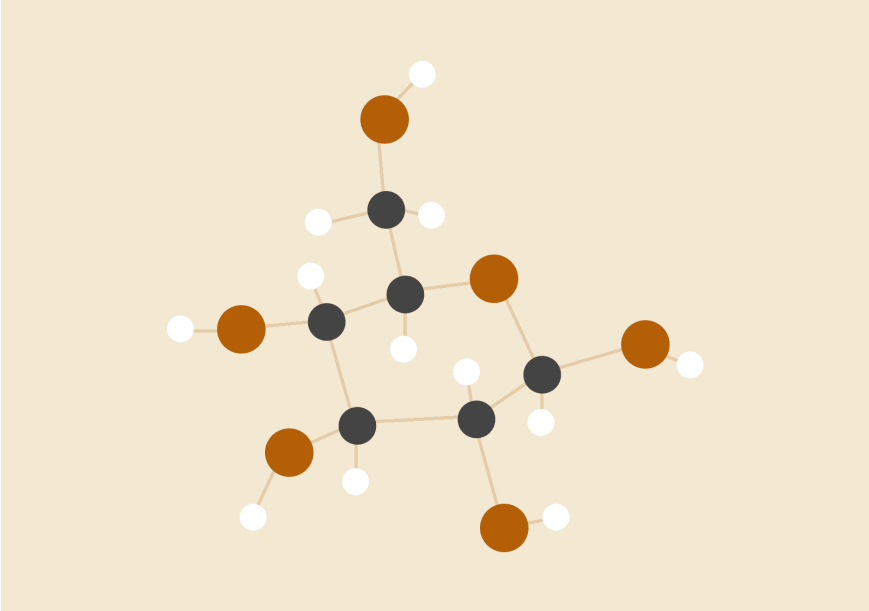
FINAL TEAM REPORT

TEAM 60 / COM2008



TEAM MEMBERS:

Zahra Hasan A Al Hilal

Ibrahim Majid HA Alkatheeri

Christopher Findon

Rory Barden

INTRODUCTION

We are creating a software system written in Java, with a user interface in Java Swing and a database in MySQL, for HomeBreaks Plc's property system.

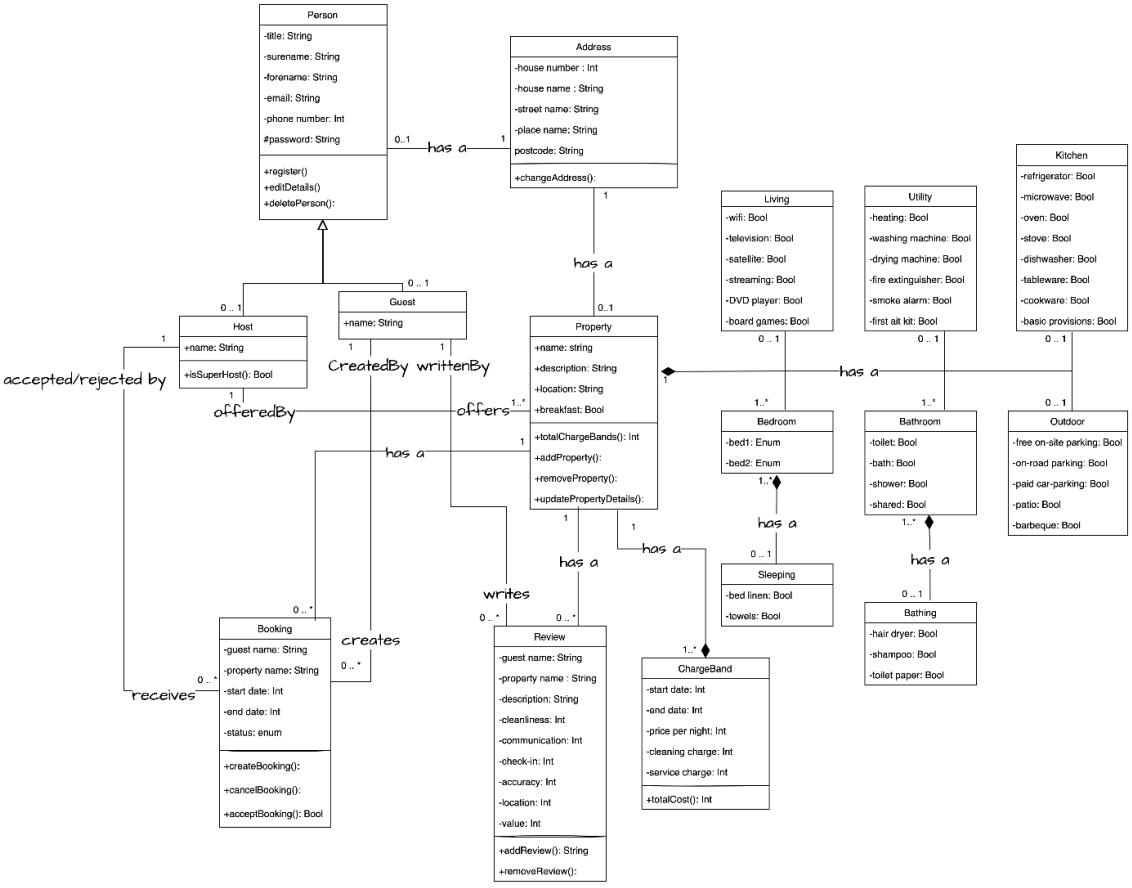
The system will allow users to register and login with either guest or host accounts, where hosts can add properties and guests can book properties. Users can also use the same email address for separate guest and host accounts. Users without an account can also view properties without being able to book them. When registering, users enter an email address, password, their full name and a display name, and their address.

Hosts have access to a large array of options to specify what amenities their properties provide and can customize a property's pricing for certain periods of time. When a booking is made on one of their properties, they must decide whether to accept or reject it. If accepted, the guest and host can see each other's confidential information.

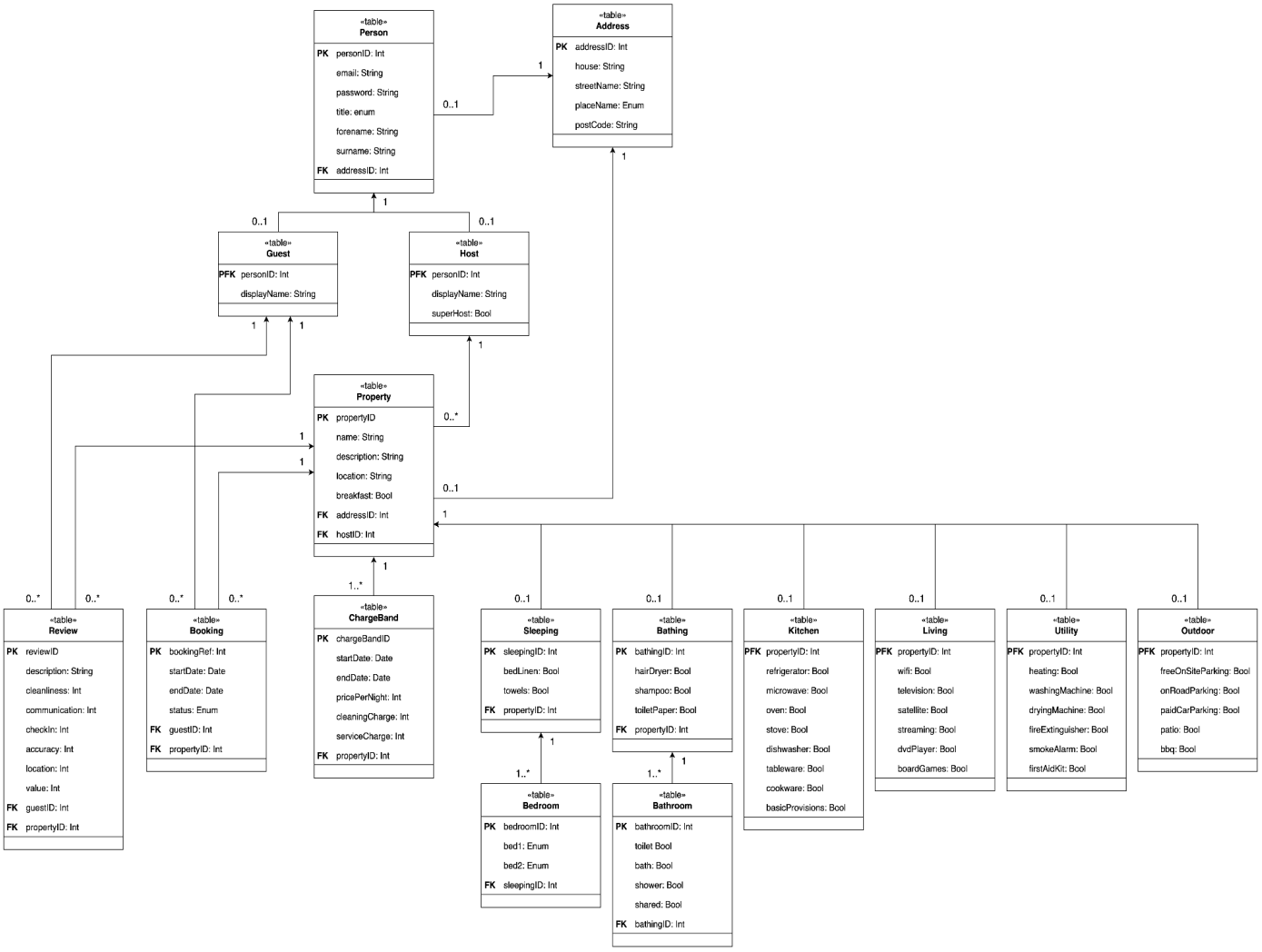
Guests have access to various filters to search for ideal properties. When they make a booking, they enter a start and end date and can see the price their booking would cost if it was accepted. However, payments are handled outside of the system. When a booking is completed, guests can write a review for their stay, which includes a text description and a 1 to 5 ratings in various categories. Reviews will then be visible to anyone who views the property. When hosts have a 4.7 average rating for their properties, they are marked as super hosts.

Finally, the database will be normalised and store data securely (such as encrypting passwords) and the system will be robust to cyber-attacks, such as SQL injections.

UML CLASS DIAGRAM OF THE INITIAL INFORMATION MODEL



UML CLASS DIAGRAM OF THE NORMALISED DATABASE MODEL



SCREENSHOTS

For the completed part of the software, the registration process is the best feature implemented carefully so that it fits the requirements of allowing users to have one email address for different roles and to be able to register an address for a person and a property at the same time.

This is a screen shot of the host registration process as an example.

Another good feature is customising the properties table for the specific role viewing it.

E.g , hosts can view their own properties, but enquirers and guests view all.

```
//method to create table of all properties
public void createTable() throws Exception {
    System.out.println("start createTable()...");
    String columns[] = {"name", "description", "location", "breakfast"};
    System.out.println("setting columns");
    int rowCounts = properties.size();
    System.out.println("row counts = "+ rowCounts);
    String[][] rows =  new String[rowCounts][4];
    System.out.println("setting up string 2x2array.."+ rows.length);
    //getting the rows details
    for (int i=0; i<rowCounts; i++) {
        rows[i][0] = properties.get(i).getName();
        System.out.println("setting the property name = ..."+ properties.get(i).getName() );
        rows [i][1] = properties.get(i).getDescription();
        rows [i][2] = properties.get(i).getLocation();
        if (properties.get(i).getBreakfast() == true) {
        rows [i][3] = "yes";
        }
        else if (properties.get(i).getBreakfast() == false) {
            rows [i][3] = "no";
        }
    }
    //setting up the table
    TableModel model = new DefaultTableModel(rows,columns);
    JTable table = new JTable(model);
    TableColumnModel columnModel = table.getColumnModel();
    columnModel.getColumn(0).setPreferredWidth(200);
    columnModel.getColumn(1).setPreferredWidth(300);
    columnModel.getColumn(2).setPreferredWidth(200);
    columnModel.getColumn(3).setPreferredWidth(200);
    JScrollPane jes = new JScrollPane(table, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
    add(jes,BorderLayout.CENTER);
}
```

```java
//method to register a new host
public Boolean registerHost() throws Exception {
    Boolean registered = false;
    PreparedStatement myStmt = null;
    ResultSet myRs = null;
    //prepare host parameters
    Address add = host.getAddress();
    String email = host.getEmail();
    String title = host.getTitle();
    String forename = host.getForename();
    String surname = host.getSurname();
    String phone = host.getPhoneNum();
    String encryptedPassword = PasswordUtils.encryptPassword(host.getPassword());
    String displayName = host.getDisplayName();
    Boolean isSuperhost = host.getIsSuperhost();

    try {
        //check if a person in the person table have the same email
        String pkEmail = host.getEmail();
        PersonDao personDao = new PersonDao(pkEmail);
        List<String> person = personDao.emailExist();
         //1- if a person with same email exist do:
        if (person != null){
                //check that this person id exist in the host table
                int id = Integer.valueOf(person.get(0));
                if (hostExist(id) == true) {//meaning host already exist
                    System.out.println("host with same email already registerd, try different email"
                            + "or register as a guest");
                }
                else if (hostExist(id) == false) {//if not, register new host
                    Host temp = new Host(id, host.getDisplayName(), host.getIsSuperhost());
                    HostDao tempH = new HostDao(temp);
                    tempH.addHost();
                    registered = true;

                }
                System.out.println("checking person with same email works..");
        }
        //2- if no, register this new person
        else if (person == null) {
                //insert to address table
                AddressDao addressDao = new AddressDao(add);
                System.out.println("declaring AddressDao works.."+ add);
                if (addressDao.addAddress() == true) {
                    //Retrieve the address ID again to use in Person table
                    int addId = addressDao.getAddressId();
                    //insert to Person table
                    myStmt = team060.prepareStatement("insert into Person"+
                            "(addressID,email,title,forename,surname,phone,password)"
                            +"values(?,?,?,?,?,?,?)");
                    myStmt.setInt(1, addId);
                    myStmt.setString(2, email);
                    myStmt.setString(3, title);
                    myStmt.setString(4, forename);
                    myStmt.setString(5, surname);
                    myStmt.setString(6, phone);
                    System.out.println("works fine before encrypt.");
                    myStmt.setString(7, encryptedPassword);
                    System.out.println("encryption works..");
                    myStmt.executeUpdate();
                    //Retrieve the person ID again to use in Host table
                    PersonDao personDao1 = new PersonDao(host.getEmail());
                    System.out.println("host email = "+ host.getEmail() );
                    int personId = personDao1.getPersonId();
                    //insert to Host table
                    Host temp = new Host(personId, displayName, isSuperhost);
                    HostDao tempH = new HostDao(temp);
                    tempH.addHost();
                    registered = true;
                }
        }
        return registered;
    }
    finally {
        DAOUtils.close(myStmt, myRs);
```

SECURITY DISCUSSION

Our system uses a special java class called jasypt which stands for java simplified encryption. You can find the documentation here : http://www.jasypt.org

One special class was created to handle the encryption process called PasswordUtils.java inside the DAO section. This class was NOT originally created by us, but it was adapted to fit our system functionality (credits: luv2code.com). The idea behind it is to ensure that no plain text passwords are stored in the database, instead, the encrypted version is stored.

Additionally, we ensured the database was normalised to prevent data redundancy and repetition and increase data consistency. The code was also designed specially to prevent SQL injections from being possible.

TASK DISTRIBUTION TABLE

Please write exactly what you have done and how many points you think you deserve.

TEAM MEMBER NAME	WHAT THEY HAVE DONE	POINTS AWARDED (TOTAL 100)
ZAHRA HASAN A ALHILAL	-arranged meeting with the team. -created the initial UML diagrams. -slightly modified the database tables (e.g made the id auto increment for easier access to data) -planned the process of creating the system. -planned the software package structure and created it. -created gitLab repository for other members to look at the code and work on it with me. -created and wrote 80% of	49

	the package (17/17 core classes) + (11/11 dao classes) + (2/9 gui classes and edited on the rest of them) -wrote the code for the registration process and secured passwords. -updated the team about the process on whatsapp group and pushed to the repository after each work session. -created the final host/guest/property objects -wrote the security discussion part of the report and added the uml diagrams.	
IBRAHIM MAJID HA ALKATHEERI	-Wrote the requirements of the project, simplified it and shared it as pdf in google documents. - Worked on the UML diagrams -worked on the report, (wrote the introduction for the project). -got in touch with my team members to make a group and get started with the project.	
CHRISTOPHER FINDON	-Wrote some of the introduction. -Worked on the initial information model diagram. -Reorganised the normalised database model diagram to match the requirements. -Normalised and organised the database to match the diagram. -Designed the basic plans	25

	for the structure and layout of the GUI. -Implemented the home, register, login and guest/host menus. -Implemented the input validation system. -Wrote a small part of the security discussion.	
RORY BARDEN	-implemented near the entire database -adjusted the database structure from the initial UML diagram to a useable and more secure format (e.g: assigning non-incremental integer IDs for certain private keys) -researched the appropriate data types for every column in the database based primarily on standards (such as the UK Data Standards) -wrote a summary of each table in the database to explain the purpose of columns, their data types, and their indices (foreign keys and unique, non-null columns etc.)	19

SIGNATURE

TEAM MEMBER SIGNATURE
Zahra hasan a alhilal
Christopher Findon
Rory Barden