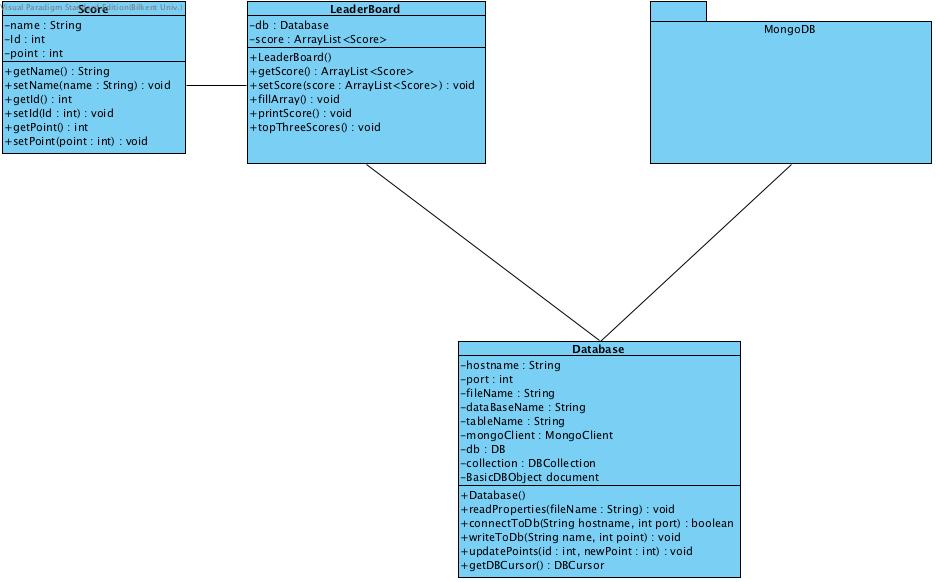
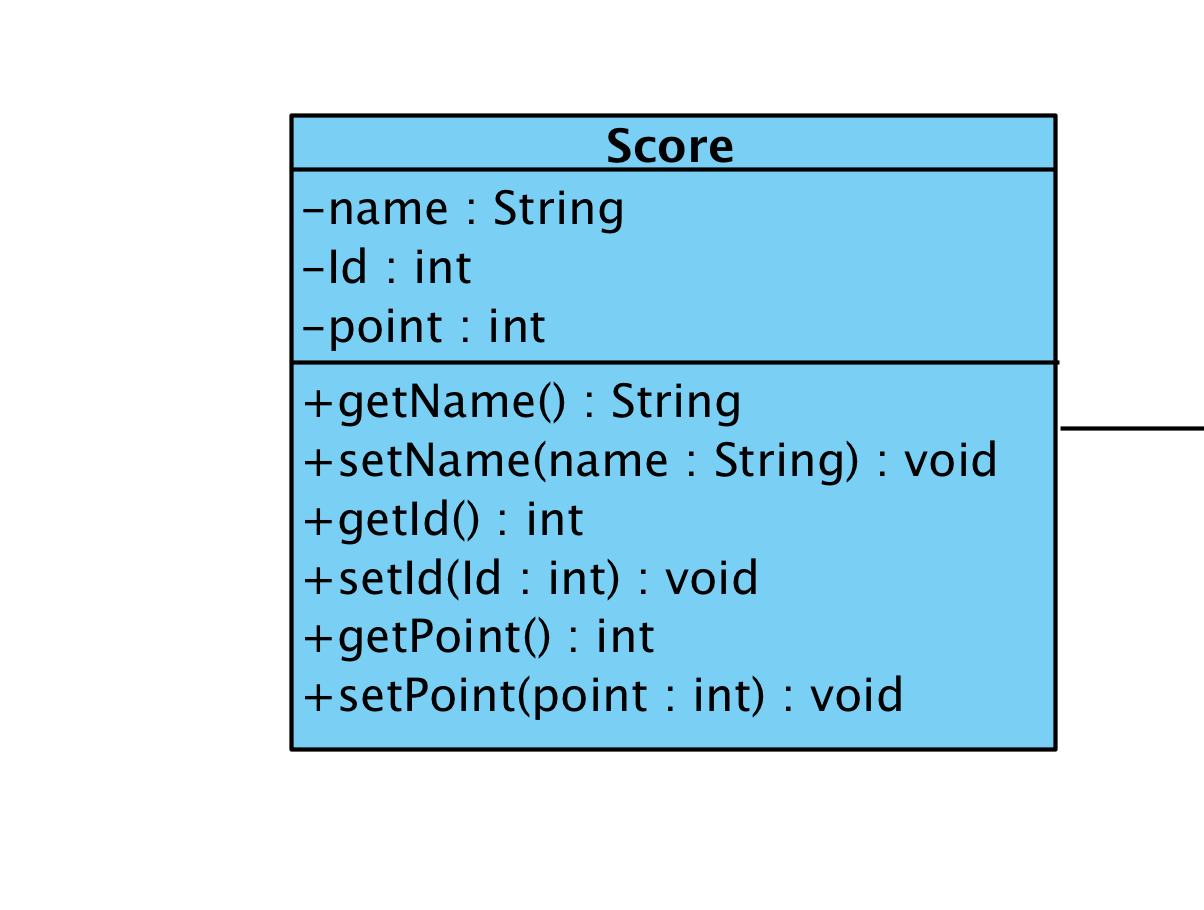
* + 1. **Adapter Pattern**

In software engineering, the adapter pattern is a software design pattern that allows the interface of an existing class to be used from another interface. It is often used to make existing classes work with others without modifying their source code. An adapter helps two incompatible interfaces to work together. In our program, adapter pattern enables two incompatible classes to work together. There classes are Database and Mongodb. MongoDB is a cross-platform document-oriented database. In order connect to database and to use it, we need adapter class, so it is Database class in our case. Database class lets Mongodb and LeaderBoard class to work together. So, LeaderBoard can connect to database using Database class’s methods. It can manage database, add, remove modify in database easily. Therefore, Adapter pattern provides us with easy access to other objects that monogdb without modifying our class.



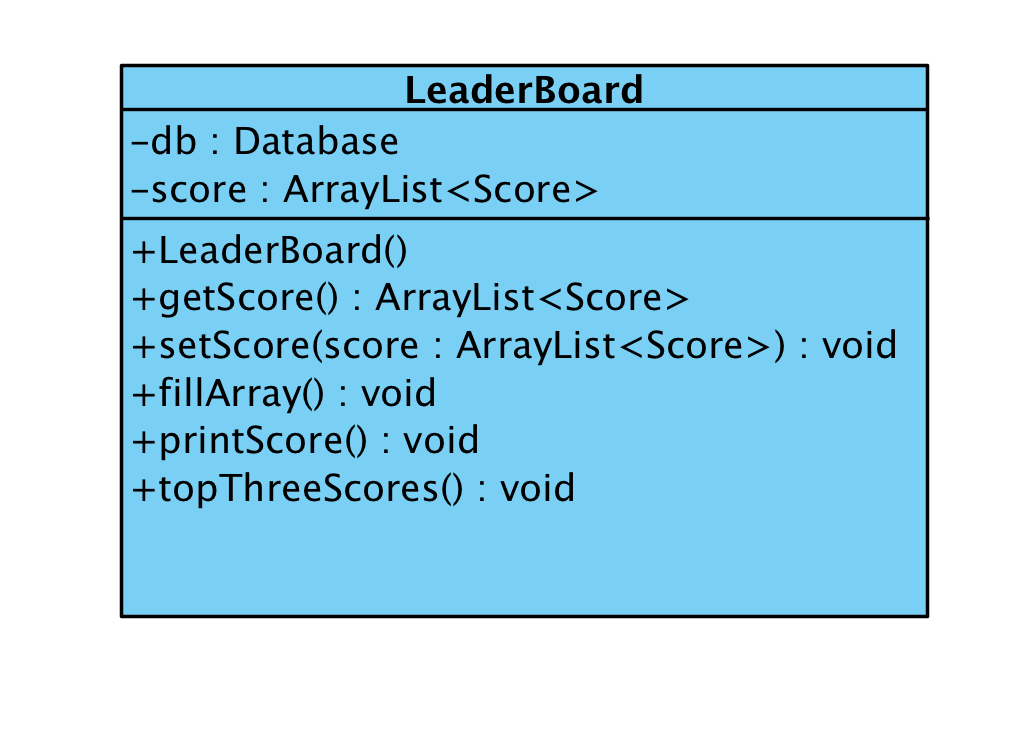
**5.1 Class Interfaces**

**SCORE CLASS**

****

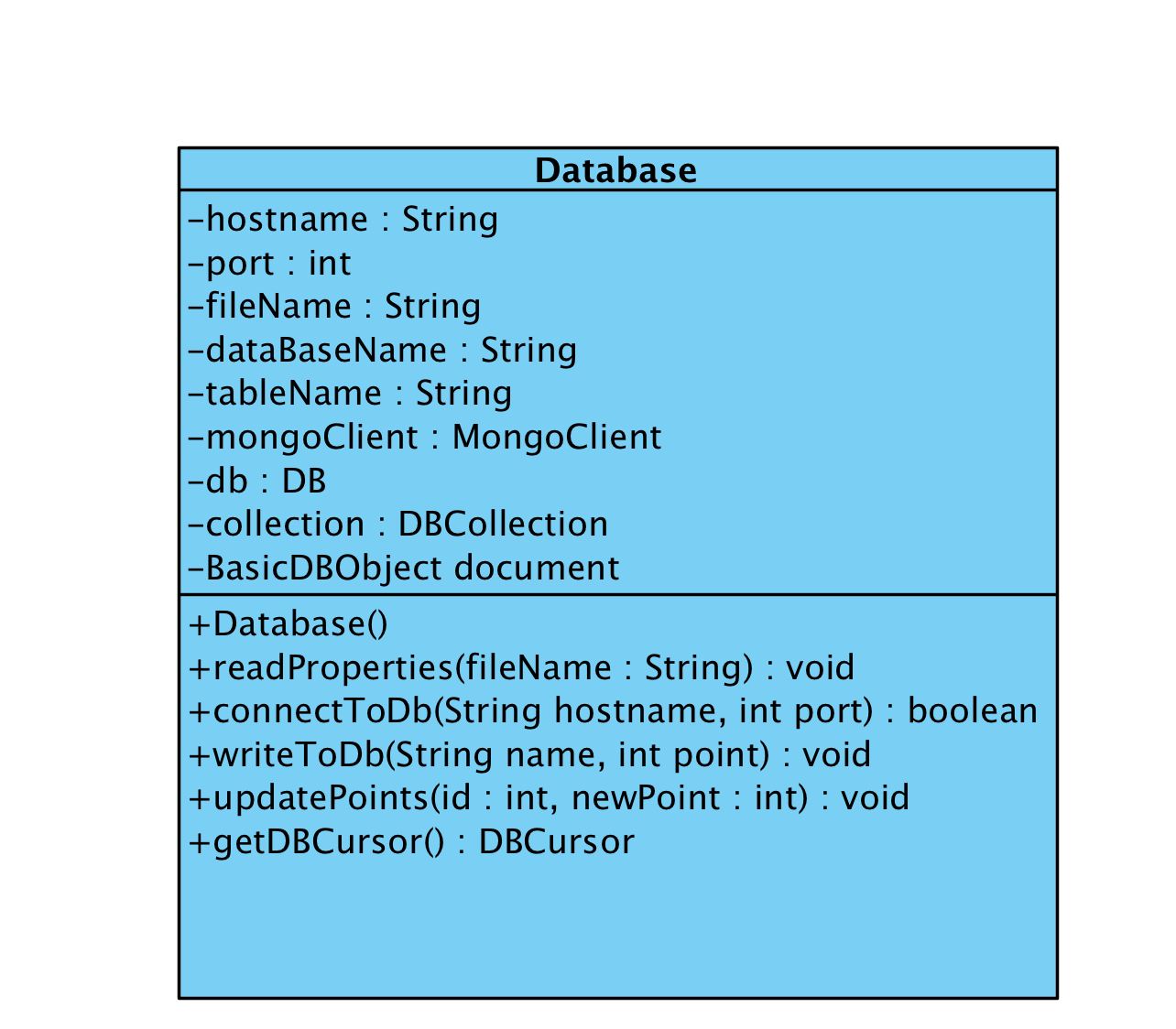
Score Class is used to take name of user, and point of his/ her after that it will write these data to database. So, Score class has 3 private property such that name(String), id(int), point(int). It has just one default constructor. However there are 6 public methods. These 6 methods are setter and getters of private variables. These methods used for easy get value or set it to score object. Due to the fact that score variables are private, we can’t access to them directly so we have to use these methods. Name of these methods are getName(), setName(String name), getId(), setId(int id), getPoint(), setPoint(int point).

**LeaderBoard CLASS**



Leaderboard class has two private variables or properties. They are db and type is Database, second one is array list so, score and type is ArrayList<Score>. The array list of score will store all score objects in this list. LeaderBoard class has one default constructor that will just call fillArray() method. Lets talk about our methods. There are 6 methods that will help us to get data and put it leaderboard. User can see all his/her opponents’ score and name. It will get all players name, score and will add all of them to array list as score object. However, it will just show 10 best players score at leaderboard panel. First method is getScore(), it will return score Arraylist which stores all of the database object or score objects in order to other classes can also use these data. Second important method is fillArray() which takes all database objects and maps it to array after that it sets name, id and point by using score class’s set methods. When it ready it returns Score as object and finally it writes this object to score array list. printScores(), and topThreeScores methods print result of score to panel. Finally, readFromDb() method is able to read all data from database.

**Database CLASS**

****

Database class is like an adapter. It enables our program to connect mongodb which is a cross-platform document-oriented database and create database if it is not exists, and it also create table or collections of course if they are not exist. Main job of database class to enable other classes to connect mongodb that is our database. With the help of this class other class can reach all data in the database. Database class has 9 private property, 3 of them is used for to connect Mongodb (database) and else is used for create database or reach our data in database. Names of these private variables are hostname (String), port(Integer), filename(String), databaseName(String), mongoClient(MongoClient), db(DB), collection (DBCollection), document (BasicDBObject). Database class has 1 constructor. Actually, most of the processes are going in this constructor. Therefore, firstly, it reads all properties from properties file. To read these data there is readProperties(String fileName) method which will read all properties and initialize them to local variables in order to use them in our program. When all variables are initialized, it calls connectToDB(String hostname, int port) method to connect mongodb(database). If this method return true, it starts to create database and tables, if it not exists, otherwise it will not create and will continue to run other lines of code. If it can’t connect to mentioned database, it will try for several times to connect. At the end if it can’t connect to there it will show error message. When our database class is connected to mongodb, so we are ready to write or read to/from there. In order to write to database, it has writeToDb(String name, int point) method that will write user name and his/her score to application’s database. We have one more method that updatePoint (int id , int newPoint). It can update point or score of current player.

**InfoPanel CLASS**

InfoPanel class uses methods of Leaderbord. It enables us to put top three players name and score to gameplay GUI, so, at the right side we can see top 3 players name and score while playing this game. In addition, it will print users own score and his/ her name.

**Help**

Help class includes all instructions about the game. Player can read all instructions such that play pause/button, which case to move or where should put his name etc. Help class will be used just for getting information about game.

## 5.3 Specifying Contracts

Database:

1. **context** Database:: readProperties(fileName:String) **pre**:

filesExists(filename:String)

In order to read properties that contain hostname and port to connect db, firstly it will check existence of file.

1. **context** Database:: connectToDb(hostname:Sting, port:int) **post**:

isConnected () = true;

If Database can connect to monogdb(database) isConnected() will return true otherwise it will be false.

1. **context** Database:: writeToDb(name:String, point:int)**pre**:

isConnected()

In order to write all data to database, we need to get true from isConnected method. If it is not true, it will try again to connect but if it is false it will print out error message.

1. **context** LeaderBoard::fillArray() **pre**:

db.getDBCursor() != null

db is database object which was created first. In order to use fillArray() method, we need to get DBCursor object. It will fill array using these objects

1. **context** LeaderBoard::fillArray() **post**:

setArray(Score score)

Leaderboard class’s fillArray() method called to fill array with Score objects, when array is fully filled, it will be set to score arraylist . After that we can get score arraylist whenever we need to.

1. **context** Score:: setPoint(int point) **pre:**

getName() != “”

in order to set point to score object, it checks whether name is null or not, if it is not null it lets to set name.