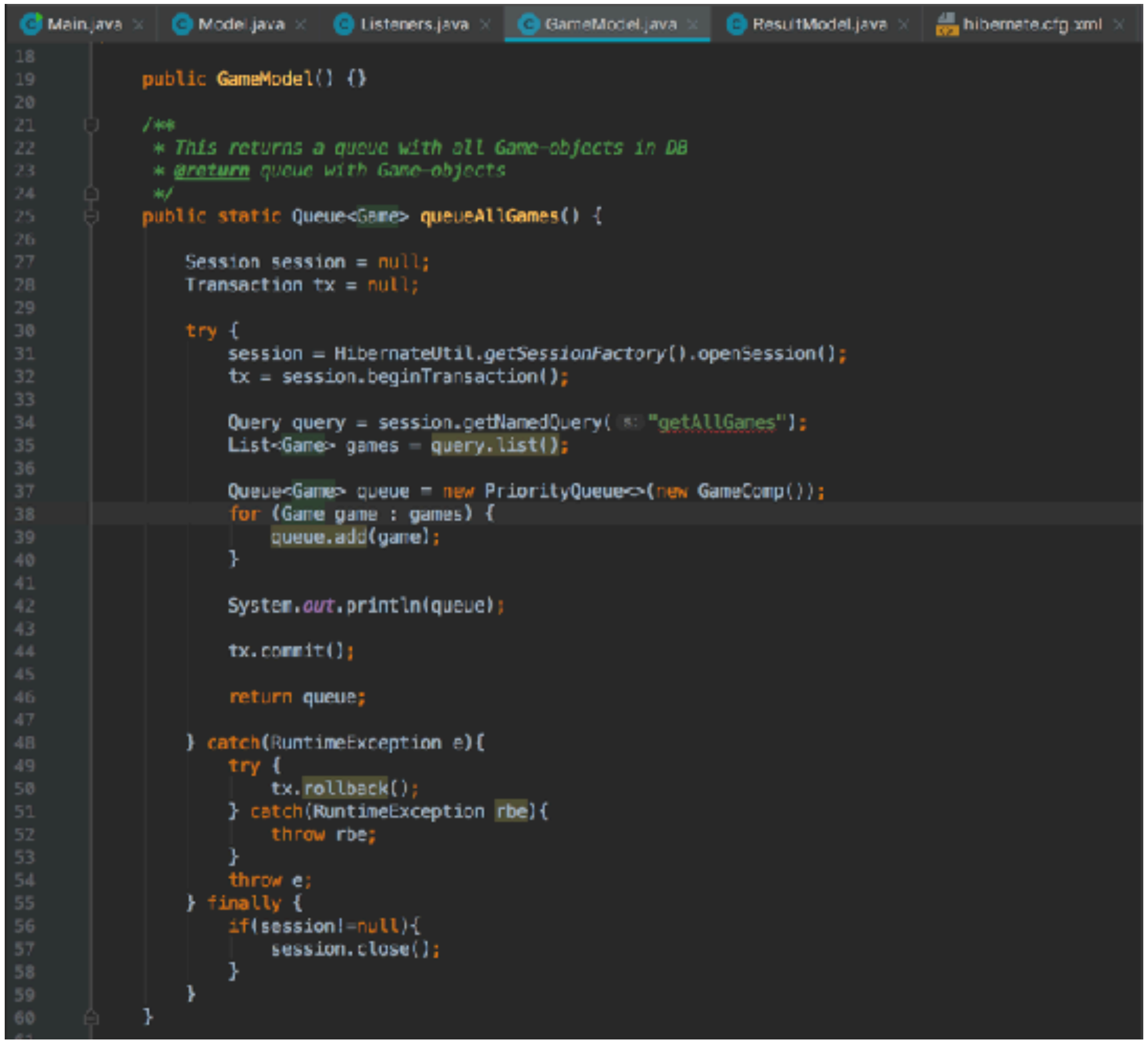
CR**U**D ✓

Show ranking table
(ResultModel.getResultsTeam()
= addMenuLinkListener3()
()

Search player
(PlayerModel.findPlayerByName()
= addSearchButtonListeners()
()

Sökfunktion ✓

PriorityQueue (datalogisk klass)

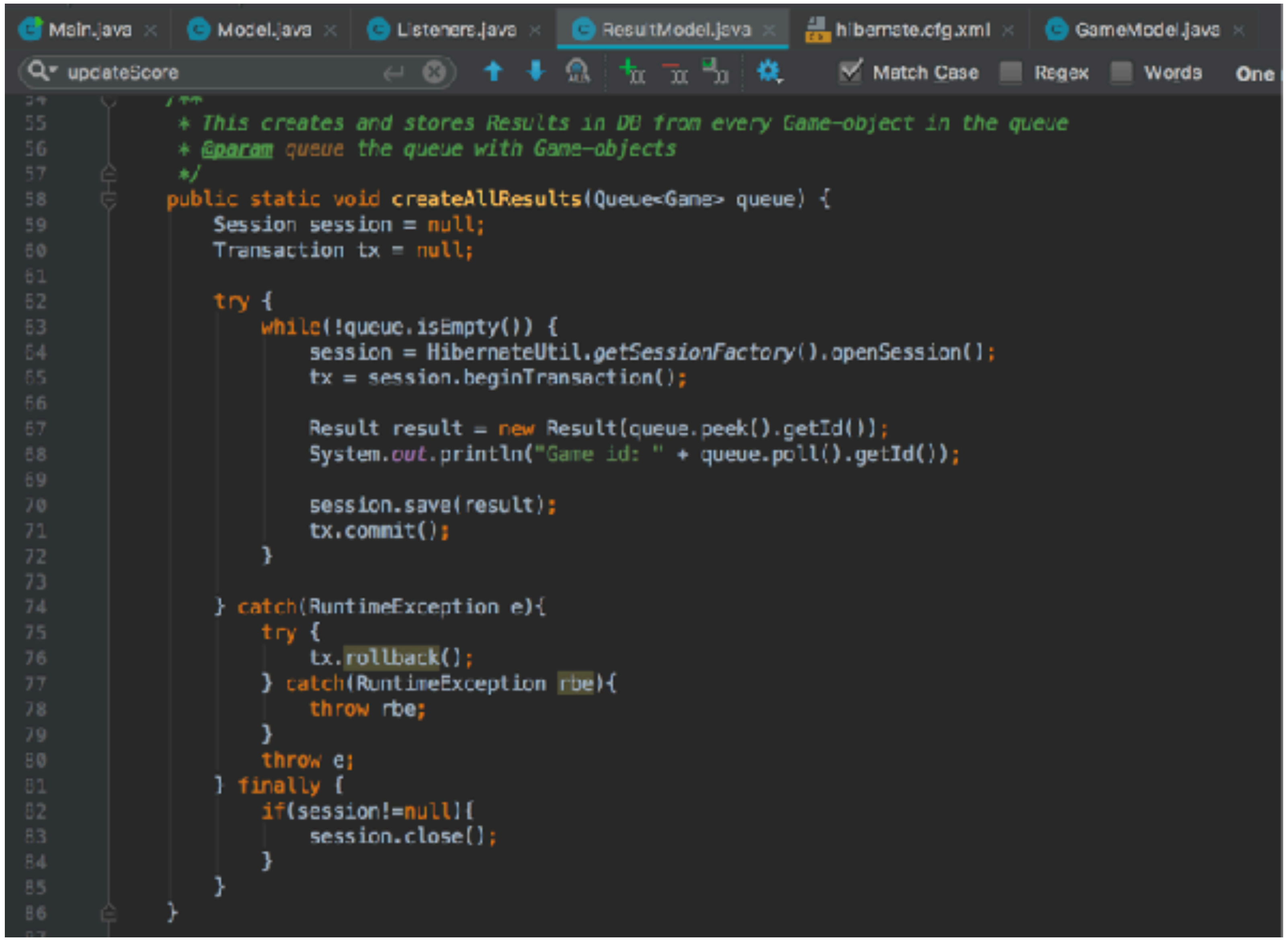

Main.java Model.java Listeners.java GameModel.java ResultModel.java hibernate.cfg.xml

```java
    public GameModel() {}

    /**
     * This returns a queue with all Game-objects in DB
     * @return queue with Game-objects
     */
    public static Queue<Game> queueAllGames() {

        Session session = null;
        Transaction tx = null;

        try {
            session = HibernateUtil.getSessionFactory().openSession();
            tx = session.beginTransaction();

            Query query = session.getNamedQuery( s: "getAllGames");
            List<Game> games = query.list();

            Queue<Game> queue = new PriorityQueue<>(new GameComp());
            for (Game game : games) {
                queue.add(game);
            }

            System.out.println(queue);

            tx.commit();

            return queue;

        } catch(RuntimeException e){
            try {
                tx.rollback();
            } catch(RuntimeException rbe){
                throw rbe;
            }
            throw e;
        } finally {
            if(session!=null){
                session.close();
            }
        }
    }
```

```java
Queue<Game> queue = new
PriorityQueue<>(new GameComp());
for (Game game : games) {
    queue.add(game);
}
```

Användning av datalogis klass

```
Result result = new Result(queue.peek().getId());
```



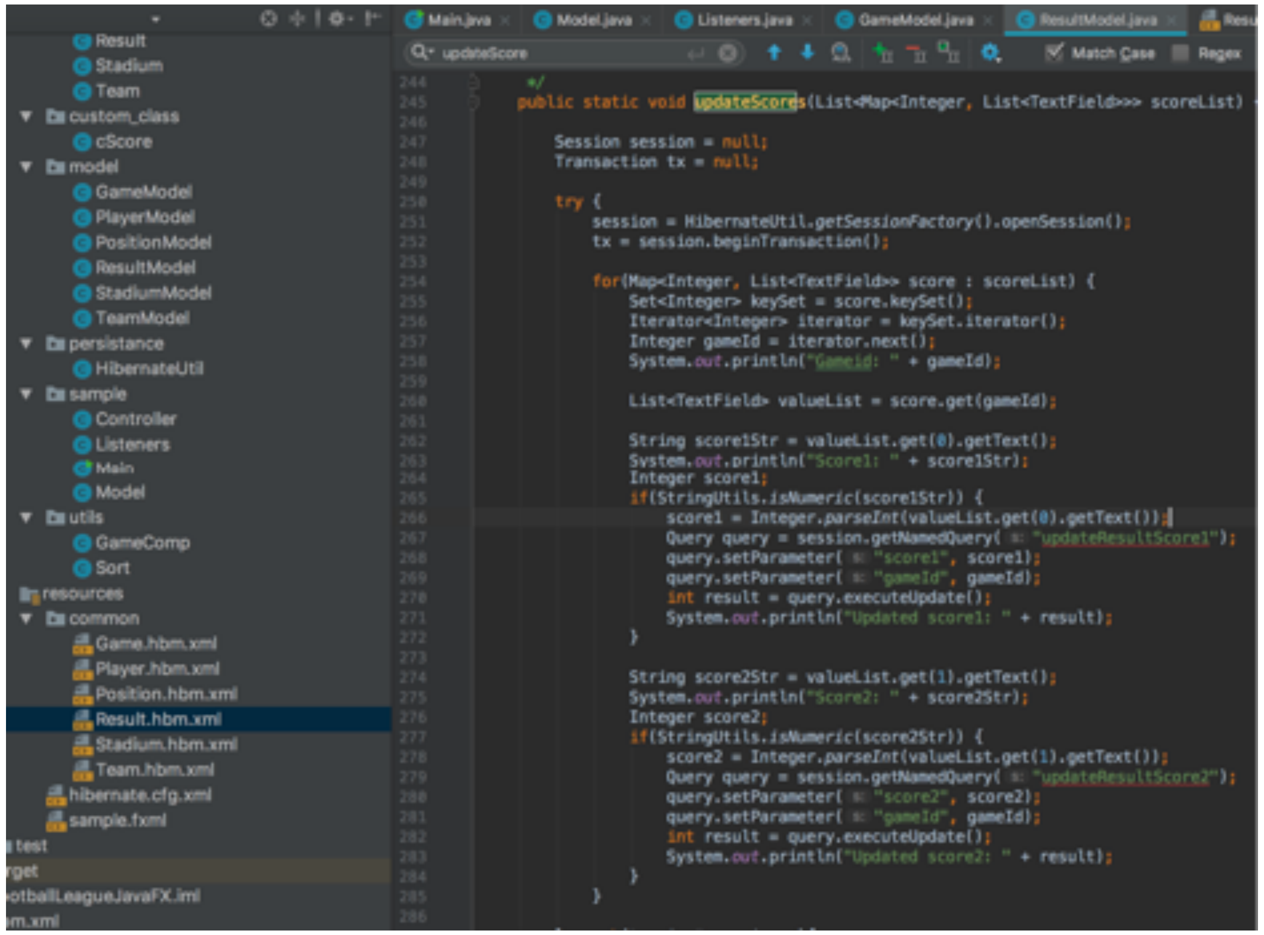
```java
/**
 * This creates and stores Results in DB from every Game-object in the queue
 * @param queue the queue with Game-objects
 */
public static void createAllResults(Queue<Game> queue) {
    Session session = null;
    Transaction tx = null;

    try {
        while(!queue.isEmpty()) {
            session = HibernateUtil.getSessionFactory().openSession();
            tx = session.beginTransaction();

            Result result = new Result(queue.peek().getId());
            System.out.println("Game id: " + queue.poll().getId());

            session.save(result);
            tx.commit();
        }

    } catch(RuntimeException e){
        try {
            tx.rollback();
        } catch(RuntimeException rbe){
            throw rbe;
        }
        throw e;
    } finally {
        if(session!=null){
            session.close();
        }
    }
}
```

Named SQL Queries

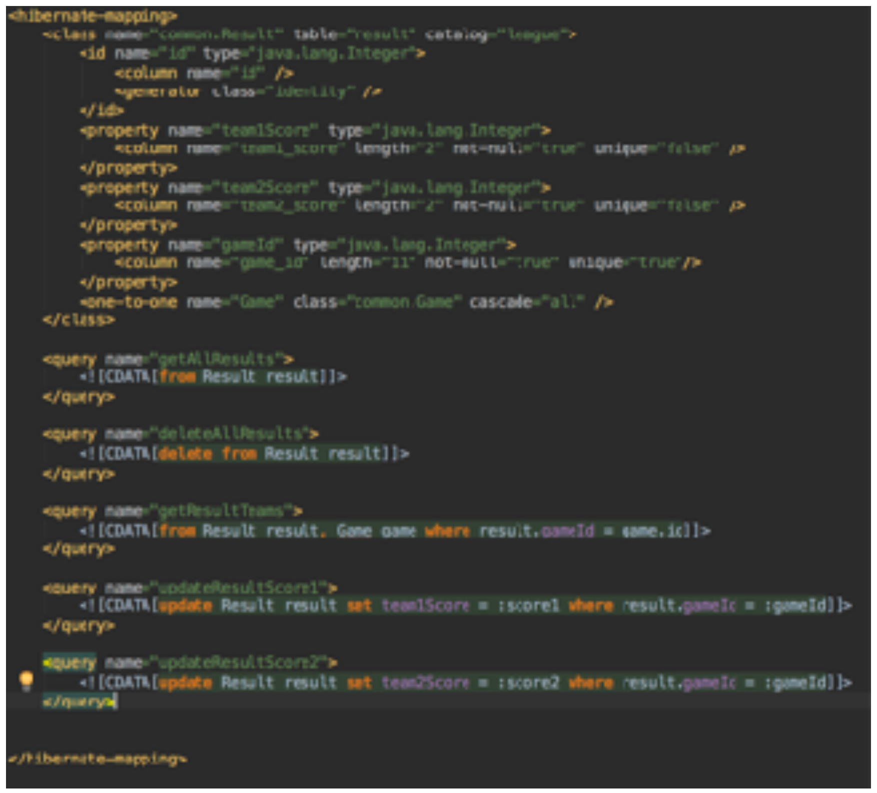
```java
Query query = session.getNamedQuery("updateResultScore1");
```

```java
query.setParameter("score1", score1);
query.setParameter("gameId", gameId);
```

```xml
<query name="updateResultScore1">
    <![CDATA[update Result result set team1Score = :score1 where result.gameId = :gameId]]>
</query>
```

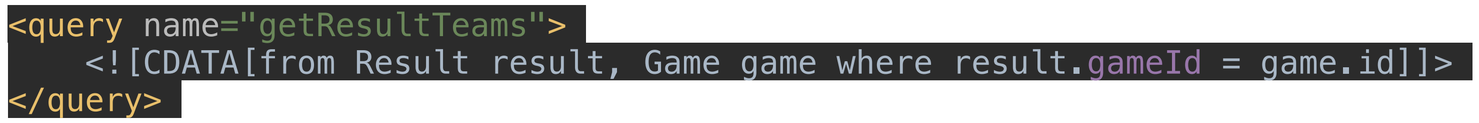


```java
     */
    public static void updateScores(List<Map<Integer, List<TextField>>> scoreList)

        Session session = null;
        Transaction tx = null;

        try {
            session = HibernateUtil.getSessionFactory().openSession();
            tx = session.beginTransaction();

            for(Map<Integer, List<TextField>> score : scoreList) {
                Set<Integer> keySet = score.keySet();
                Iterator<Integer> iterator = keySet.iterator();
                Integer gameId = iterator.next();
                System.out.println("GameId: " + gameId);

                List<TextField> valueList = score.get(gameId);

                String score1Str = valueList.get(0).getText();
                System.out.println("Score1: " + score1Str);
                Integer score1;
                if(StringUtils.isNumeric(score1Str)) {
                    score1 = Integer.parseInt(valueList.get(0).getText());
                    Query query = session.getNamedQuery( s: "updateResultScore1");
                    query.setParameter( s: "score1", score1);
                    query.setParameter( s: "gameId", gameId);
                    int result = query.executeUpdate();
                    System.out.println("Updated score1: " + result);
                }

                String score2Str = valueList.get(1).getText();
                System.out.println("Score2: " + score2Str);
                Integer score2;
                if(StringUtils.isNumeric(score2Str)) {
                    score2 = Integer.parseInt(valueList.get(1).getText());
                    Query query = session.getNamedQuery( s: "updateResultScore2");
                    query.setParameter( s: "score2", score2);
                    query.setParameter( s: "gameId", gameId);
                    int result = query.executeUpdate();
                    System.out.println("Updated score2: " + result);
                }
            }
```

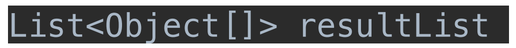


```xml
<hibernate-mapping>
    <class name="common.Result" table="result" catalog="league">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="identity" />
        </id>
        <property name="team1Score" type="java.lang.Integer">
            <column name="team1_score" length="2" not-null="true" unique="false" />
        </property>
        <property name="team2Score" type="java.lang.Integer">
            <column name="team2_score" length="2" not-null="true" unique="false" />
        </property>
        <property name="gameId" type="java.lang.Integer">
            <column name="game_id" length="11" not-null="true" unique="true"/>
        </property>
        <one-to-one name="Game" class="common.Game" cascade="all" />
    </class>

    <query name="getAllResults">
        <![CDATA[from Result result]]>
    </query>

    <query name="deleteAllResults">
        <![CDATA[delete from Result result]]>
    </query>

    <query name="getResultTeams">
        <![CDATA[from Result result, Game game where result.gameId = game.id]]>
    </query>

    <query name="updateResultScore1">
        <![CDATA[update Result result set team1Score = :score1 where result.gameId = :gameId]]>
    </query>

    <query name="updateResultScore2">
        <![CDATA[update Result result set team2Score = :score2 where result.gameId = :gameId]]>
    </query>


</hibernate-mapping>
```

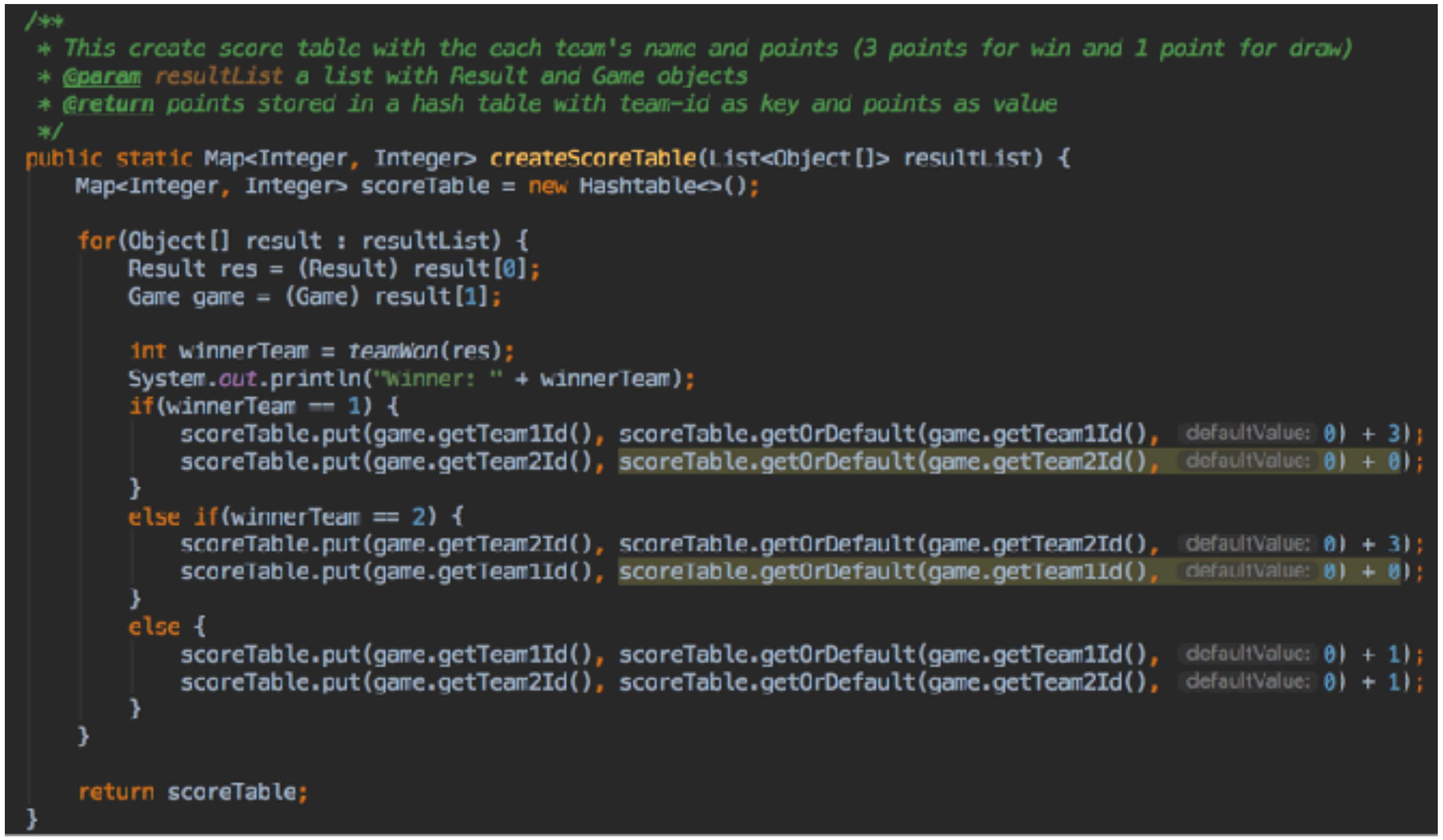
HQL join = mixat objekt



```
<query name="getResultTeams">
    <![CDATA[from Result result, Game game where result.gameId = game.id]]>
</query>
```



```
List<Object[]> resultList
```



```
/**
 * This create score table with the each team's name and points (3 points for win and 1 point for draw)
 * @param resultList a list with Result and Game objects
 * @return points stored in a hash table with team-id as key and points as value
 */
public static Map<Integer, Integer> createScoreTable(List<Object[]> resultList) {
    Map<Integer, Integer> scoreTable = new Hashtable<>();

    for(Object[] result : resultList) {
        Result res = (Result) result[0];
        Game game = (Game) result[1];

        int winnerTeam = teamWon(res);
        System.out.println("Winner: " + winnerTeam);
        if(winnerTeam == 1) {
            scoreTable.put(game.getTeam1Id(), scoreTable.getOrDefault(game.getTeam1Id(),  defaultValue: 0) + 3);
            scoreTable.put(game.getTeam2Id(), scoreTable.getOrDefault(game.getTeam2Id(),  defaultValue: 0) + 0);
        }
        else if(winnerTeam == 2) {
            scoreTable.put(game.getTeam2Id(), scoreTable.getOrDefault(game.getTeam2Id(),  defaultValue: 0) + 3);
            scoreTable.put(game.getTeam1Id(), scoreTable.getOrDefault(game.getTeam1Id(),  defaultValue: 0) + 0);
        }
        else {
            scoreTable.put(game.getTeam1Id(), scoreTable.getOrDefault(game.getTeam1Id(),  defaultValue: 0) + 1);
            scoreTable.put(game.getTeam2Id(), scoreTable.getOrDefault(game.getTeam2Id(),  defaultValue: 0) + 1);
        }
    }

    return scoreTable;
}
```