Dmytro Dzhuha

All In Auction

In my project, I've implemented English type of auction. First, the user must register/login. A user can be a seller or a buyer. There will also be administrators who control the lots, the auction process and users. When creating a lot, the seller can completely customize everything. To avoid auction disruption and fraud, users will need to top up their balance using a form of payment and if they have enough money, they will be admitted to the auction. There is also a notification system to inform users about anything: selling a slot, buying a slot, etc.

Documentation content

Fulfillment of criteria

- Encapsulation
- Patterns
  - Adapter
  - Singletone
- Own exceptions and fix of it
  - DataBaseException, UserException
- MultiThreading for creating bots
- RTTI
- Lambda
- Nested class and interface
- Interface default method
- Seriallization

Certain implementations

- Polymorphism
- Inheritance
- Interface
  - DataBase.java

Main criteria

- Polymorphism
  - Seller.java #11
  - Bidder.java #11
  - Admin.java #11
- Inheritance
  - Seller/Bidder/Admin.java #6
- Interface
  - DataBase.java

Secondary criteria

- Encapsulation
  - Every class
- Patterns
  - Adapter | ItemAdapter.java | NotificationsAdapater.java
  - Singletone | Database.java | SceneController.java
- Own exceptions and fix of it
  - DataBaseException, UserException
  - RegistrationController.java #170
- MultiThreading
  - Main.java #79-81
- RTTI
  - LoginController.java #111
- Lambda
  - Main.java #79
  - User.java #143
- Nested class and interface
  - User.java # 26
- Interface default method
  - UserInterface.java #12
- Seriallization
  - UtilController.java #15 #27

Project documentation

List of features for current version v0.0.8

- Admin side
- Bidder side
- Seller side
- Database
- SceneController
- Notification system
- Bots system

Important code

- DataBase singletone
- User class seriallization
- Auction bot system

Versions

Project contains the following fully functional versions:

- v0.0.1
- v0.0.2
- v0.0.4
- v0.0.5
- v0.0.6
- v0.0.7
- v0.0.8

Change log

Version 0.0.1

Added:

- Start menu
- DataBase connection
- Controllers
- Project structure

Version 0.0.2

Added:

- Registration
- Authorization
- Main test window(UI)

Version 0.0.3

Added:

- Scene Controller bug fixes, final implimentation

Version 0.0.4

Added:

- New SceneController
- Update main User class and it's childs
- Admin/Seller/Bidder UI (Preview)
- Admin/Seller/Bidder controllers
- User serialization
- Utils controller

Version 0.0.5

Added:

- History log about bids Admin/Seller/Bidder side
- New UI
- Item adapter

Version 0.0.6

Added:

- Admin side
    - Home page
    - Bids history
    - Accept bid menu
    - Decline bid menu

    - Create category menu

- Seller side

    - Withdrawal menu
    - Home page
    - Sell item menu
    - Selled items history

- New UI

- Custom Message Box and Dialog selector
- Dialog/Info box controller
- Lambda
- RTTI

Version 0.0.7

Added:

- Admin side

    - Fully completed
    - Ban user

- Bidder side

    - Button "More"
    - Notifications
    - Update balance
    - Notifications

- Seller side

    - Notifications

- InfoBox

    - new Dialog handler
    - new MessageBox handler

- Adaptive InfoBox'es design

- New top-menu UI/UX
- Notifications
- Notifications adapter

Version 0.0.8

Added: - Bidder side - Auction system

- Auction

    - Bot system
    - Bid system
    - Winner/Owner notifications
    - UI

- UI

    - Fixed .css file errors
    - New UI

- JavaDoc

    - Generated comments for JavaDoc

Technical details

Environment setup

- Eclipse Java EE IDE for Web Developers, version: Oxygen.3a Release (4.7.3a) Build id: 20180405-1200

- Intellij 2020

- JDK 17

- JavaFX 17.0.1
- Scene Builder
- MySQL Connector/J 8.0.23

Installation

- Update MySQL connection info in DataBase.java
- If you're using Intellij, all libraries will be automaticly installed by Gradle
- To run project in Eclipse, you need manualy install JavaFX library and Mysql JDBC Driver to your project

Compilation

- To compile project in Eclipse you need to set this as a VM arguments
  - `--module-path "lib" --add-modules javafx.controls,javafx.fxml`
  - `lib - path to JavaFX libary, for example - C:\Users\Admin\Desktop\JavaFX\lib`
- If you're using Eclipse, make sure that `PATH_STATE` in `SceneController.java` is set to `true`

Running .jar file

- First of all, in Command Line you need to open folder where .jar file located with `cd` command
- To run .jar file you need to start program via Command Line with this arguments
  - `java --module-path "lib" --add-modules javafx.controls,javafx.fxml -jar name.jar`
  - Where `lib` - path to JavaFX libary, for example `C:\Users\Admin\Desktop\JavaFX\lib`
  - Where `name` - version name, for example - `v0.0.4`

After all, you can use programm. To login as Admin use login 0xAdmin and password admin

Database

- MySQL

To connect program to your database server, you need to change connections settings in `DataBase.java` at lines 31-43 If you are running .jar file, you will automaticly connect to my database server, so you don't need to recompile your program

Simulation and demonstration

Video demonstration

- Video demonstration of my project
  - Video

UML diagrams

Project contains the following diagrams:

- UML diagram
  - Class diagram
  - Description of classes
  - DataBase diagram

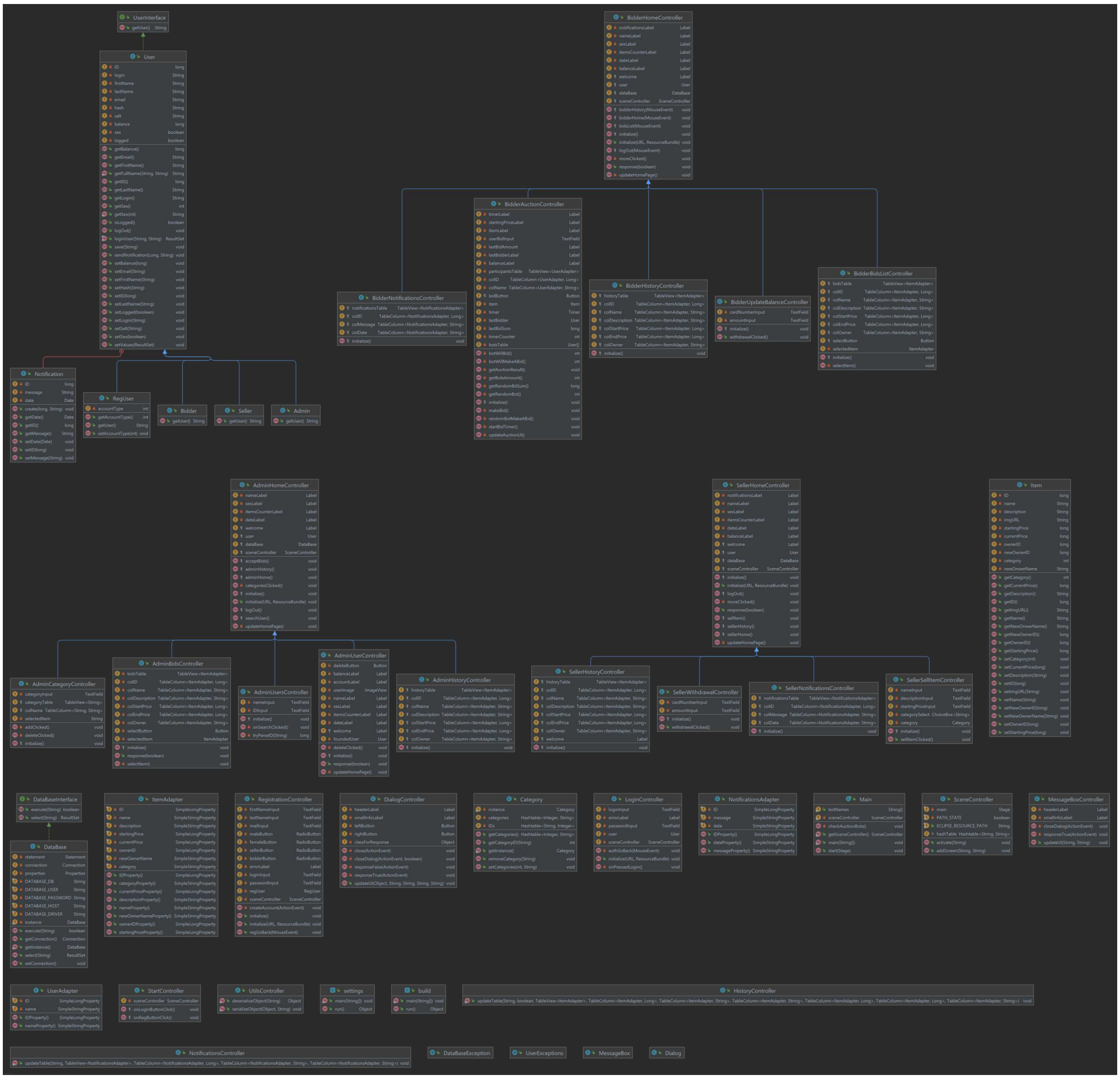
Classes

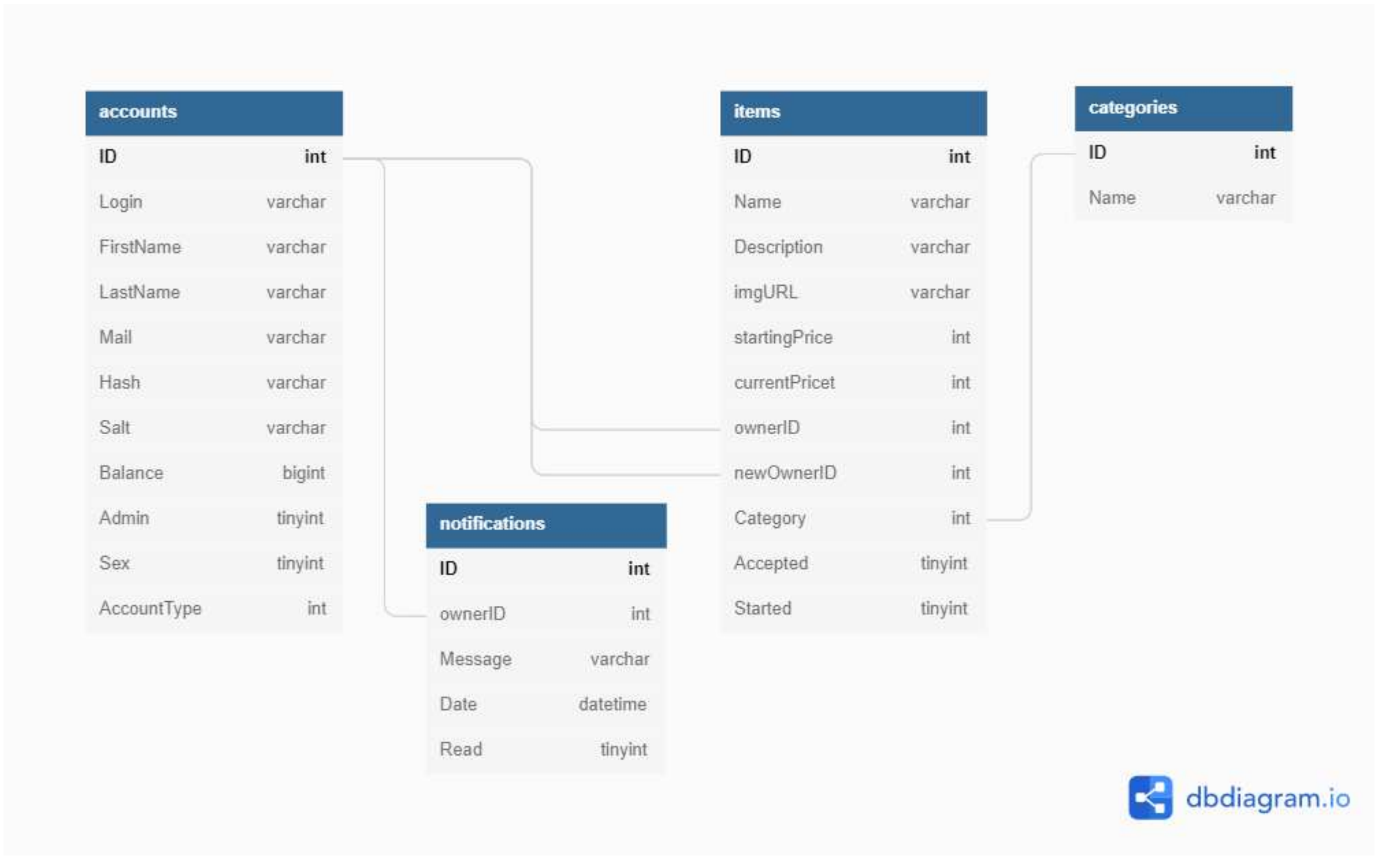
Description of classes

- Adapters
  - `ItemAdapter` - used to convert Item data to data which suits to TableView
  - `NotificationsAdapter` - used to convert Notifications data to data which suits to TableView
  - `UserAdapter` - used to convert User data to data which suits to TableView
- Item classes
  - `Category` - used to deal with all categories
  - `Item` - describes item properties
- InfoBox classes
  - `Dialog` - used to call custom Dialog window
  - `MessageBox` - used to call custom MessageBox window
  - `MessageBoxController` - controller of Dialog UI
  - `DialogController` - controller of MessageBox UI
- User classes
  - `Admin` - class of user which type is Admin
  - `Bidder` - class of user which type is Bidder
  - `Seller` - class of user which type is Seller
  - `RegUser` - class of user which register at system
  - `User` - main class which describes user properties
- UI
  - `LoginController` - controller of login UI
  - `RegistrationController` - controller of registration UI
  - `StartController` - controller of starting window
  - `Admin[...]Controller` - controllers of admin UI
  - `Bidder[...]Controller` - controllers of bidder UI
  - `Seller[...]Controller` - controllers of seller UI
  - `HistoryController` - controller for table in different UI's

- Engines
    - `SceneController` - used to switch between scenes with one line of code
    - `UtilsController` - used for different static functions
    - `DataBasee` - used to work with database
- Exceptions
    - `[...]Exception` - custom exceptions
- Main
    - `Main` - Main class, entry point

Diagrams




accounts
ID int
Login varchar
FirstName varchar
LastName varchar
Mail varchar
Hash varchar
Salt varchar
Balance bigint
Admin tinyint
Sex tinyint
AccountType int
notifications
ID int
ownerID int
Message varchar
Date datetime
Read tinyint
items
ID int
Name varchar
Description varchar
imgURL varchar
startingPrice int
currentPricet int
ownerID int
newOwnerID int
Category int
Accepted tinyint
Started tinyint
categories
ID int
Name varchar
dbdiagram.io