CS319 Object Oriented Software Engineering

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

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

Prior to the description of our system, we should first identify our design goals of the system. These goals are derived from the nonfunctional requirements that we currently identified.

1.2.1. End User Criteria

Usability: Since not all users are computer experts, the system will be easy to use by all kind of users. To achieve this the system will have a simple and user friendly interfaces that enables users to find and use the features quickly and easily. Also the system will clearly identify what it needs when user is prompted for input.

1.2.2. Performance Criteria

Memory: Since the system is a zoo management application, it will store and process the data of the inhabitant animals and plants, and it requires storage space and memory to do this.

1.2.3. Dependability Criteria

Availability: ZooMaster will be available 100% of the time, because it will also notify the zoo keepers about events like feeding or watering times of the inhabitants.

Security: The system will be password protected to prevent any authorized users to make changes in the data. The system will require the users to enter a password to start a session before doing anything and will have a timer that causes the system to logout when the system is inactive to prevent security threats due to user faults such as forgetting to logout after the use is over.

1.2.4. Maintenance Criteria

Portability: The system does not have a platform constraint and therefore it will be available in a wide range of platforms. It will achieve this by its usage of multiplatform software and programming languages. The implementation of the system will be made by Java language which through the Java Virtual Machine (JVM). The database will use MySQL Database which also supports a wide range of platforms.

Extensibility: Since it is possible for a zoo to change it structure, e.g. adding aquariums, the system will also be able to adopt new features to meet new demands. To achieve this, the system will be designed to support the addition of new entities and functionalities.

1.2.5. Cost Criteria

Development Cost: To reduce the cost of development, the system will use open source software such as MySQL Database.

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.

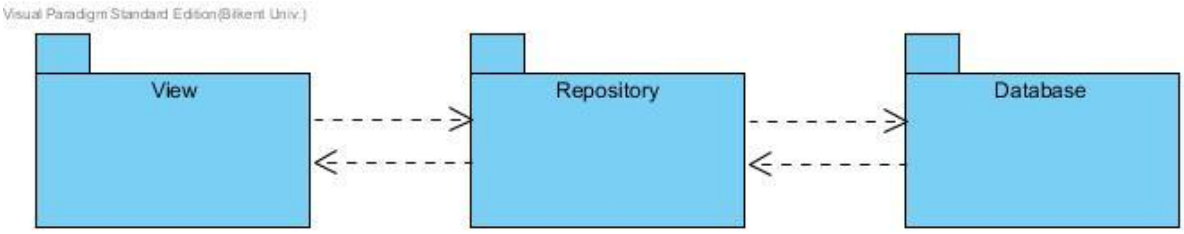

Figure 1: Subsystem Decomposition Model

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.

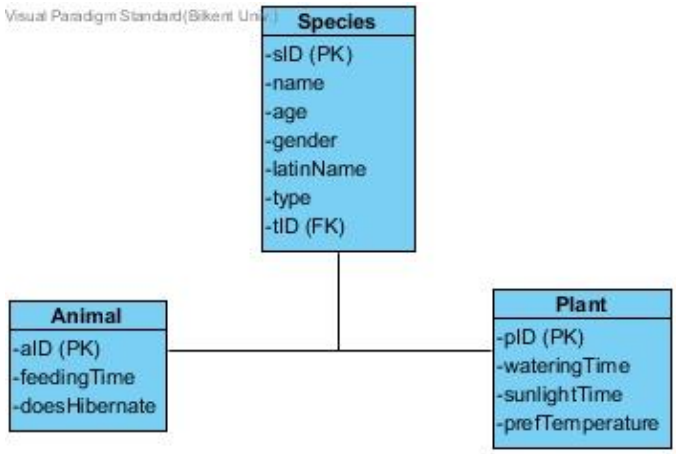

Figure 2: Model for database tables

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. Trade Offs

4.1.1. Customizability vs Robustness

The way ZooMaster 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.

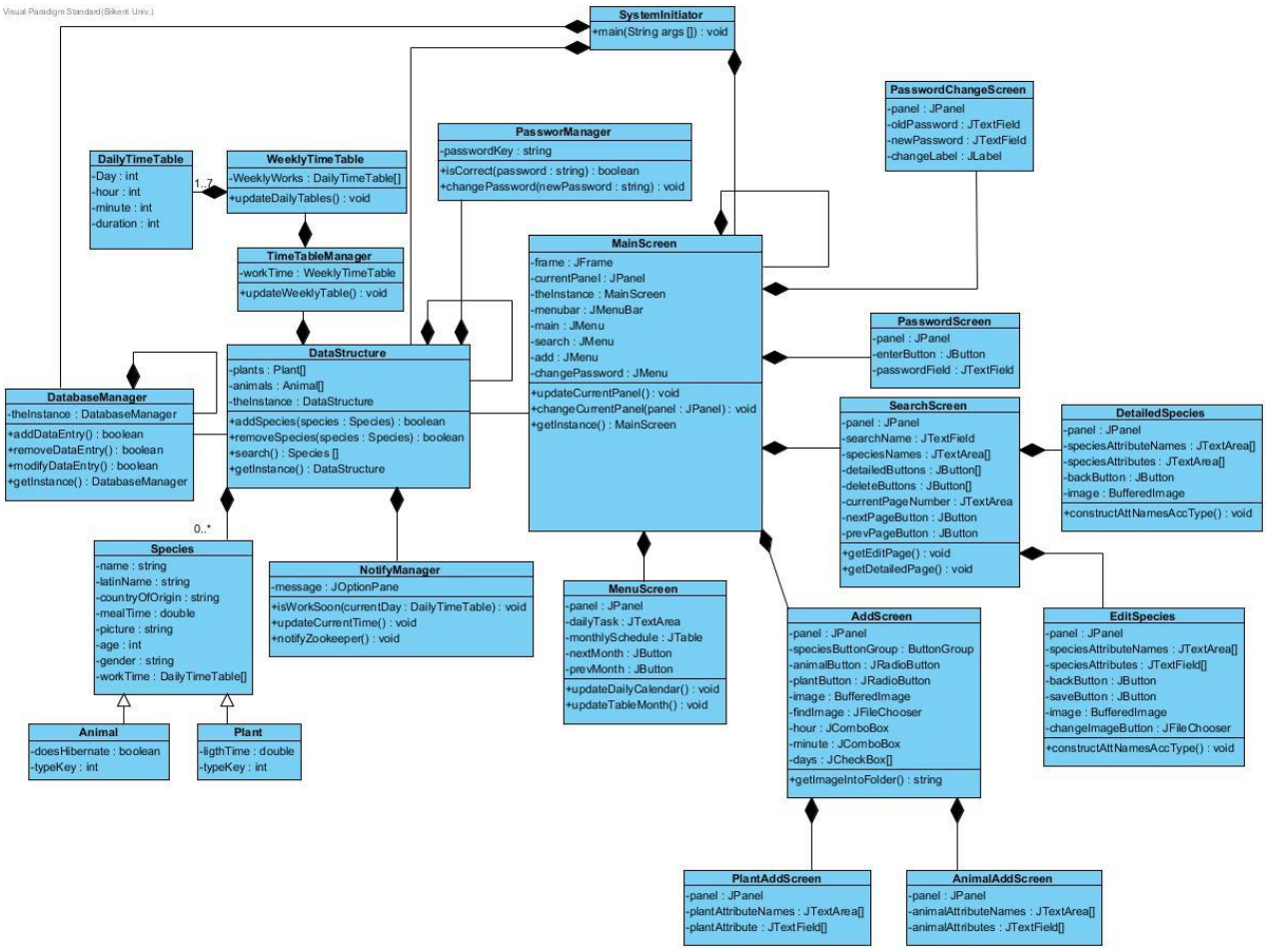

Figure 3: Object Design Model

4.2.1. Façade Pattern

The project is built using repository pattern as an architectural pattern and data transactions are essential for the software and with façade design transactions between subsystems are consistent and systematical. Without using a good pattern modifying the subsystems or adding new subsystems into the software would be catastrophic. Thus with façade pattern the subsystems are reusable and maintainable.

In the project there are classes that are top hierarchy class in their subsystems and the subsystems are connected through them. These classes can be seen in the figure. MainScreen class is top of view, DataStructure class is top of repository and DatabaseManager is top of database.

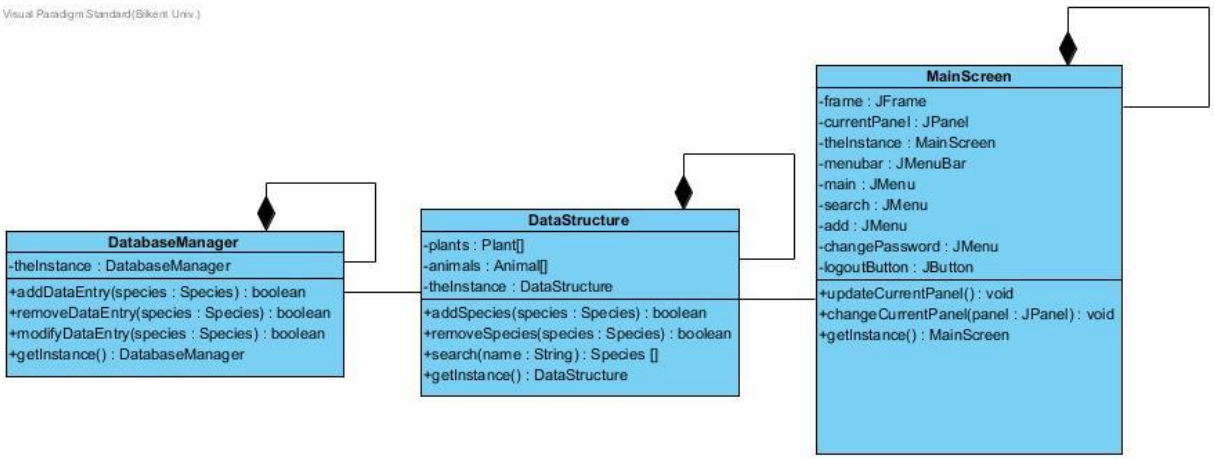

Figure 4: Façade Pattern

4.2.2. Singleton Pattern

Singleton pattern is mostly used if a class is going to be used only as one object. Since in the façade pattern there are top hierarchy classes in the subsystems and these are initialized in main only one singleton pattern is useful for the project. The communication between classes is done by using getInstance() method which can be found in all of these top hierarchy classes.

4.3. Packages

Through the section, package diagrams will be examined in detail. Class relationships, analyze of subsystems by using classes, their attributes and methods will be demonstrated.

4.3.1. View Package

View package has all GUI related classes in the software and it is designed by applying façade pattern. MainScreen class has control over all of the screen classes and only MainScreen has connection between repository package in view package. Because of only MainScreen class has connection with Repository Package, it is also designed by using singleton pattern. All screens have mostly different attributes and methods for their functionality. PlantAddScreen and AnimalAddScreen classes are subclass of AddScreen classes because their functionalities are similar to each other. MainScreen does changing screen and the data transactions between screens which taken from the user and the repository which gets the data from the database subsystem.

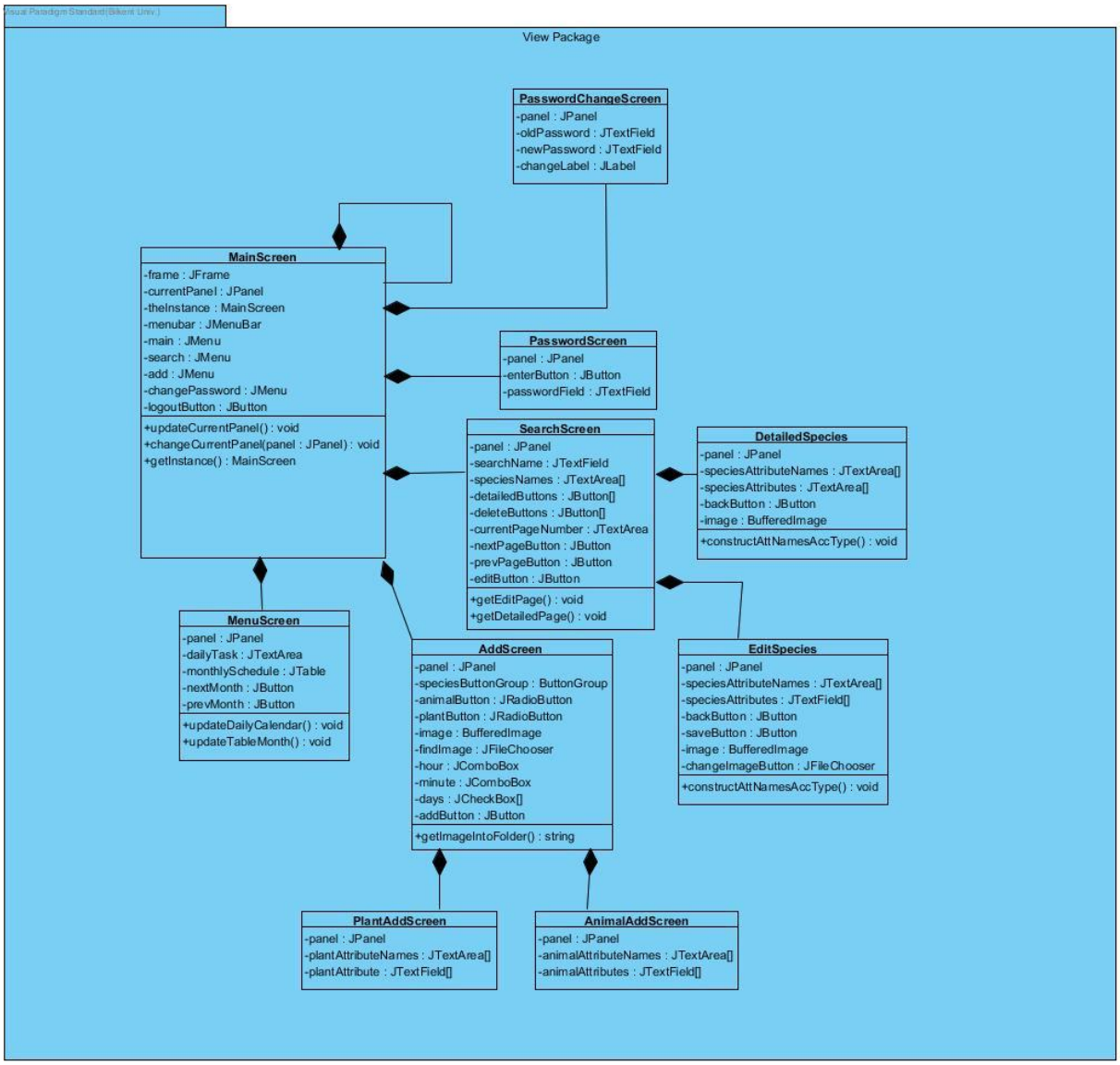

Figure 5: View Package

4.3.2. Repository Package

Repository package has all data processing related classes in the software and it is also designed by applying façade pattern. DataStructure class has control over several subsystems and it is the only class interacts with View and Database packages so that it is also designed by using

singleton pattern. TimeTableManager class is responsible for storing and updating the time tables of species work. NotifyManager class is responsible for notifying the zookeeper if any work time is soon. DataStructure also responsible for shaping the data which come from database package into a suitable way by using Species class and its subclasses Animal and Plant. PasswordManager is responsible for storing and changing the password and keep the password safe with a basic encryption.

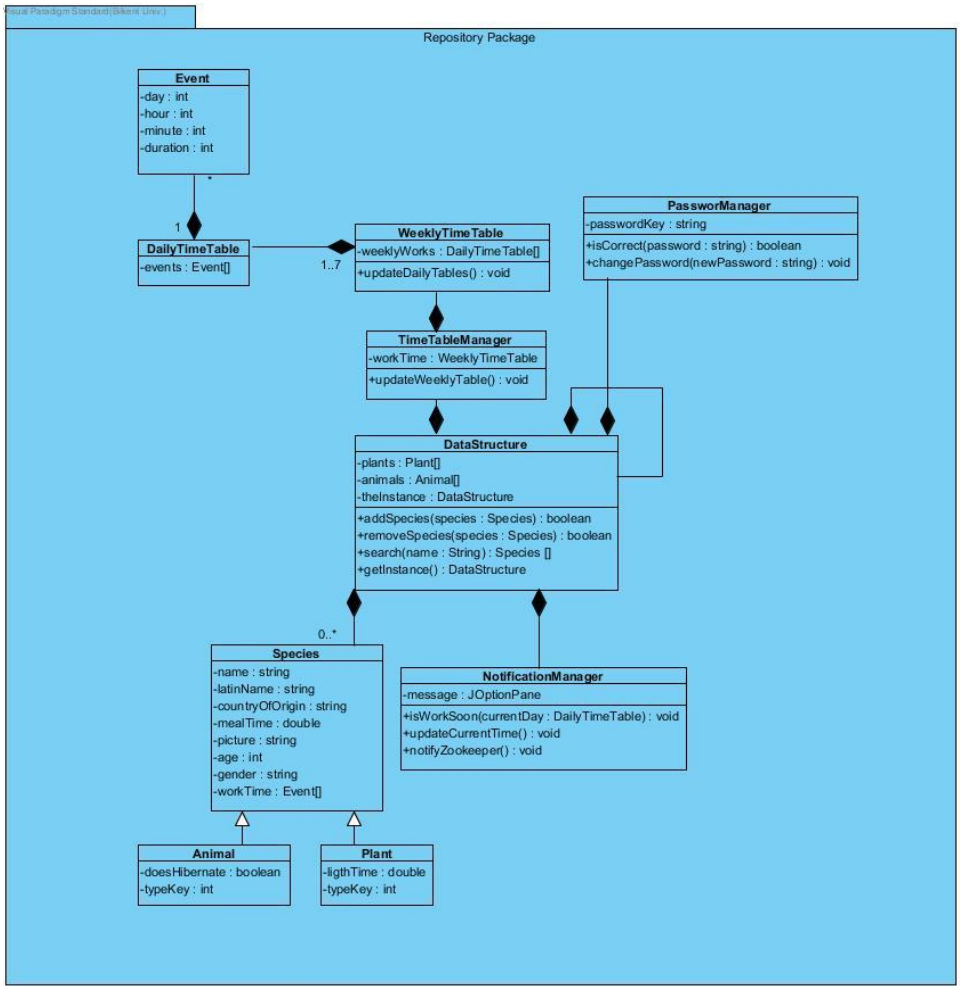

Figure 6: Repository Package

4.3.3. Database Package

Database Package is only package that has connection directly to the database server. The package is designed by using both façade and singleton patterns. DatabaseManager also has

connection with Repository package with rules of façade pattern. DatabaseManager class retrieve, modify, remove and add the information in the database with the interactions which are sent from Repository package. The class also shapes the species data come from the repository package into suitable data for storing in the database system.

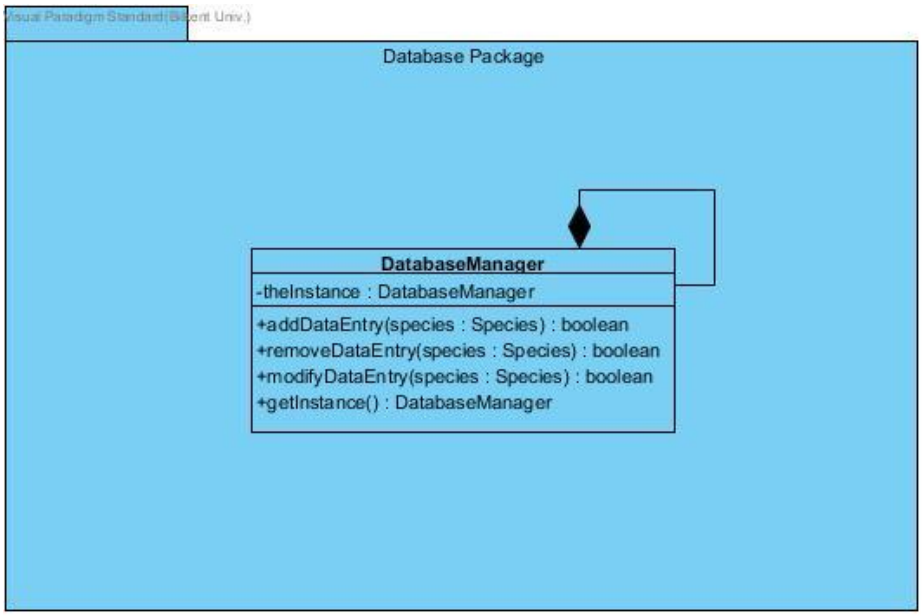

Figure 7: Database Package

4.4. Class Interfaces

4.4.1. MainScreen Class

Once the identity of the user has been identified through the password page, the user will be landed on to the MainScreen. This will be the starting point of almost all subsequent actions of the user.

Attributes

private JFrame frame: It will be the main container of the application. It will hold most of the other containers and components of the application. Even when the screens will change from

MainScreen to SearchScreen or AddScreen, the instance of this frame will remain intact. It will only be the panels that will change within this frame.

private JPanel currentPanel: This container will hold the monthly calendar and the daily task schedule. The panel itself will be held by the frame, container. It will be a transparent panel, so that the wallpaper beneath the panel will be visible. Any opaqueness will be created by the calendar and task scheduler in the panel.

private JMenuBar menubar: Thiswill be the container to hold the Menus of the application. This menubar will be visible throughout the execution of the application.

JMenu instances of it will be used to switch between the screens. For example, clicking on the search instance of JMenu will take the user to the search screen.

Methods

public void updateCurrentPanel(): Itwill be used to update the contents the panel, whenever required. This method will first check that which panel is currently being viewed by the user and then update the contents of only that panel.

public void changeCurrentPanel(): Thiswill be called by the action listener of the instance of the JMenu. This method will first check that which instance of JMenu called it, and then it will accordingly change the visibility of the panels to only make the required panel be visible to the user.

4.4.2. AddScreen, PlantAddScreen, AnimalAddScreen Classes

AddScreen will be the parent of PlantAddScreen and AnimalAddScreen. An inheritance relationship will be used. So, PlantAddScreen and AnimalAddScreen will have access to all of the attributes and methods of AddScreen. This will be done to avoid redundancy and to put all of the common attributes and methods of PlantAddScreen and AnimalAddScreen into the AddScreen.

Attributes

private JPanel panel: This JPanelwill be the container to hold all of the other components of the AddScreen. The panel itself will be held by the frame of the instance of MainScreen.

private ButtonGroup speciesButtonGroup: Itwill be used to group the radio buttons, animalButton and plantButton. It will be used to switch among the animal and plant panels. The switching will be done by changing the visibility of the appropriate panels.

Private JRadioButton animalButton: This radio button,when selected will make sure that it is the animal panel being displayed to the user.

private JRadioButton plantButton: This radio button,when selected will make sure that it is the plant panel being displayed to the user.

private BufferedImage image: Thiswill contain the picture of the species being added. It will be selected by the user.

private FileChooser findImage: Itwill be used by the user to browse through his/her computer and select the image to be displayed in the Image field.

JComboBox instances will be used to select the feeding time of the animal or the watering time for the plant.

JTextArea will be used as an input field of any additional notes by the user.

JTextField instances of this will be used as inputs of small data such as name and origin country of the species.

private JButton addButton: This buttonwill be pressed by the user after he/she has finalized all of the information. The action listener of this button will initiate the saving of all this data to the

database, and clearing all of the fields of this screen to make it ready for the addition of another species.

Methods

public String getImageIntoFolder(): Thiswill be called by the action listener of Select Image and will be used to get the image of the species from the user's files into the application. So, that even though, user might remove the picture from his/her files, it will be displayed into the application without any problem.

4.4.3. PasswordChangeScreen Class

This screen will be viewed to the user as soon as the user clicks on the PasswordChange Menu in the MenuBar. This screen will be held in the frame in the mainScreen and will be viewed to the user by changing the visibility of its JPanel to true and by changing the visibility of the rest of the panels to false.

Attributes

private JPanel panel: This panelwill be the main container of this screen. It will hold the rest of the components in it.

private JTextField oldPassword: Itwill be used as an input field to take in the old password from the user.

private JTextField newPassword: This fieldwill be used an in input field to take in the new password from the user.

private JLabel changeLabel: Itwill be used to take in the new label.

4.4.4. PasswordScreen Class

This will be the entry point to the application. As soon, as the user starts the application, this screen will pop up. And unless, the user enters the correct password, the user won't be allowed to access the rest of the application.

Attributes

private JPanel panel: Thiswill be the main container for this screen. It will hold the rest of the components.

private JButton enterButton: This buttonwill have the action listener that upon being activated will check the password typed into the passwordField.

private JTextField passwordField: Thiswill be used as an input component to take in the password from the user.

4.4.5. SearchScreen Class

This screen will be viewed to the user as soon as the user clicks on the Search Menu in the MenuBar. This screen will be held in the frame in the mainScreen and will be viewed to the user by changing the visibility of its JPanel to true and by changing the visibility of the rest of the panels to false. The screen itself will have features to be able to search the database of the species and modify them.

Attributes

private JPanel panel: Thiswill be the main container for this screen. It will hold the rest of the components.

private JTextField searchName: Thiswill be used to take the input search string from the user. As soon as the user types something into this field, the application will use that string to search through the database and update the result field.

private JTextArea[] speciesName: Thiswill be an array, used to output the search result from the database. The search result will be dependent on the input string typed into the searchName: JTextField by the user.

private JButton[] detailedButtons: Thiswill be an array to hold all of the detailed buttons, each with one output speciesName. This button will be used to display the detailScreen of the particular species.

private JButton[] deleteButton: Thiswill be an array to hold all of the delete buttons, each with one output speciesName. This button will be used to delete the specific species from the database.

private JTextArea currenPageNumber: Thiswill be used to output the current search page the user is on.

private JButton nextPageButton: Thiswill be used to display the next search page to the user.

private JButton previousPageButton: Thiswill be used to display the previous search page to the user.

private JButton editButton: Thiswill have the action listener to instantiate and open up the edit panel.

Methods

public void getEditPage(): Thiswill display the edit page to the user. The method will be called by the action listener of the editButton.

public void getDetailedPage(): void will display the detailed page to the user. The method will be called by the action listener of the detailedButton.

4.4.6. DetailedSpecies Class

This panel will pop up to the user upon pressing the detailedButton on the searchScreen. This panel will include the detailed information of particular species.

Attributes

private JPanel panel: Thiswill be the main container for this screen. It will hold the rest of the components.

JTextArea instances of this component will be used to display the corresponding information to the user about the species.

private JButton backButton: Thiswill close the detailedSpecies screen and go back to the previous screen which will be searchScreen.

private BufferedImage image: Thiswill be used to display the image of the species from the database to the user.

4.4.7. EditSpecie Class

This panel will be instantiated in the SearchScreen and will hold the information about particular species. This information will be editable and upon exiting will be used to update the database.

Attributes

private JPanel panel: Thiswill be the main container for this screen. It will hold the rest of the components.

JTextArea instances of this component will be used to display the corresponding information to the user about the species.

JTextField instances of this component will be used to display the corresponding information to the user about the species.

private JButton backButton: Thiswill close the detailedSpecies screen and go back to the previous screen which will be searchScreen.

private BuffereImage image: Thiswill be used to display the image of the species from the database to the user.

private JButton saveButton: Thiswill be used to update the database in accordance with all of the information from the components of this screen.

private JFileChooser changeImageButton: Thiswill open up the file browser which can be used by the user to browse through his/her files and replace the current image of the species.

4.4.8. SystemInitiator Class

This class is the main class of the system. It will initiate the application and create necessary objects and start processes.

Methods

public static void main(String args[]): This is the main method of the application.

4.4.9. PasswordManager Class

This class handles password authentication process. Based on the input that it receives it authorizes sessions.

Attributes

private String passwordKey: This string stores the password key that is needed to log in to the system.

Methods

public boolean isCorrect(password : String): This returns true if the inputted password is correct. Otherwise, returns false.

public void changePassword(newPassword :String): This function changes the old password with the newPassword parameter.

4.4.10. Event Class

This class is an event that will happen during a day such as feeding time for zebras or watering time for petunias.

Attributes

private int day: This stores the value for which day that this event occurs.

private int hour: This stores in which hour of the day that the event will occur.

private int minute: This stores in which minute of the hour that the event will occur.

private int duration: This stores the duration of the event.

4.4.11. DailyTimeTable Class

This class manages and stores the events that will happen during that day.

Attributes

private Event[] events: This array stores the events that is associated with that day.

4.4.12. WeeklyTimeTable Class

This class manages the daily time tables and creates a weekly time table for the zookeepers.

Attributes

private DailyTimeTable[] weeklyWorks: This array stores 7 DailyTimeTable classes which togerther create a weekly time table.

Methods

publicvoid updateDailyTables(): This methods updates the daily tables and creates a new weekly table.

4.4.13. TimeTableManager Class

This class manages the time tables of events and is responsible for their updates.

Attributes

private WeeklyTimeTable worktime: This is the weekly time table object for the current week.

Methods

publicvoid updateWeeklyTable(): This method updates the weekly time table at the end of the week.

4.4.14. NotificationManager Class

The NotificationManager class handles the notifiations of events. It notifies the zookeepers when the the time for events are close.

Attributes

private JoptionPane message: This displays a message about the event when the event is close by.

Methods

public void isWorkSoon(currentDay: DailyTimeTable): Checks if the work associated with the daily events will happen soon.

public void updateCurrentTime(): Updates the current time if it is not syncronized with the OS' clock.

public void notifyZookeeper(): Displays notificaiton to inform the zoo keeper about the upcoming event.

4.4.15. DataStructure Class

This class is a data structure that holds the species entries to achieve fast search capabilites. The data structure to be used is not decided yet. It will be determined during the implementation stage.

Attributes

privatePlant[] plants: This array holds the plant entries in the data structure.

private Animal[] animals: This array holds the animal entries in the data structure.

private DataStructure theInstance: This is the data structure instance.

Methods

public boolean addSpecies(species : Species): This method adds a species entry to the data structure. If the addition has been successfully done, returns true, else returns false.

public boolean removeSpecies(species : Species): This method removes species entry from the data structure. Returns true if the removal has been successfully done, else returns false.

public Species[] search(name : String): Searches the entries and returns the matching entries as an array of species.

public DataStructure getInstance(): Getter method for theInstance.

4.4.16. Species, Animal and Plant Classes

These are container classes for species informations. Animal and Plant classes inherit from species class

Attributes

private String name: Name of the species.

private String latinName: Latin name of the species.

private String countryOfOrigin: Country of origin of the species.

private double mealTime: Meal time for the species.

private String picture: Image url for the species.

private int age: Age of the species.

private String gender: Gender of the species.

private Event[] workTime: Work times for the species.

4.4.17. DatabaseManager Class

This is the manager class for the database. It queries the database for information and inserts/deletes tuples from the database.

Attributes

private DatabaseManager theInstance: It is the instance of the database manager.

Methods

public boolean addDataEntry(species : Species): Inserts data entry to the database.

public boolean removeDataEntry(species : Species): Deletes data entry from the database.

public boolean modifyDataEntry(species : Species): Updates the attributes of a tuple.

public DataBaseManager getInstance(): Getter method for theInstance attribute.