Matt O’Leary
Avery Yeats
Brendan Lancaster

Semester Project Update PDF

Github Link: https://github.com/mattjoelar/CSCI4448-semester-project-hw7.git

Status Summary

Project Name: Projectmon
Team: Matt O’Leary, Avery Yeats, Brendan Lancaster
Work Done:

Matt O’leary- UML Diagram, BDD Scenarios, Update PDF

Brendan Lancaster: I researched a framework for us to use and coordinated work between us. I wrote out the server game logic up to its current point. I also spent a significant amount of time trying to get OpenGL ES to work with Android Studio, but dropped it to conserve time :(

Avery Yeats- Created MySql server database and database functions using SpringBoot, CrudRepository, Spring Data JPA and Spring Web

Issues Encountered:

Learning how to use Websockets had some hiccups, but were very minor.

As we need to set up a multiplayer game which involves serializing much of the game state, implementing object-oriented programming techniques is proving to be difficult. We’ve attempted to stay faithful to this class in the ways we write code but adhering to them exactly doesn’t make sense for many things we’re trying to accomplish.

Patterns:

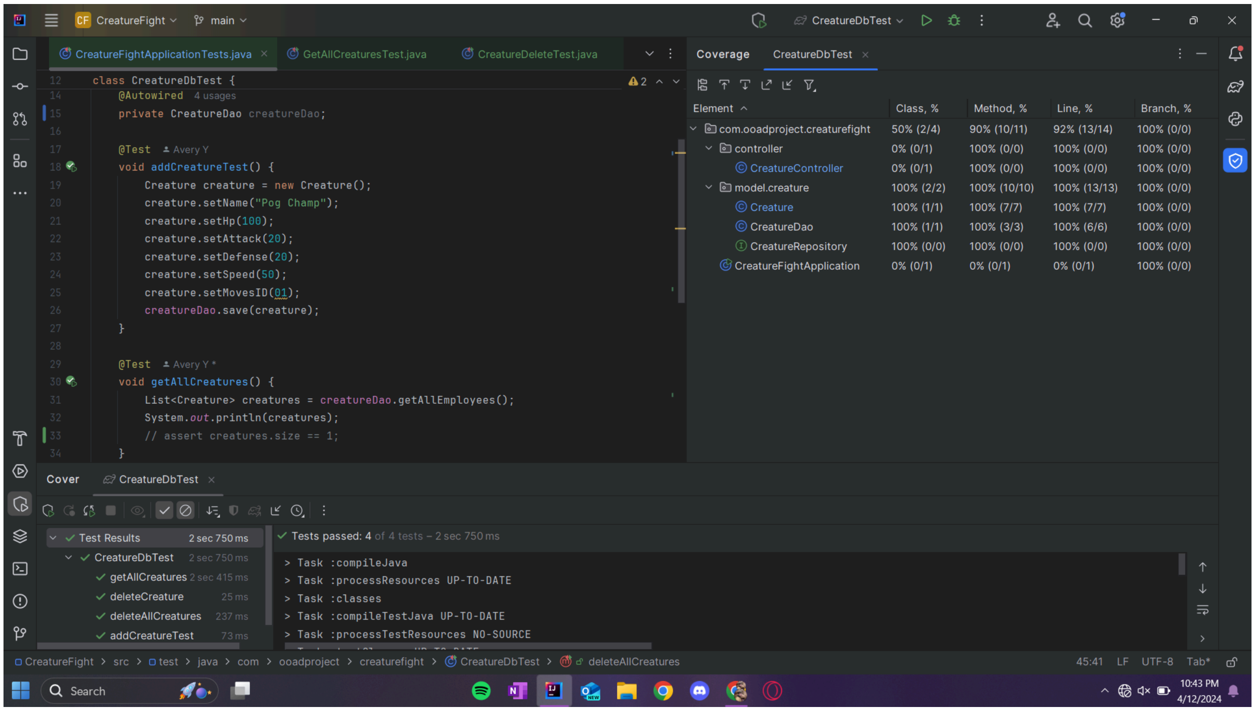
Observer Pattern: Clients wait for signals from the server to receive and process messages using JSON

Factory Pattern: Projectmon and their data structures are created through factory classes or factory-like functions

Singleton Pattern: Projectmon builders access an “Entries” singleton which contains all pre-built structures

Adapter Pattern: Player requests and world updates are parsed from JSON formats (in anticipation of using JSON to communicate between the server and clients)

Test Coverage jpg:



```java
class CreatureDbTest {
    @Autowired  4 usages
    private CreatureDao creatureDao;

    @Test  Avery Y
    void addCreatureTest() {
        Creature creature = new Creature();
        creature.setName("Pog Champ");
        creature.setHp(100);
        creature.setAttack(20);
        creature.setDefense(20);
        creature.setSpeed(50);
        creature.setMovesID(01);
        creatureDao.save(creature);
    }

    @Test  Avery Y *
    void getAllCreatures() {
        List<Creature> creatures = creatureDao.getAllEmployees();
        System.out.println(creatures);
        // assert creatures.size == 1;
    }
```

Element	Class, %	Method, %	Line, %	Branch, %
com.ooadproject.creaturefight	50% (2/4)	90% (10/11)	92% (13/14)	100% (0/0)
controller	0% (0/1)	100% (0/0)	100% (0/0)	100% (0/0)
CreatureController	0% (0/1)	100% (0/0)	100% (0/0)	100% (0/0)
model.creature	100% (2/2)	100% (10/10)	100% (13/13)	100% (0/0)
Creature	100% (1/1)	100% (7/7)	100% (7/7)	100% (0/0)
CreatureDao	100% (1/1)	100% (3/3)	100% (6/6)	100% (0/0)
CreatureRepository	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
CreatureFightApplication	0% (0/1)	0% (0/1)	0% (0/1)	100% (0/0)

✓ Test Results	2 sec 750 ms
✓ CreatureDbTest	2 sec 750 ms
✓ getAllCreatures	2 sec 415 ms
✓ deleteCreature	25 ms
✓ deleteAllCreatures	237 ms
✓ addCreatureTest	73 ms

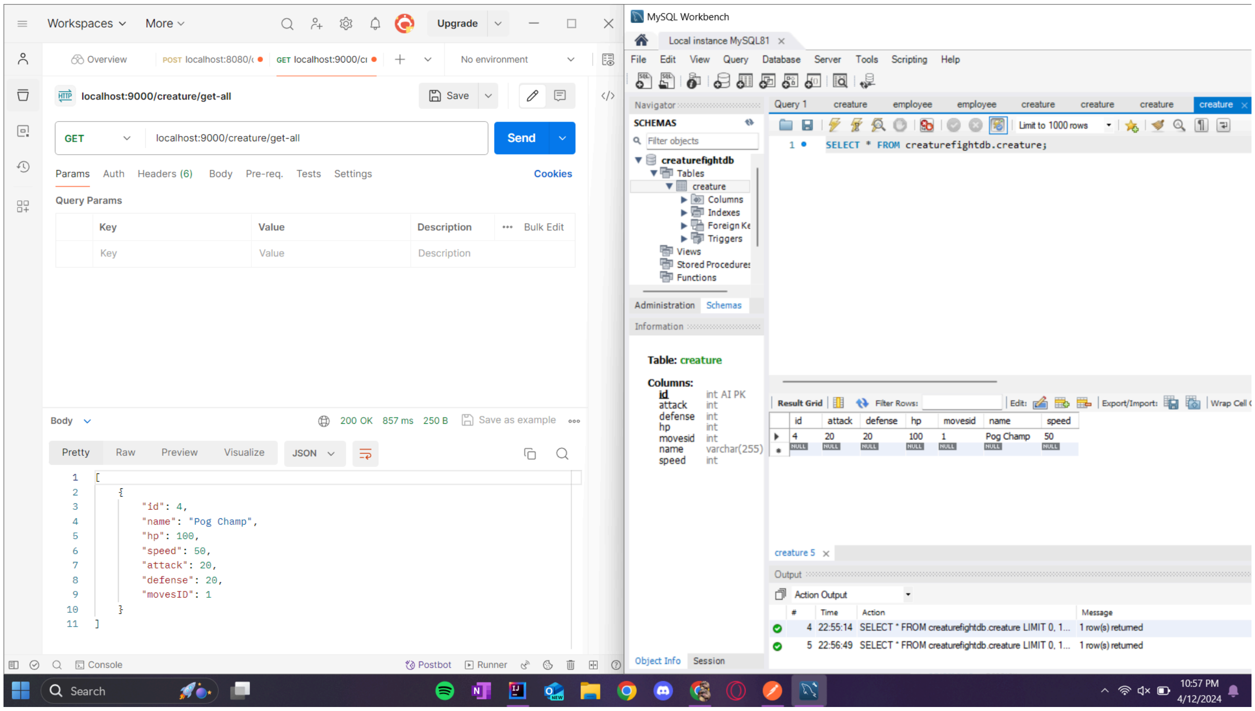
✓ Tests passed: 4 of 4 tests – 2 sec 750 ms

```
> Task :compileJava
> Task :processResources UP-TO-DATE
> Task :classes
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
```

CreatureFight > src > test > java > com > ooadproject > creaturefight > CreatureDbTest > deleteAllCreatures

45:41 LF UTF-8 Tab*

* The test coverage for the get and post mapping is not considered in the coverage report because it is tested using the Postman application.



GET localhost:9000/creature/get-all

Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body 200 OK 857 ms 250 B

Pretty Raw Preview Visualize JSON

```
[
    {
        "id": 4,
        "name": "Pog Champ",
        "hp": 100,
        "speed": 50,
        "attack": 20,
        "defense": 20,
        "movesID": 1
    }
]
```

MySQL Workbench — Local instance MySQL81

```
SELECT * FROM creaturefightdb.creature;
```

SCHEMAS: creaturefightdb > Tables > creature (Columns, Indexes, Foreign Ke, Triggers), Views, Stored Procedures, Functions

Table: creature

Columns:

id	int AI PK
attack	int
defense	int
hp	int
movesid	int
name	varchar(255)
speed	int

Result Grid

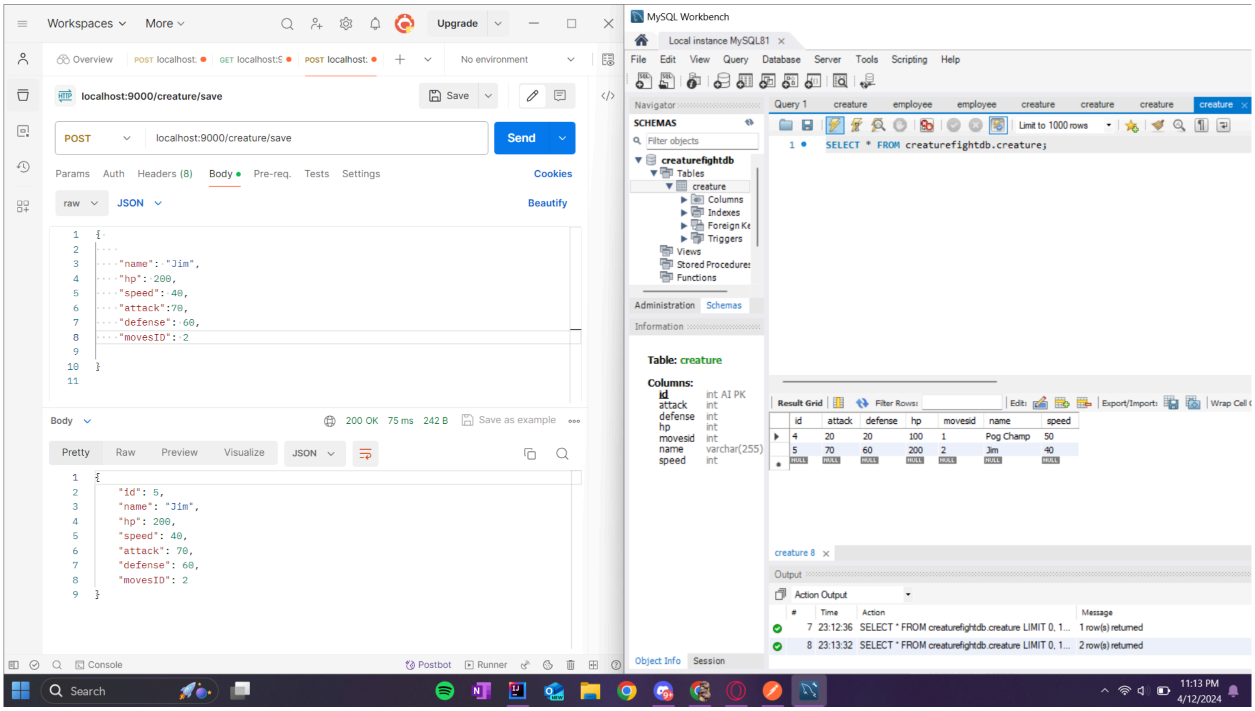
id	attack	defense	hp	movesid	name	speed
4	20	20	100	1	Pog Champ	50
NULL	NULL	NULL	NULL	NULL	NULL	NULL

creature 5

Output — Action Output

```
#  Time      Action                                               Message
4  22:55:14  SELECT * FROM creaturefightdb.creature LIMIT 0, 1...  1 row(s) returned
5  22:56:49  SELECT * FROM creaturefightdb.creature LIMIT 0, 1...  1 row(s) returned
```

10:57 PM 4/12/2024



POST localhost:9000/creature/save

Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

```
{
    
    "name": "Jim",
    "hp": 200,
    "speed": 40,
    "attack":70,
    "defense": 60,
    "movesID": 2


}
```

Body 200 OK 75 ms 242 B

Pretty Raw Preview Visualize JSON

```
{
    "id": 5,
    "name": "Jim",
    "hp": 200,
    "speed": 40,
    "attack": 70,
    "defense": 60,
    "movesID": 2
}
```

MySQL Workbench — Local instance MySQL81

```
SELECT * FROM creaturefightdb.creature;
```

Table: creature

Columns:

id	int AI PK
attack	int
defense	int
hp	int
movesid	int
name	varchar(255)
speed	int

Result Grid

id	attack	defense	hp	movesid	name	speed
4	20	20	100	1	Pog Champ	50
5	70	60	200	2	Jim	40
NULL	NULL	NULL	NULL	NULL	NULL	NULL

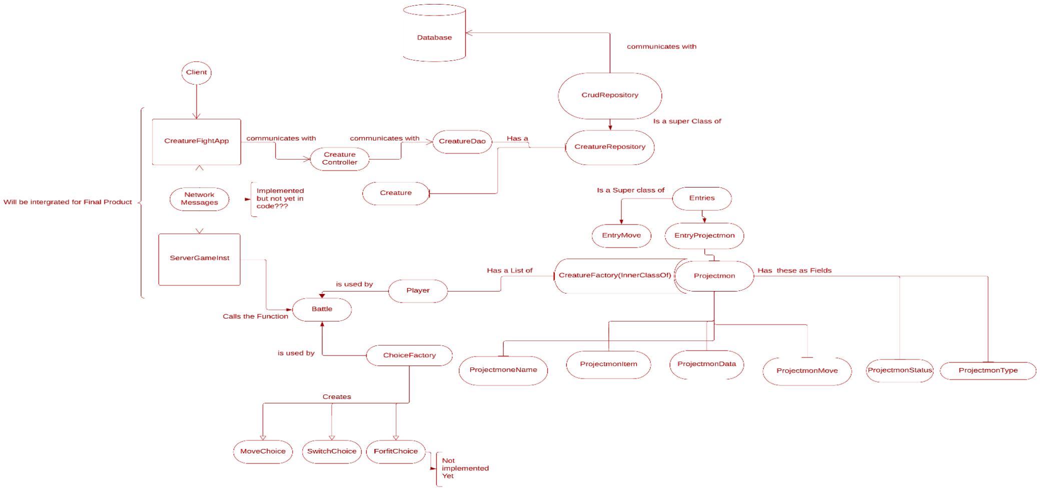
creature 8

Output — Action Output

```
#  Time      Action                                               Message
7  23:12:36  SELECT * FROM creaturefightdb.creature LIMIT 0, 1...  1 row(s) returned
8  23:13:32  SELECT * FROM creaturefightdb.creature LIMIT 0, 1...  2 row(s) returned
```

11:13 PM 4/12/2024

Class UML Diagram



BDD Testing

Scenario:

Given user is connected to the server

When I wait for another user to connect

Then we should start playing the game

Scenario:

Given user is in a battle

When I make my choice for the turn

Then the game will wait for the other player to make their choice

Scenario:

Given that the other player has made a choice

When I make my choice for the turn

Then the game will run the turn

Scenario:

Given a player chooses creatures for their party

When the player hits save

The game will save their party to the database

Scenario:

Given a two players connect to the server

When the game runs

Each player will be shown a list of creatures grabbed from the database

Scenario:

Given a player has saved a party previously

When the player clicks the delete button

When party is deleted from the database

Plan For Next Iteration:

The next stage of the project is to integrate the server “game” logic into the network code, allowing two clients to connect to the server and interact with that system that sends information back to the clients. In addition, more creative work needs to be done to determine the gameplay traits of the Projectmon creatures and game mechanics. For the final project, we also plan to include a matchmaking system and display in a separate project for players.

Recorded Demo Link:

Please note that much of the code is still a work-in-progress and as a result is not fully functional: https://youtu.be/HSB2Rm95Zng