CS319 Object Oriented Software Engineering

Draft Design Report

ZooMaster

Instructor: Bora Güngören

Group Members

Ege Berkay Gülcan	21400461
Uğur Can Uyumaz	21301417
Muhammad Hamza Khan	21402884
Kaan Kale	21000912

Table of Contents

1. Introduction

1.1. Purpose of the System

The context of this report is about the initial design of the project. The reports contain an overview of the application which elucidates how the application works, and how to use it. ZooMaster is a desktop based application that assists zookeepers. The purposes of the application are to store the information of animals and plants in the zoo and notify the user to feed the animals and water the plants. In order to show the information of the species, the zookeeper has to add the information of the species into the system because of this it must be user friendly. New species can be added to the system, and the ones who are already in the system can be deleted or edited.

1.2. Design Goals

1.2.1. Ease of Use

Easiness in the usage is one of the most important design goals because it will determine zookeepers continuity to use.

1.2.2. Availability

ZooMaster should be always available, because it will also notify the zookeepers about the feeding times of the inhabitants. Therefore, it should be available for 24 hours in all 7 days of a week.

1.2.3. Low Cost

ZooMaster can be done with small budget because it is a web based application and it does not need any program that needs money.

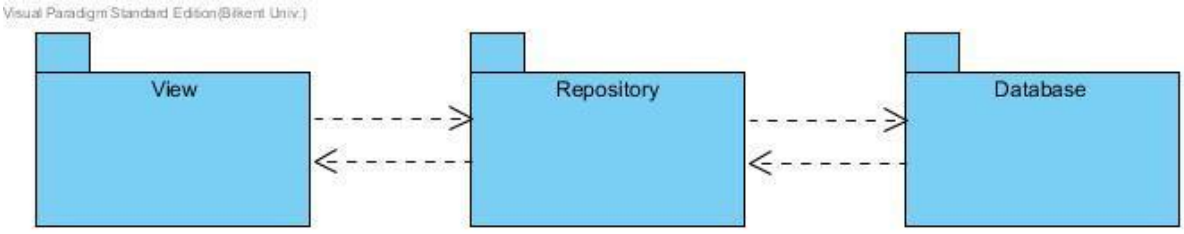
1.2.4. Reliability

ZooMaster is aimed to be bug free in order to prevent from errors or crashes. Crashes may be result with problems for animals like feeding them in correct time interval.

2. Software Architecture

2.1. Subsystem Decomposition

Because of we use mostly data of species which is stored in a database in Zoomaster, The Repository Architecture is chosen to implement the project. In this design pattern, the subsystems are separated based on their common goals. In this way, the design has a readable structure of the project and it makes the project easier to maintain in the future.



The Database subsystem includes the data storing of the project. The data of species will be held in this subsystem. The data can be retrieved with queries or changed within the

repository subsystem. The subsystem has the tables for species and all the data is stored systematically so that this make it faster to access specific information or the whole the data.

The Repository subsystem is the bridge between the user interface and the database, and it includes main functionalities of the project. Firstly, it has access to the database, and structures the data into readable and changeable forms which are sent to view subsystem. It also gets the inappropriate data from the view and forms it into storable data and adding into the database. Other than the data functionality it may also have several other functionalities that adds other features to the system such as security, ease of use, and necessary functionalities for the project.

The View subsystem is the visual part of the project. It includes user interface windows which show the readable data that taken from Repository subsystem, interact with the user by taking new data or changing the current data.

2.2. Hardware/Software Mapping

Zoo Master's final deployment will be in the form of an executable .jar file, which is implemented using Java. We will be using the latest version of Java i.e 8. So, in order to run Zoo Master, the client's machine must have installed at least version 8 of the Java Runtime Environment (JRE).

In order to navigate through the user interface of the application, the client's machine must also have a pointing mechanism such as a mouse or touch a based input device, and a typing mechanism such as a keyboard or a touchpad.

Since, Zoo Master will be a stand-alone application; it won't require an internet connection.

Although, Zoo Master will use mySQL, we will ship the appropriate JDBC driver with the application. So, the user doesn't need to worry about it.

2.3. Persistent Data Management

Our system will use a simple MySQL database to store the information of inhabitants of the zoo. The data base will have 3 tables:

- Species table: In this table we will store the common information of both animals and plants along with a type attribute to determine whether it is a plant or an animal and a foreign key of the animal or plant's ID of its type table.
- Animal table: This table will store the animal specific information of the animals in the zoo such as eating habits, feeding hours, etc.
- Plant table: This table will store the plant specific information of the plants in the zoo such as watering times, lighting times, preferred temperature, etc.

The model in figure X represents the database that has been described above.

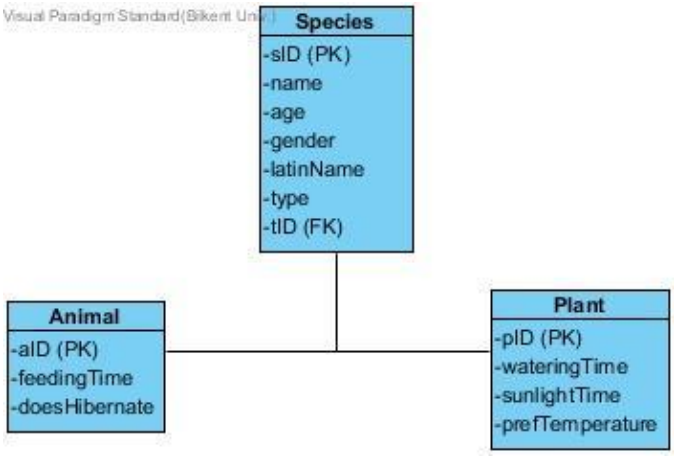


Figure 1: Model for database tables

2.4. Access Control and Security

The application does not require a network connection. However, it does use a password protection service to secure the system from unauthorized users such as wandering or lost guests. After entering the correct password the user could use all of the features of the application. In addition, the system will have an active timer to ensure that the system logs off if the user forgets to log off after using the system.

2.5. Boundary Conditions

2.5.1. Initialization

Zoo Master's final deployment will be in the form of .jar file. So, the only initial condition for it to run will be that the client's machine must have at least version 8 of the Java Runtime Environment (JRE). Furthermore, the user must know the initial password of the application, which will be prompted as soon as the .jar file will be executed.

2.5.2. Termination

Client will be able to close the user interface of the application at anytime by simply clicking on the cross sign in the upper right corner of the application window. However, even after doing so, the application will still be running in the background, in order to give notifications to the user. So, if the user wants to fully terminate the application, then he/she would have to end its process from the taskbar of the operating system he/she will be using.

2.5.3. Failure

If the application has been fully terminated by user and is not running in the background, then ZooMaster will fail to deliver appropriate notifications to the user at the appropriate time.

The application will also fail to perform as intended, if the system date or time is incorrect because ZooMaster heavily relies on it for its notifications feature.

If, somehow, the database of the species gets corrupted then the application will fail as well, and the corrupted species would have to be added again to the database by the user.

We will heavily test ZooMaster before shipping it to the user. So, there won't be any unintended bugs in it.

3. Subsystem Services

The project system has three subsystems that are View, Repository, and Database. In this section functionalities of the subsystems are described in detail.

3.1. Database Subsystem

The subsystem has DatabaseManager class and it is directly connected to MySQL database of the project. The purpose of DatabaseManager is to get the specific data of a species or whole data and transfer them to classes in the repository subsystem. This requires to modify the data after receiving it from the database. In the subsystem, the given data from the repository subsystem is modified by the database subsystem to store it in the database. These modifications are necessary to access, remove, and edit data efficiently in the database. The database of the system is an offline database so that no internet connection is required.

3.2. Repository Subsystem

Repository Subsystem has the many functionalities therefore there are many classes. It has a main class called DataStructure which is connected to the both database and view subclasses and has main interactions between all subsystems. Prime objective of this subsystem which is also our project notify the zookeeper about feeding animals or watering plants. It requires work time for each species. This data is taken from the database subsystem and modify as time table in the weeklyTimeTable class then this information is sent to the view subsystem. Another feature is the getting detailed information about species and store them as an object array during the search. If there is a modification in these species the repository subsystem notifies the database subsystem. The subsystem stores the password with a basic encryption. While the application is open the repository, subsystem is notified if there is a data modification in the view subsystem.

3.3. View Subsystem

View subsystem is the GUI part of the project. It doesn't have any modification or storing feature. Its only aim is to demonstrate the information which is received from the repository subsystem through graphical pages and increase significantly ease of use to the user. It has a MainFrame class which modify or change the page according to user interactions and send the modified data to the repository subsystem. The aim of the subsystem is to have user-friendly interactions.

4.Low-Level Design

4.1 Object Design Trade-Offs

4.1.1 Customizability vs Robustness

The way Zoo-master functions will be fixed, and the zoo keepers won't have the option to be able to change its functionality through a settings option. It's designed this way, keeping in view that the zoo keepers won't have enough technical knowledge and mingling with settings can introduce errors into the application. So, by limiting the customizability of the application and not having a settings option, we will make sure that the application performs only in the predefined, intended way.

4.1.2 Memory vs. Performance

Since, storage space is not going to be an issue on our client computers, because they won't have a lot many of other applications installed in their work stations, we will focus more on the speed of the application during its implementation.

4.1.3 Simplicity vs Number of Options

We have designed zoo-master in a minimalistic way by only having the most necessary interactions in the user interface. This is to avoid the zoo keeper from getting confused by looking at a lot many of intractable options in the UI.

4.2 Final Object Design

Through this section, the design patterns which are used in the project will be demonstrated and the reasons why these patterns are chosen will be explained. The UML diagram has changed since the analysis report due to compatibility of design patterns. New UML can be seen in figure 3.