



# *ZOO MANAGEMENT SYSTEM*

-DATABASE & OBJECT-ORIENTED PROGRAMMING PROJECT-

2<sup>nd</sup> YEAR, 1<sup>st</sup> SEMESTER

Student: Mara Mureșan

Group: 30423

Project Supervisor: Dr. Ing. Călin Cenan

Project Supervisor: Ing. Alexandru Ștefan Gârleanu

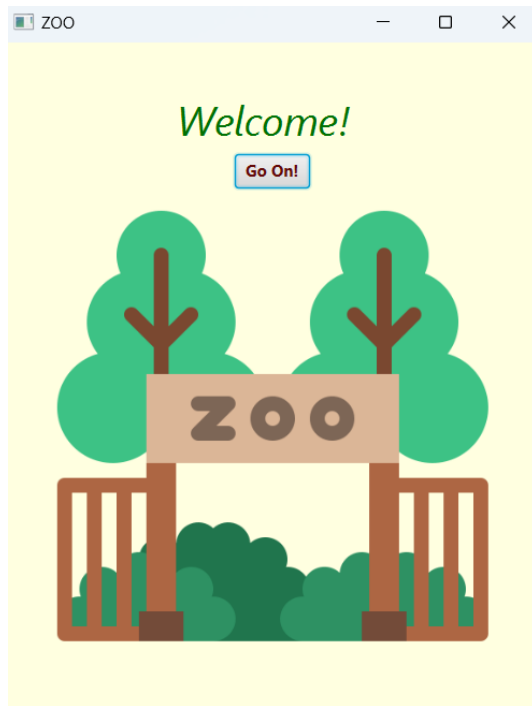
9.01.2024

## *TABLE OF CONTENTS*

1.	<i>TASK DESCRIPTION</i> .....	3
2.	<i>DATABASE</i> .....	8
	Components .....	8
	Scripts .....	10
3.	<i>JAVA INTERFACE</i> .....	11
	Components .....	11
	Diagrams .....	12
4.	<i>FUTURE DEVELOPMENTS</i> .....	16
5.	<i>REFERENCES</i> .....	16

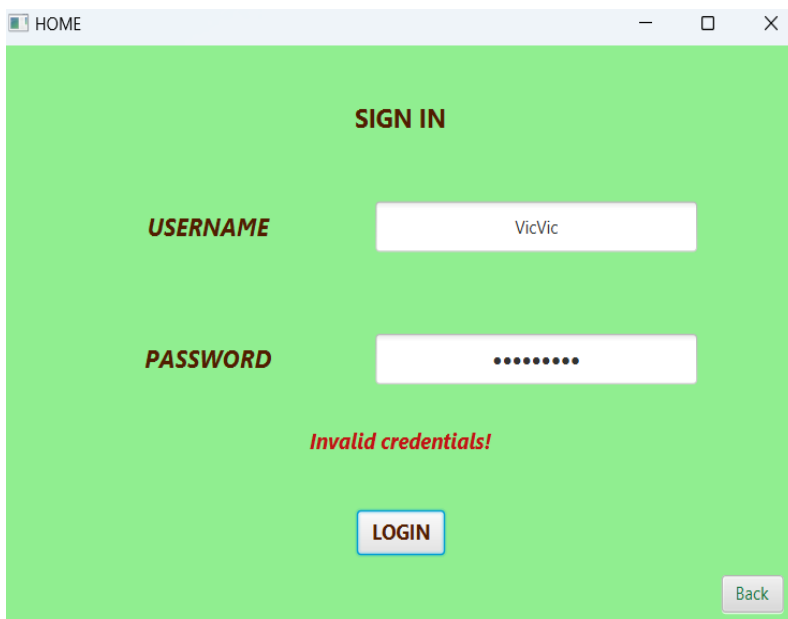
## 1. TASK DESCRIPTION

This is a Java application designed for zoo management. The application utilizes a zoo database to enhance the experience for both workers and visitors. It covers key functionalities such as worker portals, environment and animal tracking, visitor management, ticketing, and reviews.



The first window that appears when the application is run.

When pressing the 'Go On!' button, the next window appears.

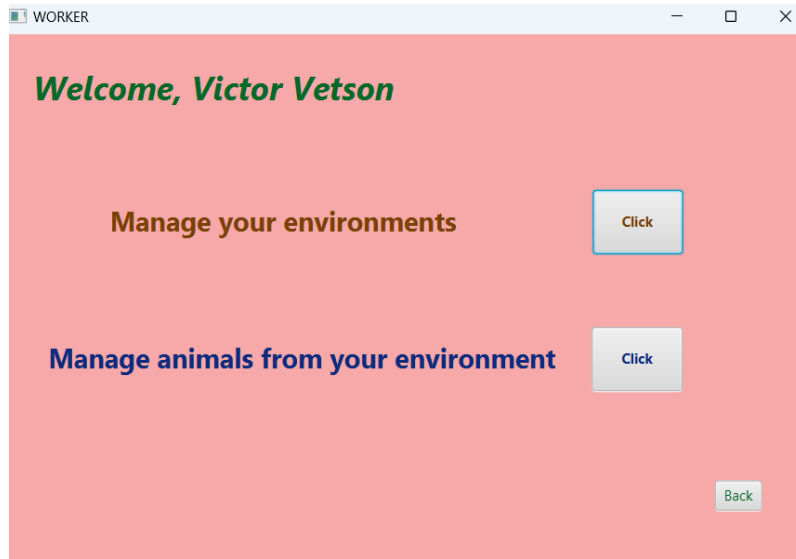


The user should write his username/password and press 'LOGIN' button.

If this information was wrongly introduced, "Invalid credentials" message is displayed.

Else, it goes to the account window depending on the type of the user: worker/visitor.

'Back' button goes to the previous window.

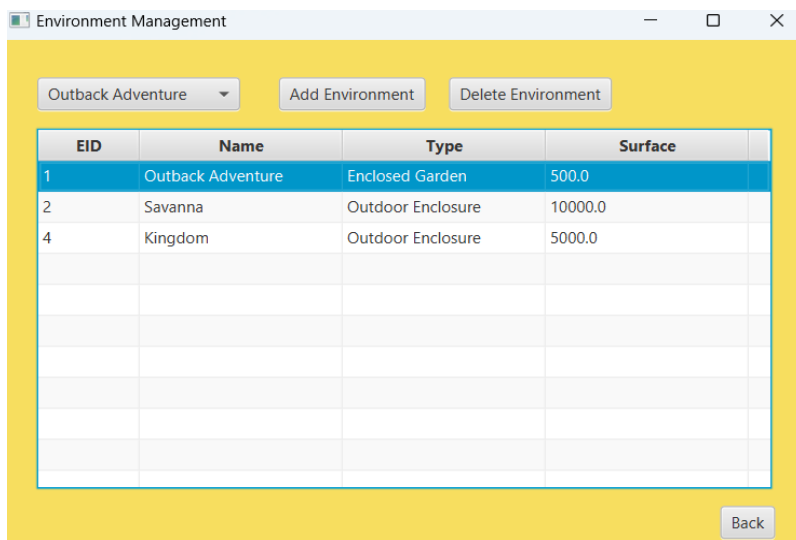


If the user is a worker, it has 2 options:

to manage the environments in which he works (by clicking the first button)

to manage the animals from the environments in which he works (by clicking the second button)

'Back' button goes to the previous window.

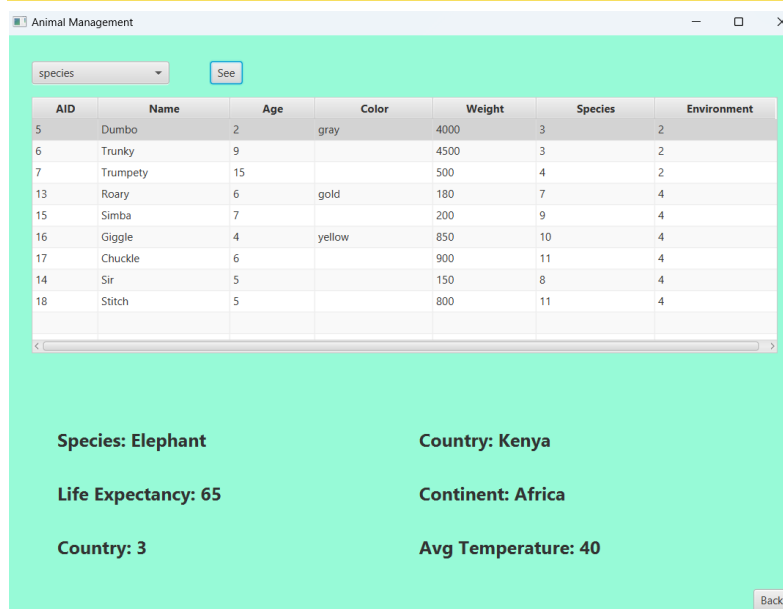


The worker can see the environments in which he works.

He can choose an environment from the ComboBox and add it to his list. ('Add' button)

If he selects an environment from the table, he can delete it. ('Delete' button)

'Back' button goes to the previous window.



The worker can see the animals from the environments in which he works.

He can select an animal from the table, and an option from the ComboBox.

If he presses the 'See' button he can see the selected properties for the selected animal.

If he chooses 'species' he can see the species and the country.

'Back' button goes to the previous window.

Animal Management

environment See

AID	Name	Age	Color	Weight	Species	Environment
5	Dumbo	2	gray	4000	3	2
6	Trunky	9		4500	3	2
7	Trumpety	15		500	4	2
13	Roary	6	gold	180	7	4
15	Simba	7		200	9	4
16	Giggle	4	yellow	850	10	4
17	Chuckle	6		900	11	4
14	Sir	5		150	8	4
18	Stitch	5		800	11	4

Environment: Savanna

Type: Outdoor Enclosure

Surface: 10000.0

Back

If he chooses ‘environment’ he can see the environment in which the selected animal lives.

Animal Management

food See

AID	Name	Age	Color	Weight	Species	Environment
5	Dumbo	2	gray	4000	3	2
6	Trunky	9		4500	3	2
7	Trumpety	15		500	4	2
13	Roary	6	gold	180	7	4
15	Simba	7		200	9	4
16	Giggle	4	yellow	850	10	4
17	Chuckle	6		900	11	4
14	Sir	5		150	8	4
18	Stitch	5		800	11	4

Food: Hay

Type: solid

Quantity: 800

Food: Apples

Type: solid

Quantity: 500

Back

If he chooses ‘food’ he can see the list of foods that the selected animal eats.

HOME

SIGN IN

USERNAME

OillyJ

PASSWORD

.....|

LOGIN

Back

Now, the user is a visitor.  
‘Back’ button goes to the previous window.

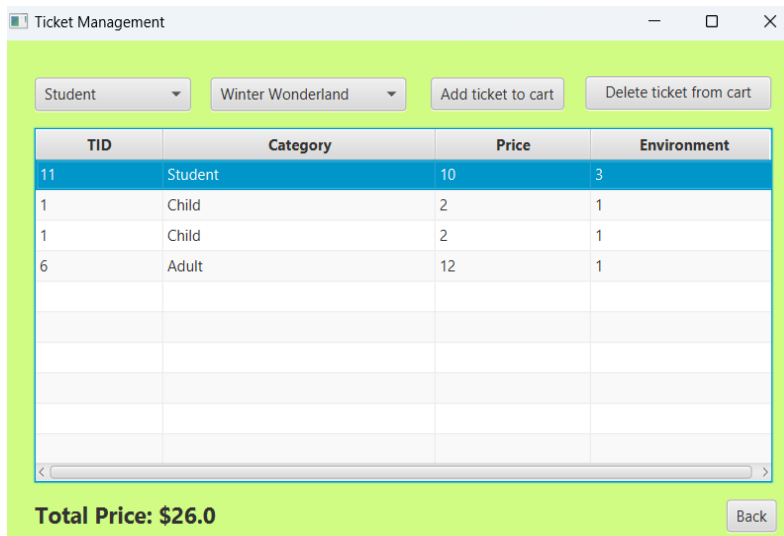


If the user is a visitor, it has 2 options:

to buy a ticket to the zoo (by clicking the first button)

to write a review of his visit (by clicking the second button)

'Back' button goes to the previous window.

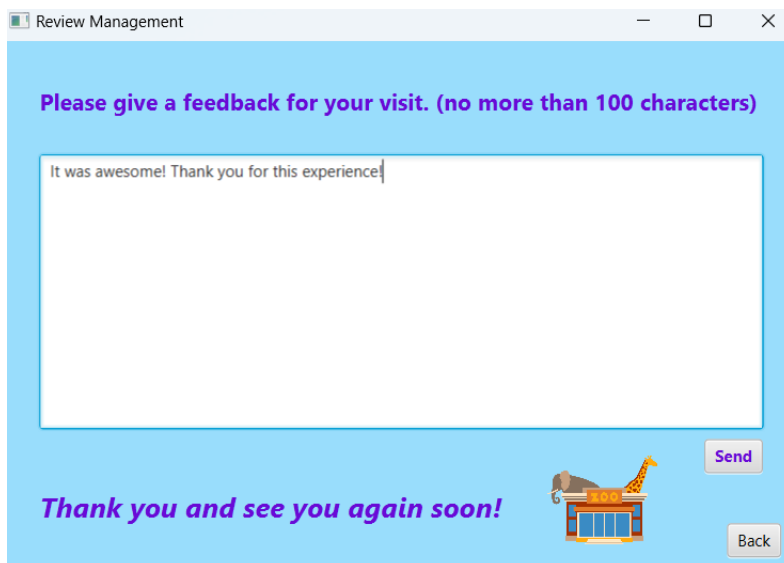


The visitor can see his shopping cart and the total price of the tickets from his cart.

He can choose a category and an environment from the ComboBoxes and buy the ticket with these criteria. ('Add' button)

If he selects a ticket from the table, he can delete it from the cart. ('Delete' button)

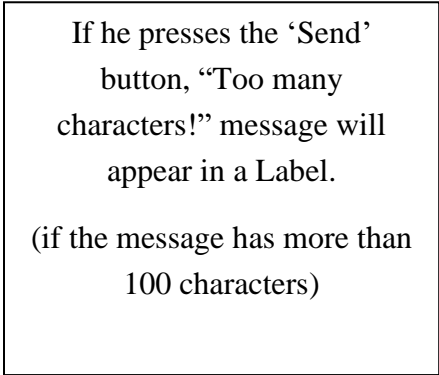
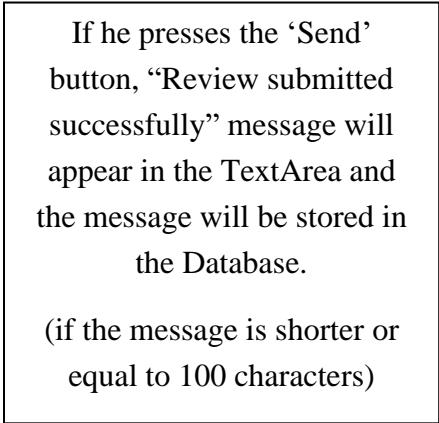
'Back' button goes to the previous window.



The visitor can leave a feedback message.

He can write in the TextArea a text with maximum 100 characters.

'Back' button goes to the previous window.



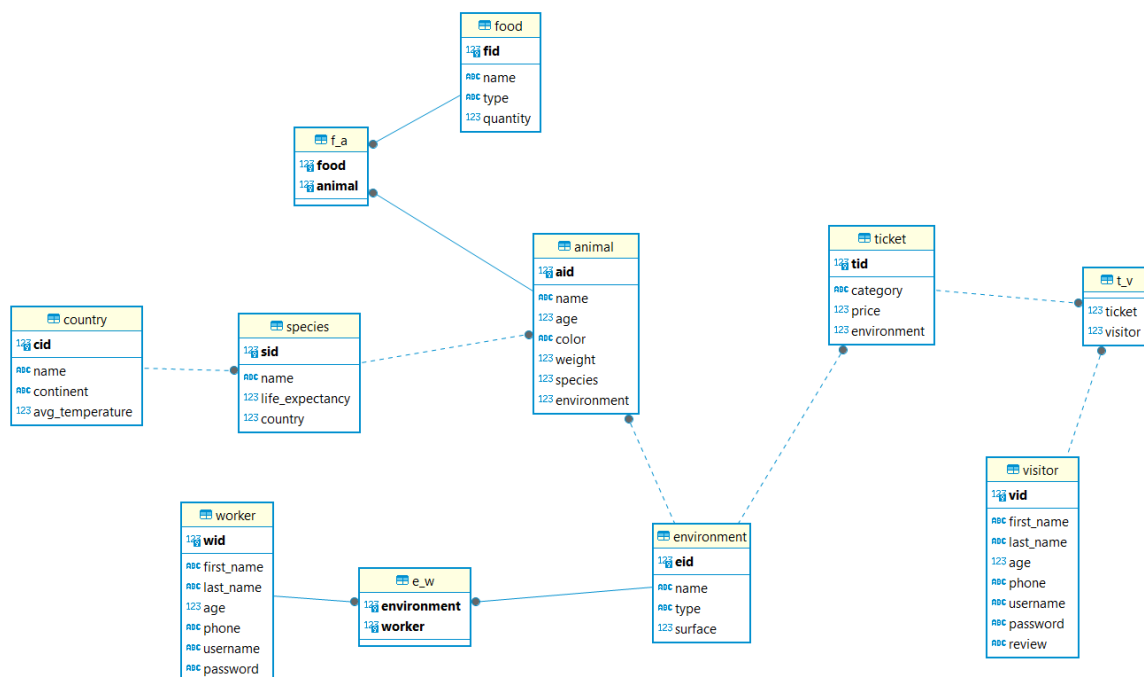
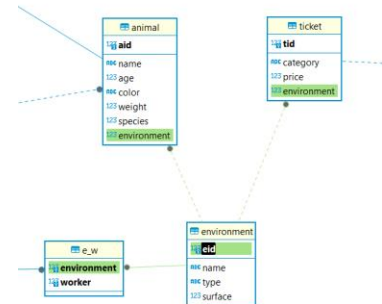
## 2. DATABASE



First of all, I made a Zoo database using PostgreSQL, in DBeaver.

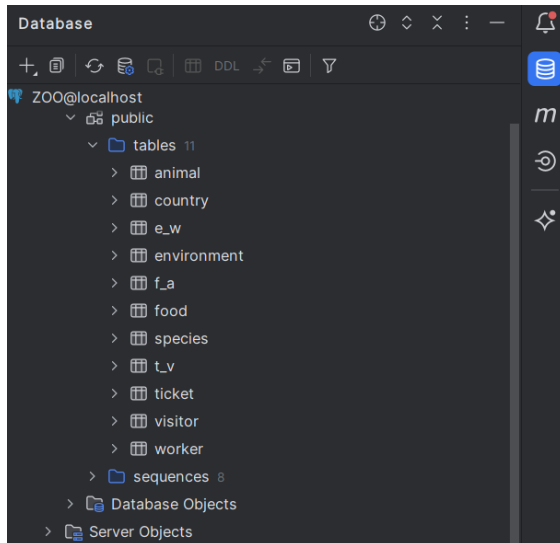
### Components

- 11 tables
- 1-to-many relationships (ex: a country can have many species)
- many-to-many relationships, solved by adding auxiliary tables (ex: table f\_a, a food can be eaten by many animals and an animal can eat many foods)
- constraints:
  - primary keys
  - unique keys (for first\_name & last\_name)
  - check constraint (price < 50)
- foreign keys





After that, I connected the database with my Java application project as New Data Source, using IntelliJ IDEA.



```
1 usage
private Connection establishConnection() {
    try {
        // JDBC URL, username, and password of MySQL server
        String jdbcUrl = "jdbc:postgresql://localhost:5432/ZOO";
        String username = "postgres";
        String password = "postgres";

        // Establish the connection
        return DriverManager.getConnection(jdbcUrl, username, password);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException("Failed to connect to the database");
    }
}
```

## Scripts

In the Java code, I integrated some SQL scripts in order to retrieve, add or delete information from the tables.

- Get all the environments

```
//'e-w' is the join table between 'Worker' and 'Environment'  
String query = "SELECT DISTINCT e.* FROM environment e " +  
               "JOIN e_w ew ON e.eid = ew.environment";
```

- Get the environments for a specific worker

```
String query = "SELECT e.* FROM environment e " +  
               "JOIN e_w ew ON e.eid = ew.environment " +  
               "WHERE ew.worker = ?";
```

- Add environment for a worker

```
String query = "INSERT INTO e_w (environment, worker) VALUES (?, ?)";
```

- Delete environment for a worker

```
String query = "DELETE FROM e_w WHERE environment = ? AND worker = ?";
```

- Get the country for an animal

```
String query = "SELECT c.* FROM country c " +  
               "JOIN species s ON c.cid = s.country " +  
               "JOIN animal a ON a.species = s.sid " +  
               "WHERE a.aid = ?";
```

- Get the ticket by category and environment

```
String query = "SELECT * FROM ticket WHERE category = ? AND environment = ?";
```

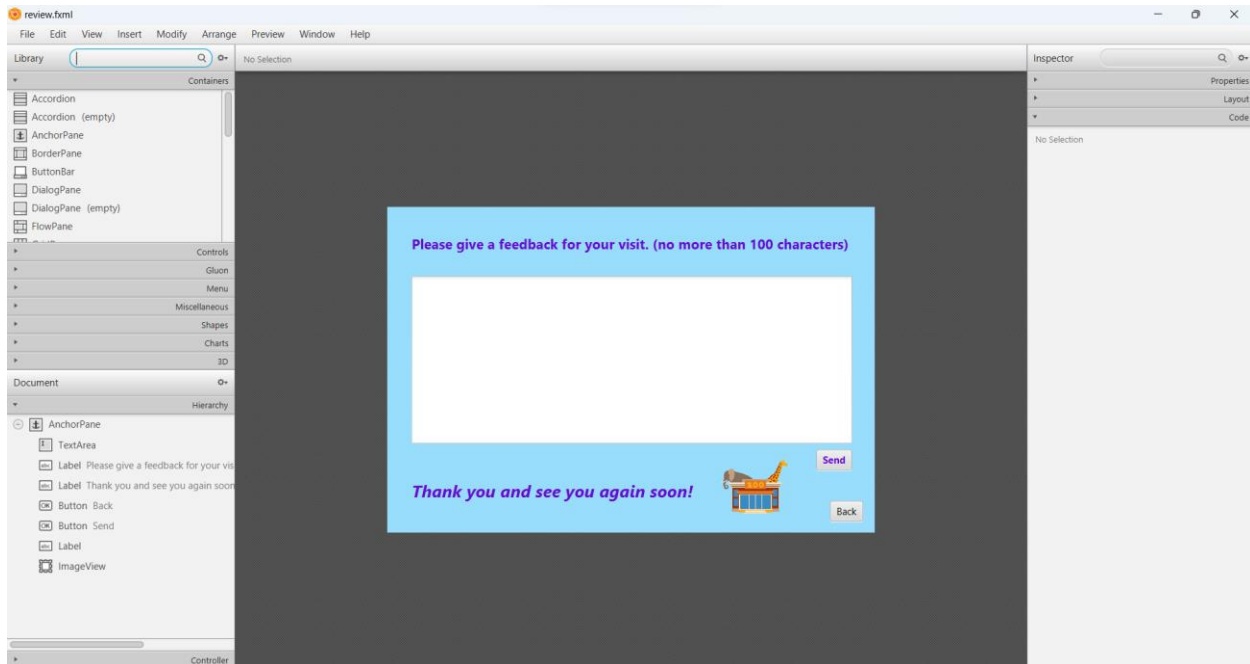
- Update the visitor's review

```
String query = "UPDATE visitor SET review = ? WHERE vid = ?";
```

### 3. JAVA INTERFACE



I used Java logic for creating useful methods and JavaFX & FXML files for making the GUI interface. All the .fxml files were opened and visually customized using SceneBuilder.



### Components

- Classes for tables
  - constructors
  - getters & setters for each column of the table
- Repositories
  - general methods that interact with the database
- Controllers
  - methods that load/interact with the .fxml files
  - methods called when pressing the buttons

## Diagrams

Animal	
Animal(int, String, int, String, int, int, int)	
Animal()	
Animal(String, int, String, int, int, int)	
age	int
species	int
aid	int
name	String
weight	int
environment	int
color	String
name	String
aid	int
environment	int
age	int
color	String
weight	int
species	int

Species	
Species(int, String, int, String)	
Species()	
Species(String, int, String)	
sid	int
name	String
country	String
toString()	String
lifeExpectancy	int
name	String
sid	int
country	String

Ticket	
Ticket(int, String, int, int)	
Ticket(String, int, int)	
Ticket()	
price	int
environment	int
tid	int
category	String
toString()	String
environment	int
category	String
tid	int
price	int

**Classes** have short methods for getting and setting the values for each column of the table. They are very useful for finding information about a class.

(ex: I use 'species.getName()' for easily finding the name of a certain species)

UserInfo	
UserInfo(int, String, String, user_type)	
id	int
userType	user_type
firstName	String
lastName	String
id	int

Environment	
Environment()	
Environment(int, String, String, int)	
Environment(String, String, int)	
type	String
eid	int
name	String
surface	int
toString()	String
name	String
eid	int
type	String
surface	double

Food	
Food()	
Food(int, String, String, int)	
Food(String, String, int)	
quantity	int
type	String
fid	int
name	String
toString()	String
fid	int
name	String
quantity	int
type	String

Country	
Country(String, String, int)	
Country()	
Country(int, String, String, int)	
cid	int
name	String
continent	String
avg_temperature	int
toString()	String
name	String
avg_temperature	int
cid	int
continent	String

t_vRepository	
t_vRepository (Connection)	
deleteTicketForVisitor (int, int)	void
addTicketForVisitor (int, int)	void
getTicketByCategoryAndEnvironment (String, int)	Ticket
getTicketsForVisitor (int)	List<Ticket>
allTickets	List<Ticket>
allEnvironments	List<Environment>

LoginManager	
LoginManager()	
loggedInUser	UserInfo
isValidLogin (String, String)	UserInfo
login (String, String)	boolean
loggedInUser	UserInfo

animalRepository	
animalRepository (Connection)	
getAnimalsForWorker (int)	List<Animal>
getEnvironmentForAnimal (int)	Environment
getSpeciesForAnimal (int)	Species
getCountryForAnimal (int)	Country
getFoodForAnimal (int)	List<Food>

e_wRepository	
e_wRepository (Connection)	
addEnvironmentForWorker (int, int)	void
getEnvironmentsForWorker (int)	List<Environment>
deleteEnvironmentForWorker (int, int)	void
allEnvironments	List<Environment>

**Repositories** have methods that take information from the database and return it in the Java code.

ex: 'isValidLogin()' method gets, using SQL query, the username and password of the user that logged in and searches it in the 'visitor' and 'worker' tables;

if the credentials were found in the 'visitor' table, the method returns a UserInfo (the class previously defined) with 'vid', 'first\_name', 'last\_name' and an enum type 'VISITOR';

if the credentials were found in the 'worker' table, the method returns a UserInfo (the class previously defined) with 'wid', 'first\_name', 'last\_name' and an enum type 'WORKER';

else, if the credentials were not found anywhere, the method returns the id 0, empty strings for the names and the enum type 'INVALID'

SceneController		
SceneController (Stage)		
SceneController ()		
stage	Stage	
switchToVisitor (String, String)	void	
switchToHome (ActionEvent)	void	
onButton2Click (ActionEvent)	void	
switchToReviewManagement()	void	
onButton4Click (ActionEvent)	void	
switchToAnimalManagement ()	void	
switchToEnvironmentManagement ()	void	
switchToHello (ActionEvent)	void	
onLogInClick (ActionEvent)	void	
switchToTicketManagement()	void	
onButton1Click (ActionEvent)	void	
onButton3Click (ActionEvent)	void	
switchToWorker (String, String)	void	
stage	Stage	

TicketController		
TicketController ()		
stage	Stage	
sceneController	SceneController	
establishConnection ()	Connection	
initializeTicketComboBox2 ()	void	
onDeleteTicketClick()	void	
initialize ()	void	
initializeTicketComboBox1 ()	void	
loadTicketsForVisitor (int)	void	
calculateTotalPrice()	void	
onButtonbClick (ActionEvent)	void	
onAddTicketClick()	void	
stage	Stage	
sceneController	SceneController	

ReviewController		
ReviewController ()		
stage	Stage	
sceneController	SceneController	
establishConnection ()	Connection	
onButtonbClick (ActionEvent)	void	
onButtonsendClick (ActionEvent)	void	
updateVisitorReview(int, String)	void	
stage	Stage	
sceneController	SceneController	

Main		
Main()		
start(Stage)	void	
main(String[])	void	

EnvironmentController		
EnvironmentController ()		
sceneController	SceneController	
stage	Stage	
onDeleteEnvironmentClick ()	void	
loadEnvironmentsForWorker (int)	void	
onAddEnvironmentClick ()	void	
establishConnection ()	Connection	
initializeEnvironmentComboBox ()	void	
initialize ()	void	
onButtonbClick (ActionEvent)	void	
stage	Stage	
sceneController	SceneController	

WorkerController		
WorkerController ()		
stage	Stage	
sceneController	SceneController	
initialize (UserInfo)	void	
stage	Stage	
sceneController	SceneController	

VisitorController		
VisitorController ()		
stage	Stage	
sceneController	SceneController	
initialize (UserInfo)	void	
stage	Stage	
sceneController	SceneController	

AnimalController		
AnimalController ()		
sceneController	SceneController	
stage	Stage	
initializeAnimalComboBox ()	void	
onButtonSeeClick ()	void	
establishConnection ()	Connection	
initialize ()	void	
loadAnimalsForWorker (int)	void	
onButtonbClick (ActionEvent)	void	
stage	Stage	
sceneController	SceneController	

**Controllers** have methods written mainly for the FXML interface that use methods from the repositories. The functions' headers are written in the buttons' 'onAction' fields.

They are used for connecting with the database, setting the proper stage, initializing tables, comboBoxes, buttons, labels and switching between different scenes.



## *4. FUTURE DEVELOPMENTS*

By now, I couldn't find an appropriate method to implement a 'Back' button that switches between scenes from different controllers. It gives an error like 'No specified controller'. So, solving this issue would be a future goal.

Also, it could be a good idea to try to make a Sign-Up page and a window that keeps track of the environments of the zoo seen by each visitor.

## *5. REFERENCES*

- Database Lab Works and Courses
- OOP Lab Works and Courses
- Youtube tutorials
- Web resources