Bidder Estate Auction House

Michal Darovec

2022-05-16

Contents

Chapter 1

Documentation

1.1 Bidder Estate Auction House

1.2 Documentation

My project is about creating a real estate auction that works on principles of an English auction. It is being developed in Java in Intellij IDEA IDE with the help of JavaFX. I have implemented an easy version of login, with saving the data in a text document. The passwords are not hashed yet, but they will be in the final version.

The project has a home scene which allows you to log in. If the username is too long or too short, you have to change it. If you have logged in before, you have to log in with the same password. After clicking the button, you are automatically redirected to the auction scene. Here, you get a random property generated from the Storage class. The time for bidding is always 5 seconds and you are playing against bots, so you don't bet alone. Every time a bid is placed, the timer restarts. The auction ends when the time runs up and last bid always wins. So, either you get a congratulations message or a better luck next time message.

I have decided to create a singleton class called AuctionProcess where all the data is stored and transferred to other controllers.

1.3 Info

To run the program with the user list, in the Controller class, you first have to specify the path to the file where users.txt is or you can create your own file anywhere. You just need to specify the path to get all functionalities, even though it doesn't work completely right yet.

1.4 Functionality

1.5 Login scene

Enter username and password. If you are logging in again, you have to type the same password Both username and password are validated

1.6 Auction scene

Bid button and bid price text field allow the user to bid as much as they want. The bots are always bidding at the same time, so you have to beat them. Every time a bid is placed, the timer restarts. If the user wins, a congratulations message is displayed.

1.7 Features

1.8 Main criteria

1.8.1 Polymorphism

- **Admin Buy**
  - buy method is overridden, doesn't subtract money from Admin's account
  - package users, class Admin, line 20

1.8.2 Aggregation

- **User Contact and Address**
  - each user has a contact and address that are created when registering
  - package users, class User, lines 15, 16

1.8.3 Inheritance

- **Properties**
  - each property has inherited methods from an abstract Class named Property
  - package properties, class Property
- **Users**
  - Bot and Admin are inherited from the User class
  - package users, classes Bot, Admin

1.8.4 App Logic and GUI

- application logic is separated from GUI by using singleton classes and other helper classes and putting the major part of logic into them

- some controller classes do contain a bit of logic necessary to run the GUI window with no problems
- OOP principles are used in the project to separate the logic from the GUI as much as possible
- the methods passed from controllers to helper classes are passed by interfaces that communicate between these classes

1.9 Other criteria

1.9.1 Design patterns

- **Observer**
  - used for notifying all bots that they can bid
  - package auctionControl, class AuctionProcess, line 171 - method notifyAllBots()
- **Factory method**
  - used for encapsulated creation of Property objects outside the Storage class
  - package storage, class StorageFactory, is used in Storage, lines 21, 27
- **Singleton**
  - used as control classes that can store data and communicate with other classes
  - package auctionControl, class AuctionProcess, LoginControl

1.9.2 My own exceptions

- **Wrong Login Exception**
  - package exceptions, class WrongLoginException
  - used for wrong login input, in package auctionControl, class ControllerLogin, line 105

1.9.3 Multithreading

- **Timer Task**
  - for creating a Timer, we use the class TimerTask with an implemented run() method, which starts a new Thread
  - package auctionControl, class AuctionProcess, line 115

1.9.4 RTTI

- **Instance of List**
  - checking whether elements of a list are instance of class User
  - used in package serialization, class Serialize, line 12

1.9.5 Lambda expressions

- **Timer Task**
  - used for Timer Task creation, directly overriding and implementing the run() method
  - package auctionControl, class AuctionProcess

1.9.6 Serialization

- **Saving User Data**
  - used for knowing which user has or has not registered
  - keeping track of all users
  - package serialization, class Serialize

Chapter 2

UML diagram

Project contains the following diagrams:

- UML diagram
  - Use Case
  - Class diagram

2.1 Use Case

2.2 Class diagram

Showcase of all classes in the project and the links between them.

2.3 Field diagram

Showcase of all fields in the project and the links between them.

2.4 Method diagram

Showcase of all methods in the project and the links between them.

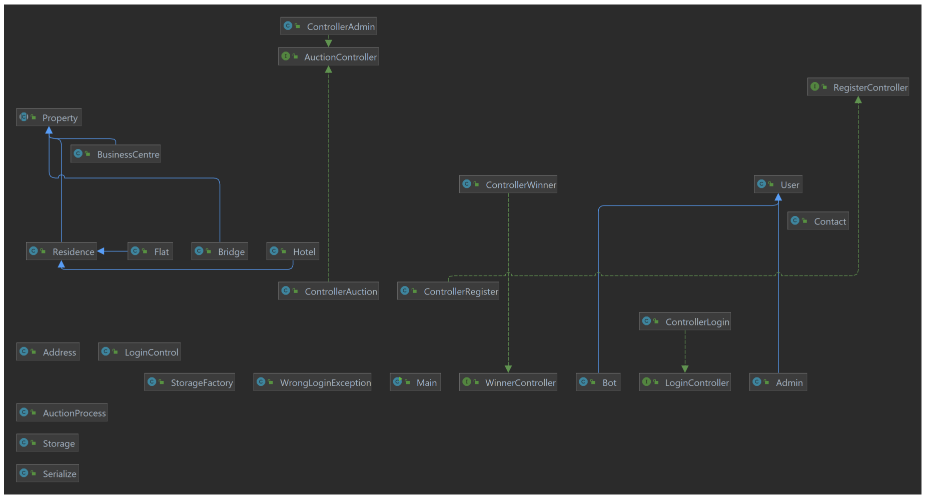

Figure 2.1: alt text

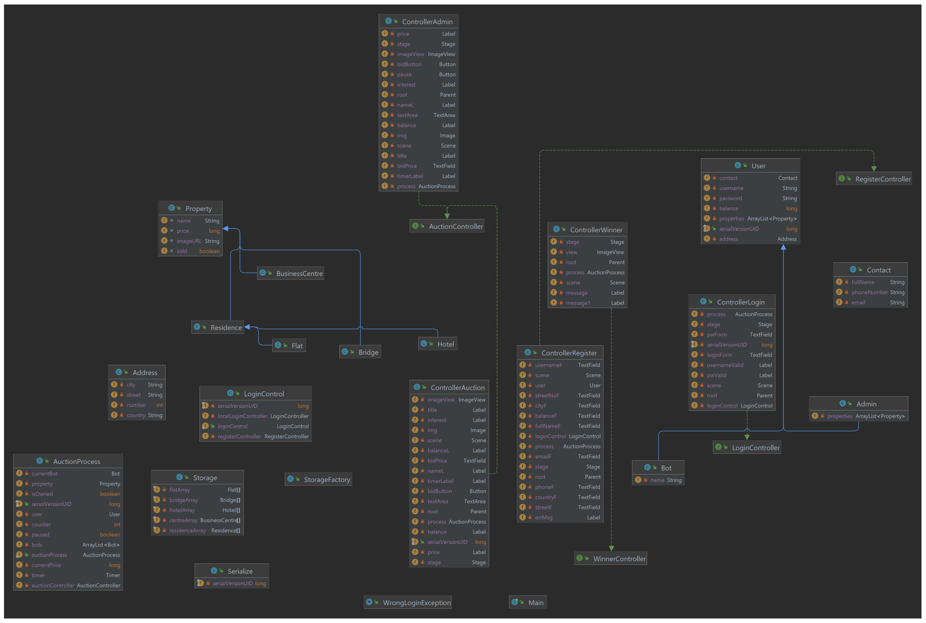

Figure 2.2: alt text

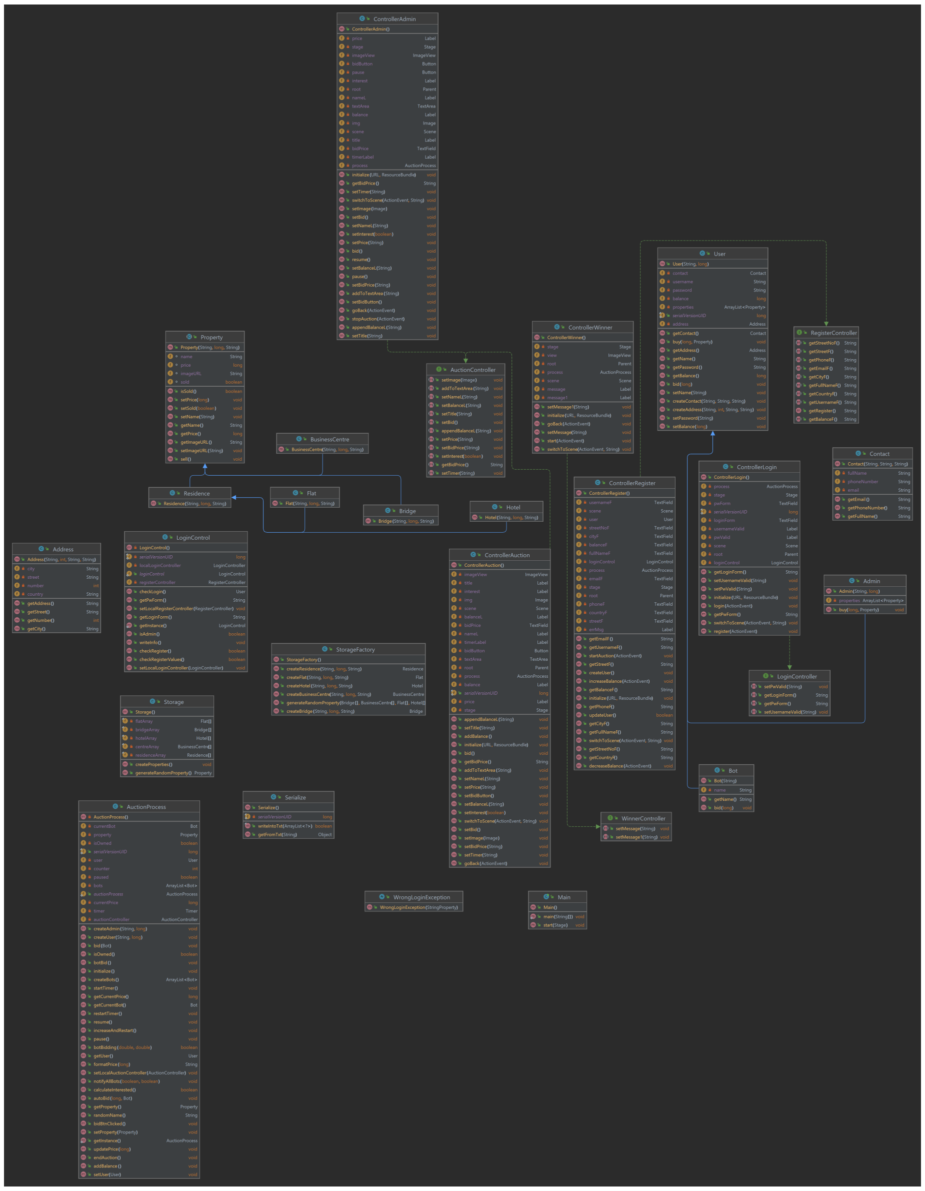

Figure 2.3: alt text

Chapter 3

Description of production versions

3.1 Version changes

3.1.1 Version 1.0

- starting gui and user class

3.1.2 Version 1.1

- first functional version of the project
- bug fixes

3.1.3 Version 1.2

- fully functional version of auction
- added functionality of users to buy
- winner window

3.1.4 Version 1.3

- added admin functionality
- added data saving - serialization
- added design patterns
- added other features
- current version of the project

Chapter 4

Environment setup

Example:

- Eclipse IDE for Java Developers (includes Incubating components)

Version: 2022-03 (4.23.0)

- JDK 17
- JavaFX 17
- Scene Builder

4.1 Installation

4.2 How to Run

Make sure to have Java 17 and also JavaFX 17. The correct location for the javaFX jar files is under `C:\javafx-sdk-17.0.2\lib` You have to add this file as the VM argument: `--module-path "C:\javafx-sdk-17.0.2\lib" --add-modules javafx.controls,javafx.fxml`

Make sure to have the right version of JRE System Library. For running the project, you need to use JRE 17. To change this library, go to Window -> Preferences -> Java -> Installed JREs and add the JRE 17. Then just set the current JRE to Project JRE in the Build Path and the program will run.

Chapter 5

Simulation and demonstration of use

The first thing is to add the VM arguments to the project as shown in the technical details. All the important code is highlighted by TODOs, so for the important lines of code, just check TODOs of the project and it will be much simpler.