Hotel Booking System

D0977966 劉峻愷 Kevin Liu

School of Computer Science

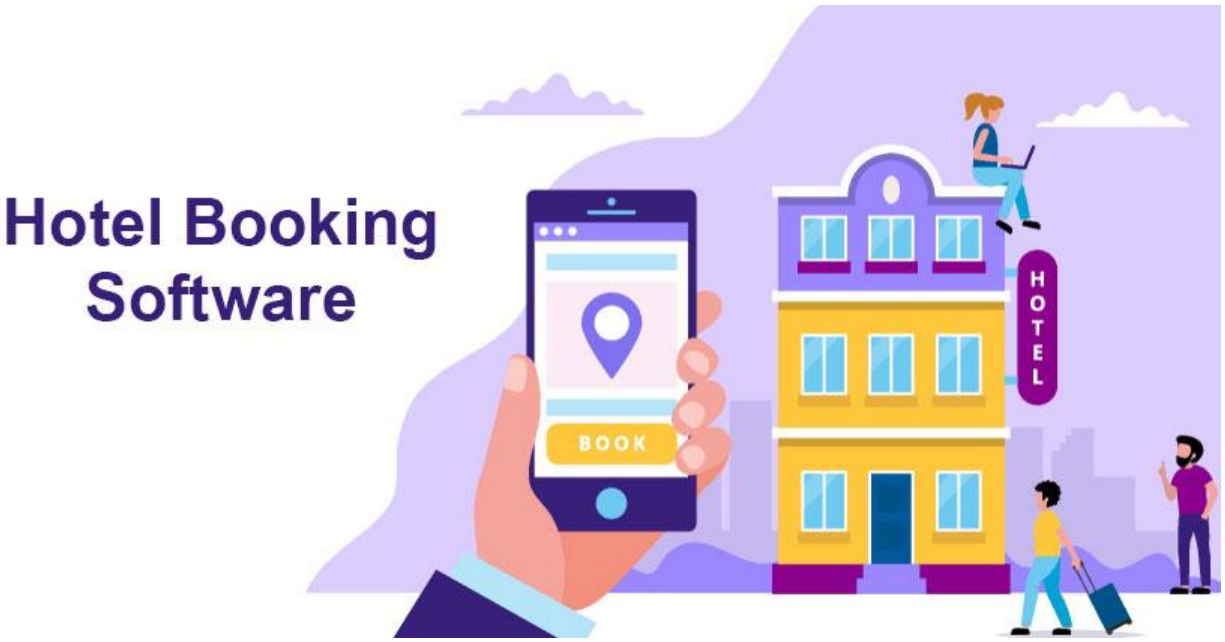
D0969212 張宸奕 Neo

School of Computer Science

D0965719 陳心榆 Shelly

School of Computer Science

Instructor: Prof Avinash Shankaranarayanan



Booking System

Content:

1. Introduction to the Hotel Booking system

When we travel around the world, we will have to book a hotel to make our trip more comfortable.
We designed this program to let users quickly book their rooms.

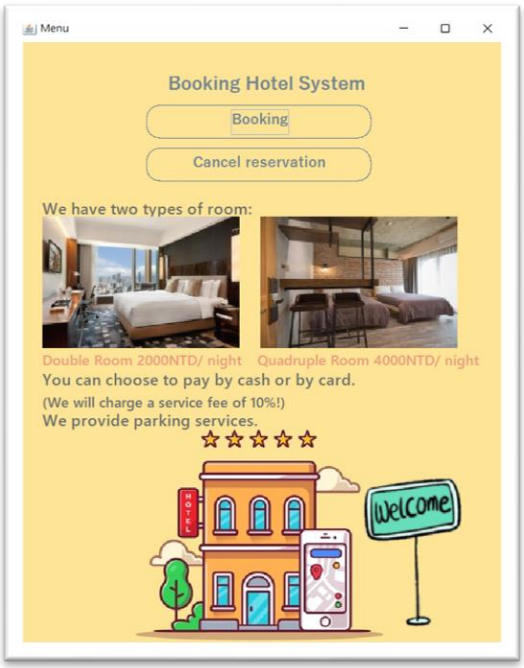
On this system home page, the user will have two choices: booking rooms and canceling reservation.

If users click the first option, it will open up another page for them to book the room. First, they have to enter their personal information like their name and phone number. Second, choose the day they want to live in and also how many days they want to stay. Third, choose which kind of room they want. There are two kinds of rooms, double room, and quadruple room. Users can also choose whether they want to park the car or not. After finishing all the blank, click confirm, the system will automatically calculate the price of your stay.

If users click the second option, it will need users to enter their name in order to find their information. Next, click on the delete button. Then, it will successfully cancel your reservation.

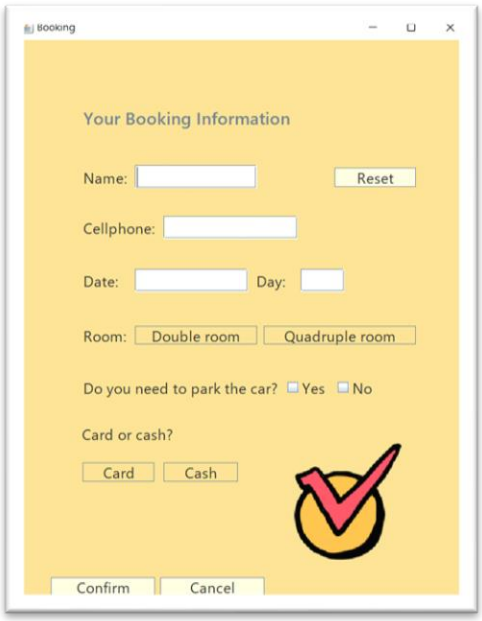
1.1 Menu interface:

In the first booking system interface, there have two options.one is booking and another is cancel reservation. We use Screen(); to switch the screen, if we click the button, it will open the relative coding program. In the middle of the page, it our simple introduction of our hotel and out principle of the reservation.



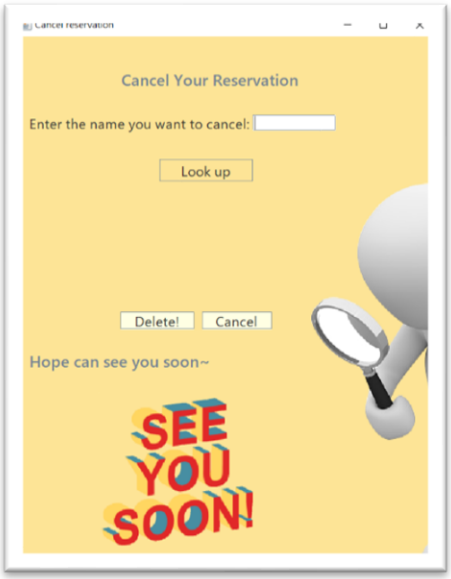
1.2 Booking system

In the booking system interface, there are some blank for the customer to enter their personal information and choose what type of room they want to book. If the user click the reset button, all information will be reset, so that they can enter the correct information again. And if the customer click the confirm button, it will show the successful booking.



1.3 Cancel reservation

In the cancel reservation interface, it has a blank for the customer to input their name, and the program will check whether the name is exist or not, if it is exist you can decide to delete or turn off the program. If your name exists, it will show you the information you have book and But if the name doesn't exist it will show you sorry your name is not found please check again.

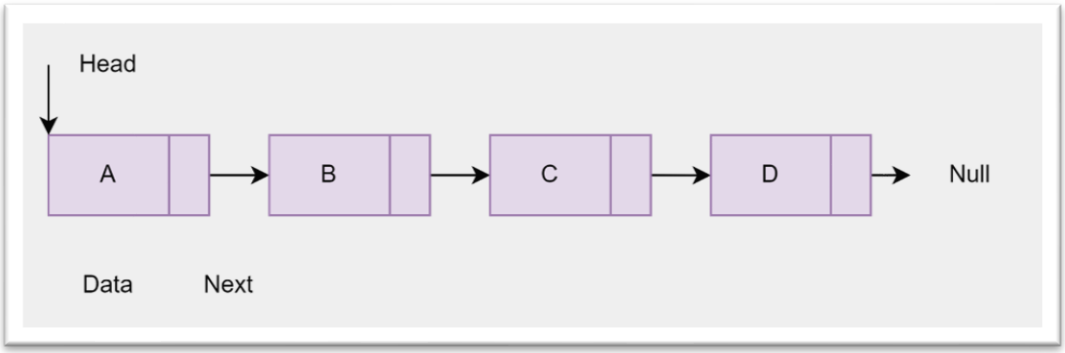


If you click the button delete, the program will delete the file and then show you that the delete is successful, if you click cancel, the program will turn off the page.

2. Methodology

2.1 Linked list

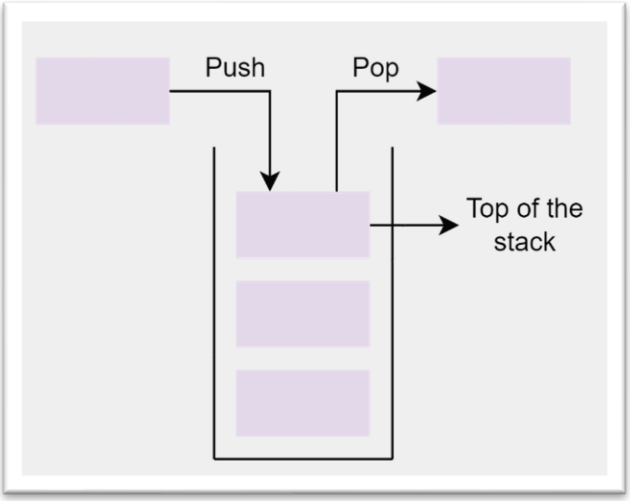
Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers. A linked list is represented by a pointer to the first node of the linked list. The first node is called the head. If the linked list is empty, then the value of the head is NULL.
Each node in a list consists of at least two parts:
1) data
2) Pointer (Or Reference) to the next node
In Java, LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.



《Schematic diagram of linkedlist》

2.2 Stack

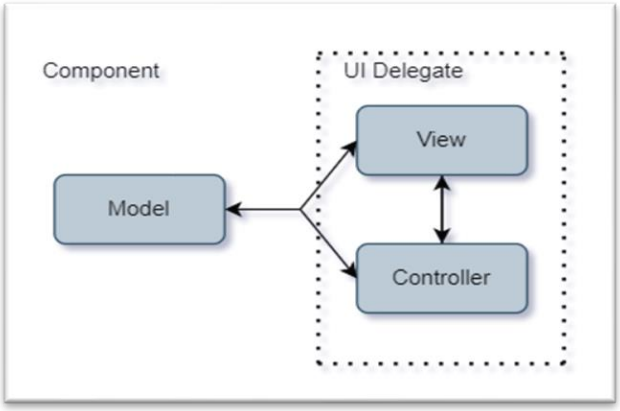
The stack is a linear data structure that is used to store the collection of objects. It is based on Last-In-First-Out . Java collection framework provides many interfaces and classes to store the collection of objects. One of them is the Stack class that provides different operations such as push, pop, search, etc.
In this section, we will discuss the Java Stack class, its methods, and implement the stack data structure in a Java program. But before moving to the Java Stack class have a quick view of how the stack works.
The stack data structure has the two most important operations that are push and pop. The push operation inserts an element into the stack and pop operation removes an element from the top of the stack. Let's see how they work on stack.



《Schematic diagram of Stack》

2.3 Java Swing

Swing is a set of toolkits provided by Java for the development of graphical interface applications and contains various elements for building Graphical User Interfaces (GUI), such as windows, labels, and buttons. It provides many screen display elements that are better than Abstract Window Toolkit (AWT). To distinguish from AWT components, Swing components are under the javax.swing.* package, and the class names all start with J, for example: JFrame, JLabel, JButton.
We choose to use Graphical User Interface (GUI) based on it can design the customized visualization and is easy to view and operate. We design our accounting program by WindowsBuilder from the marketplace.



«Java Swing MVC – Model Delegate»

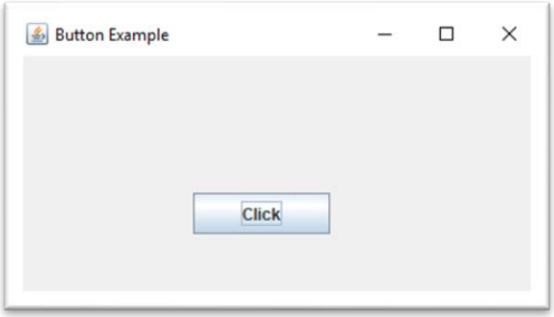
2.4 JLabel

Display area for short text strings or images or both. Labels do not react to input events. As a result, it cannot get keyboard focus. However, labels can display keyboard alternatives for components that have a keyboard alternative nearby but cannot display it.
JLabel objects can display text, images, or both. You can specify the alignment position of the label content in the label display area by setting vertical and horizontal alignment. By default, labels are vertically centered in their display area. By default, plain text labels are front-aligned; by default, only image labels are centered horizontally.

We can also specify the position of the text relative to the image. By default, the text is on the trailing edge of the image, and the text and image are vertically aligned.

2.5 Jbutton

Implementation of the "pressed" button.
Buttons can be configured through actions and controlled to a certain extent. There are many benefits to using actions with buttons beyond directly configuring buttons.

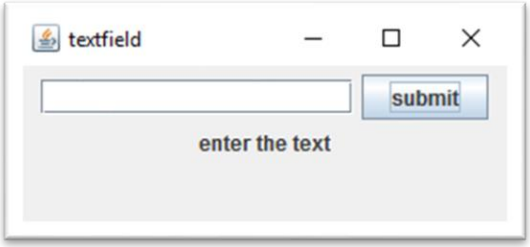


2.6 JTextField

JTextField is a lightweight component that allows editing of a single line

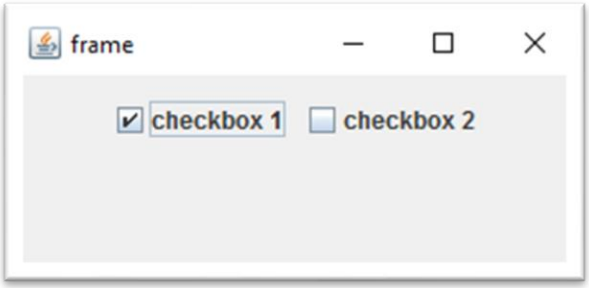
of text. For information and examples of using text fields, see How to Use Text Fields in the Java Tutorial.
JTextField is designed to be source-compatible with java.awt.TextField, and it makes sense to do so. This component has functionality not found in the java.awt.TextField class. The superclass should be consulted for additional functionality.

JTextField has a method to build strings that are used as command strings for triggered action events. java.awt.TextField uses the field's text as the ActionEvent's command string. If not empty, JTextField will use the command string set via the setActionCommand method, otherwise it will use the field's text for compatibility with java.awt.TextField.



2.7 JCheckbox

Implementation of Checkboxes - Items that can be selected or deselected and display their status to the user. By convention, any number of checkboxes in a group can be selected. For examples and information on using checkboxes, see How to Use Buttons, Checkboxes, and Radio Buttons in the Java Tutorial.
Buttons can be configured through actions and controlled to a certain extent. There are many benefits to using actions with buttons beyond directly configuring buttons. For more details, see Swing Components That Support Actions, which you can find in the How to Use Actions section of the Java Tutorial.



2.8 JOptionPane

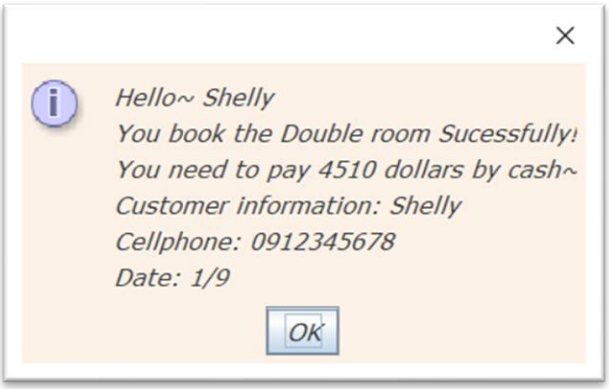
The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box, and input dialog box. These dialog boxes are used to display information or get input from the user.
The JOptionPane class inherits JComponent class.
The JOptionPane class has four different dialog boxes:

1.ConfirmDialog: Ask the question and user must press the button (Yes/No).

2.InputDialog: Prompt to enter text.

3.MessageDialog: Display information.

4. OptionDialog: Combine the other three dialog types.



《Schematic diagram of JOptionPane in our BookingSystem program》

2.9 Filewriter

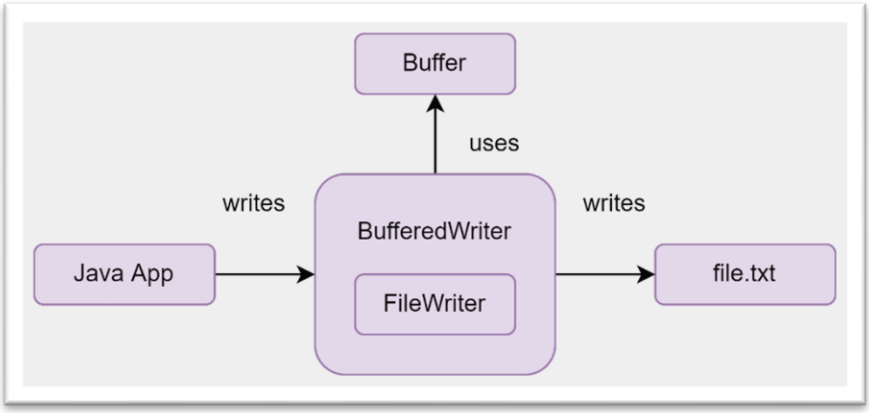
The Java.io.FileWriter class is a convenience class for writing character files. Following are the important points about FileWriter
1. The constructors of this class assume that the default character encoding and the default byte-buffer size are acceptable.
2. FileWriter is meant for writing streams of characters. For writing streams of raw bytes, use FileOutputStream.

《Schematic diagram of Filewriter》

2.10 BufferedWriter

The Java.io.BufferedWriter class writes text to a character-output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings. Following are the important points about BufferedWriter −

1. The buffer size may be specified, or the default size may be used.
2. A Writer sends its output immediately to the underlying character or byte stream.



《Schematic diagram of BufferedWriter》

2.11 Code analysis

[BookingSystem.java](#)

```
57          JButton btnNewButton = new JButton("Booking");
58          btnNewButton.setForeground(new Color(112, 128, 144));
59          btnNewButton.setFont(new Font("Yu Gothic", Font.BOLD, 16));
60          btnNewButton.addActionListener(new ActionListener() {
61              public void actionPerformed(ActionEvent e) {
62                  BookingSystem2 s1 = new BookingSystem2();
63                  s1.Screen_2();
64              }
65          });
```

Line 57 ~65: Create the JButton “Booking” and set its foreground and font, then add the ActionListener, if press this button, it will pop up the BookingSystem2 window.

```
67        JLabel lblNewLabel_2_3_1 = new JLabel("We provide parking services.");
68        lblNewLabel_2_3_1.setForeground(SystemColor.controlDkShadow);
69        lblNewLabel_2_3_1.setFont(new Font("Yu Gothic UI", Font.BOLD, 18));
70        lblNewLabel_2_3_1.setBounds(30, 404, 379, 37);
71        frmMenu.getContentPane().add(lblNewLabel_2_3_1);
72
73        JLabel lblRoom1_1 = new JLabel("");
74        lblRoom1_1.setIcon(new ImageIcon("C:\\Users\\shell\\Desktop\\456.png"));
75        lblRoom1_1.setFont(new Font("Yu Gothic UI", Font.PLAIN, 16));
76        lblRoom1_1.setBounds(340, 451, 299, 237);
77        frmMenu.getContentPane().add(lblRoom1_1);
```

Line 67 ~ 71, 109 ~ 113, 121 ~ 125, 133~ 137, 139 ~ 143, 145 ~ 149, 151 ~ 155: Create the JLabel to show the introduction of hotel.
Line 73 ~ 77, 115 ~ 119, 127 ~ 131: Create the JLabel and use setIcon function to show the picture by following the path.

BookingSystem2.java

```
85        JLabel lblNewLabel = new JLabel("Name:");
86        lblNewLabel.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
87        lblNewLabel.setBounds(83, 148, 62, 37);
88        frmBooking.getContentPane().add(lblNewLabel);
```

Line 85 ~ 88, 96 ~ 99, 107 ~ 110, 181 ~ 121, 166 ~ 169, 176 ~ 179: Use JLabel to show the text of some information, such as name, cellphone and room.

```
90        textField = new JTextField();
91        textField.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
92        textField.setBounds(146, 152, 147, 28);
93        frmBooking.getContentPane().add(textField);
94        textField.setColumns(10);
```

Line 90 ~ 94, 101 ~ 105, 112 ~ 116, 355 ~ 359: Use textField to let user enter their information.

```
127       JButton btnNewButton = new JButton("Double room");
128       btnNewButton.setForeground(UIManager.getColor("CheckBox.highlight"));
129       btnNewButton.setBackground(UIManager.getColor("CheckBox.shadow"));
130       JLabel lbldoubleJLabel = new JLabel("Double room");
131       btnNewButton.addActionListener(new ActionListener() {
132           public void actionPerformed(ActionEvent e) {
133               price = price + 2000;
134               lbldoubleJLabel.setFont(new Font("Yu Gothic", Font.PLAIN, 16));
135               lbldoubleJLabel.setBounds(150, 345, 100, 37);
136               lbldoubleJLabel.setVisible(true);
137               frmBooking.getContentPane().add(lbldoubleJLabel);
138               lbldoubleJLabel.setForeground(SystemColor.blue);
139               btnNewButton.setVisible(false);
140               btnQuadrupleRoom.setVisible(false);
141               Droom = true;
142           }
143       });
144       btnNewButton.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
145       btnNewButton.setBounds(146, 346, 147, 23);
146       frmBooking.getContentPane().add(btnNewButton);
```

Line 127 ~ 146: use JButton to let user choose whether they want to book, like room and whether they want to park the car. Add Actionlistener, because the price of double room is 2000 dollars, so if user press the Double room button, the integer price will increase 2000. Then, it will generate the JLabel to present what choice the user makes, and setVisible false to two buttons of room in order to avoid the situation of repeat selection.

```
220         JButton btnConfirm = new JButton("Confirm");
221         btnConfirm.setBackground(SystemColor.info);
222         btnConfirm.addActionListener(new ActionListener() {
223             public void actionPerformed(ActionEvent e) {
224
225                 String name = textField.getText().toString();
226                 String cellphone = textField_1.getText().toString();
227                 String date = textField_2.getText().toString();
228                 String day = textField_3.getText().toString();
229                 String choice = "";
230                 String room = "";
231                 if (Card) {
232                     choice = "card";
233                 } else if (Cash) {
234                     choice = "cash";
235                 }
236                 if (Droom) {
237                     room = "Double room";
238                 } else if (Qroom) {
239                     room = "Quadruple room";
240                 }
241                 int d = Integer.parseInt(textField_3.getText());
242                 price = (int) (price * d * 1.1);
243                 UIManager.put("OptionPane.background", new ColorUIResource(250, 240, 230));
244                 UIManager.put("Panel.background", new ColorUIResource(250, 240, 230));
245                 UIManager.put("OptionPane.buttonFont",
246                         new FontUIResource(new Font("MS Reference Sans Serif", Font.ITALIC, 13)));
247                 UIManager.put("OptionPane.messageFont",
248                         new FontUIResource(new Font("MS Reference Sans Serif", Font.ITALIC, 13)));
249                 namelist.add(name);
250                 phonelist.add(cellphone);
251                 datelist.add(date);
252                 daylist.add(day);
253                 roomlist.add(room);
254                 choicelist.add(choice);
255                 username = textField.getText();
```

Line 220 ~ 255: create the confirm button, if the user has entered all information and press this button, it will initial string and get text from the textfield and other choices. Next, it will calculate the price, and use UIManager to design the color and the font of JOptionpane. Then, add the string to the linkedlist.

```
257                 if (name.contains(" ")) {
258                     JOptionPane.showConfirmDialog(lblNewLabel_2, "You cannot enter the space! Please try agan.", null,
259                             JOptionPane.CLOSED_OPTION, JOptionPane.INFORMATION_MESSAGE);
260                 }
```

Line 257 ~ 260: if the name has the space, it will pop up a JOptionpane to remind the user that they cannot enter the space, so user should enter their information again.

```
261                 else {
262                     try {
263                         File file = new File(
264                                 "C:\\Users\\shell\\eclipse-workspace\\finalproject\\src\\finalproject\\users\\"
265                                         + username + ".txt");
266                         if (file.exists() && !file.isDirectory()) {
267                             JOptionPane.showConfirmDialog(lblNewLabel_2, "You've already booked.", null,
268                                     JOptionPane.CLOSED_OPTION, JOptionPane.INFORMATION_MESSAGE);
269                         } else {
270                             FileWriter newfileFile = new FileWriter(
271                                     "C:\\Users\\shell\\eclipse-workspace\\finalproject\\src\\finalproject\\users\\"
272                                             + username + ".txt",
273                                     true);
274                             BufferedWriter bw = new BufferedWriter(newfileFile);
275                             JOptionPane.showConfirmDialog(lblNewLabel_2,
276                                     "Hello~ " + name + "\nYou book the " + room + " Sucessfully!\nYou need to pay "
277                                             + price + " dollars by " + choice + "~" + "\nCustomer information: " + nam
278                                             + "\nCellphone: " + cellphone + "\nDate: " + date,
279                                     null, JOptionPane.CLOSED_OPTION, JOptionPane.INFORMATION_MESSAGE);
280                             bw.write(namelist.getLast() + System.lineSeparator());
281                             bw.write(phonelist.getLast() + System.lineSeparator());
282                             bw.write(datelist.getLast() + System.lineSeparator());
283                             bw.write(daylist.getLast() + System.lineSeparator());
284                             bw.write(roomlist.getLast() + System.lineSeparator());
285                             bw.write(choicelist.getLast() + System.lineSeparator());
286                             bw.close();
287                             newfileFile.close();
288                         }
289                     } catch (IOException exception) {
290                         JOptionPane.showConfirmDialog(lblNewLabel_2,
291                                 "An error occurred. You might enter the wrong information.");
292                         exception.printStackTrace();
293                     }
294                 }
295                 frmBooking.setVisible(false);
296             }
```

Line 261 ~ 297: define the new file in the absolute path. If the file exists, JOptionpane will show this name has already booked. Else, use filewriter to automatically create the new txt file and the name is the user input then use BufferedWriter to this file because BufferedWriter improves the efficiency in multiple small writes. At that time, JOptionpane will pop up and show the information and successful booking. At last, use write function to write the data in the txt file. Use try and catch to catch the error. If there is an error, it will show the JOptionpane to remind user they might enter the wrong information.

The reason for using BufferedWriter to write information in the file is the bufferedwriter uses FileWriter as the parameter, so we must use FileWriter before we use BufferedWriter. Moreover, the efficiency of the BufferedWriter used is much higher than that of the FileWriter. Because the former effectively uses the buffer, and only outputs to the file after the buffer is full or closed.

```
313         JCheckBox chckbxNewCheckBox = new JCheckBox("Yes");
314         chckbxNewCheckBox.setBackground(Color.WHITE);
315         JCheckBox chckbxNo = new JCheckBox("No");
316         chckbxNo.setBackground(new Color(251, 226, 146));
317         chckbxNewCheckBox.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
318         chckbxNewCheckBox.setBounds(326, 409, 62, 23);
319         chckbxNewCheckBox.setBackground(new Color(251, 226, 146));
320         JLabel lblyesJLabel = new JLabel("Yes");
321         chckbxNewCheckBox.addActionListener(new ActionListener() {
322             public void actionPerformed(ActionEvent e) {
323                 price = price + 50;
324                 lblyesJLabel.setFont(new Font("Yu Gothic", Font.PLAIN, 16));
325                 lblyesJLabel.setBounds(330, 409, 62, 23);
326                 lblyesJLabel.setForeground(SystemColor.blue);
327                 frmBooking.getContentPane().add(lblyesJLabel);
328                 chckbxNewCheckBox.setVisible(false);
329                 chckbxNo.setVisible(false);
330             }
331         });
332         frmBooking.getContentPane().add(chckbxNewCheckBox);
```

Line 313 ~ 346: use checkbox to let user chooses whether they want to park the car, if user presses yes, the price will add 50.

```
361         JButton btnReset = new JButton("Reset");
362         btnReset.setBackground(SystemColor.info);
363         btnReset.addActionListener(new ActionListener() {
364             public void actionPerformed(ActionEvent e) {
365                 price = 0;
366                 textField.setText(" ");
367                 textField_1.setText(" ");
368                 textField_2.setText(" ");
369                 textField_3.setText("");
370                 btnCard.setVisible(true);
371                 btnCash.setVisible(true);
372                 btnQuadrupleRoom.setVisible(true);
373                 btnNewButton.setVisible(true);
374                 chckbxNewCheckBox.setVisible(true);
375                 chckbxNo.setVisible(true);
376                 lblnoJLabel.setVisible(false);
377                 lblcashJLabel.setVisible(false);
378                 lblcardJLabel.setVisible(false);
379                 lbldoubleJLabel.setVisible(false);
380                 lblqJLabel.setVisible(false);
381                 chckbxNewCheckBox.setSelected(false);
382                 chckbxNo.setSelected(false);
383                 Card = false;
384                 Cash = false;
385                 Droom = false;
386                 Qroom = false;
387             }
388         });
389         btnReset.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
390         btnReset.setBounds(387, 155, 98, 23);
391         frmBooking.getContentPane().add(btnReset);
```

Line 364 ~ 391: Create the reset button and set its action. The all information will be reset, price will equal 0, textfield will be “ ”, and the button will be visible.

BookingSystem3.java

```
67          JLabel lblNewLabel = new JLabel("Enter the name you want to cancel: ");
68          lblNewLabel.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
69          lblNewLabel.setBounds(20, 94, 310, 37);
70          frmCancelReservation.getContentPane().add(lblNewLabel);
```

Line 67 ~ 70: Create the JLabel to let user know what information they have to enter.

```
72          JTextField textField = new JTextField();
73          textField.setBounds(310, 102, 108, 21);
74          textField.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
75          frmCancelReservation.getContentPane().add(textField);
76          textField.setColumns(10);
```

Line 72 ~ 76: Create the JTextField to let user enter their information.

```
97          btnNewButton.setBackground(UIManager.getColor("CheckBox.light"));
98          btnNewButton.addActionListener(new ActionListener() {
99              @SuppressWarnings("resource")
00              public void actionPerformed(ActionEvent e) {
01                  username=textField.getText();
02                  file = new File("C:\\Users\\shell\\eclipse-workspace\\finalproject\\src\\finalproject\\users\\"+username+".txt");
03                  Scanner sc=new Scanner(System.in);
04                  if (file.exists()&&!file.isDirectory()) {
05                      try {
06                          sc = new Scanner(file);
07                          String Name=sc.nextLine();
08                          String Phone=sc.nextLine();
09                          String Date=sc.nextLine();
10                          String Day=sc.nextLine();
11                          String Room=sc.nextLine();
12                          String Choice=sc.nextLine();
13                          lbllist_2.setForeground(new Color(112, 128, 144));
14                          lbllist.setForeground(new Color(112, 128, 144));
15                          lbllist_1.setForeground(new Color(112, 128, 144));
16                          lbllist_2.setText("Your booking name is: \n"+Name);
17                          lbllist.setText("Your cellphone: \n"+Phone);
18                          lbllist_1.setText("You have booked "+Room+" at "+Date+" for "+Day+" day(s)~");
19                      }
20                      catch (FileNotFoundException e1) {
21                      // TODO Auto-generated catch block
22                          e1.printStackTrace();
23                      }
24                  sc.close();
25                  }
26                  else {
27                      lbllist_2.setText("Sorry");
28                      lbllist.setText("Your name is not found!");
29                      lbllist_1.setText("Please check again~");
30                      lbllist_2.setForeground(Color.red);
31                      lbllist.setForeground(Color.red);
32                      lbllist_1.setForeground(Color.red);
33                  }
34              }
```

Line 96 ~138: Create the button look up, if user enters the name and presses this button, it will show their booking information. If file exists, scan this file and store in string. Else, it will show that the name is not found.

```
140         JButton btnDelete = new JButton("Delete!");
141         btnDelete.setBackground(SystemColor.info);
142         btnDelete.addActionListener(new ActionListener() {
143             public void actionPerformed(ActionEvent e) {
144                 file.delete();
145                 textField.setText("");
146                 lbllist.setText("Delete Successfully!");
147                 lbllist_1.setText("");
148             }
149         });
150         btnDelete.setFont(new Font("Yu Gothic UI", Font.PLAIN, 18));
151         btnDelete.setBounds(139, 358, 95, 23);
152         frmCancelReservation.getContentPane().add(btnDelete);
```

Line 140 ~ 152: The function of this button is to delete. So, if the user presses this button, the file about this user's booking information will be deleted and the JLabel will show if it is deleted successfully.

3.Results / Findings

3.1Complexity and run time

In order to check either linked list or stack will run faster, we use nanotime to record the result of two program, and the results are:

Run time:

Linked list add function: 34300 nano time
Stack push: 53000 nano time
Linked List Get Last: 103700 nano time
Stack pop: 136500 nano time

```
Linkedlist add 34300 nanotime
Linkedlist getLast 103700 nanotime
Linkedlist total: 138000
```

```
Stack push 53000 nanotime
Stack pop 136500 nanotime
Stack total: 189500
```

Time complexity:

In the single linked list, every time we add the new element in the linked list, operations will take constant O(1) time. When we decide to check up our appointment, the linked list will run from the head to the information which it located in, so the time complexity of getlast is O(n).in order to save time, we can use doubly linked list because it keeps

references to both head and tail, so the time complexity will be O(1).

In the stack function, if we push the item, the time complexity is O(1). And for the pop, the time complexity is still O(1), because it already know the information which need to pop out is the one which push in at last.

Space complexity:

In order to store the information for each customer, when there is 1customer, the space complexity is 1, if there are 2 customers, the space complexity is 2, when there are n costumers the space complexity is n, so we can summarize that the space complexity is O(n).

3.2 Comparison of Stack and LinkedList

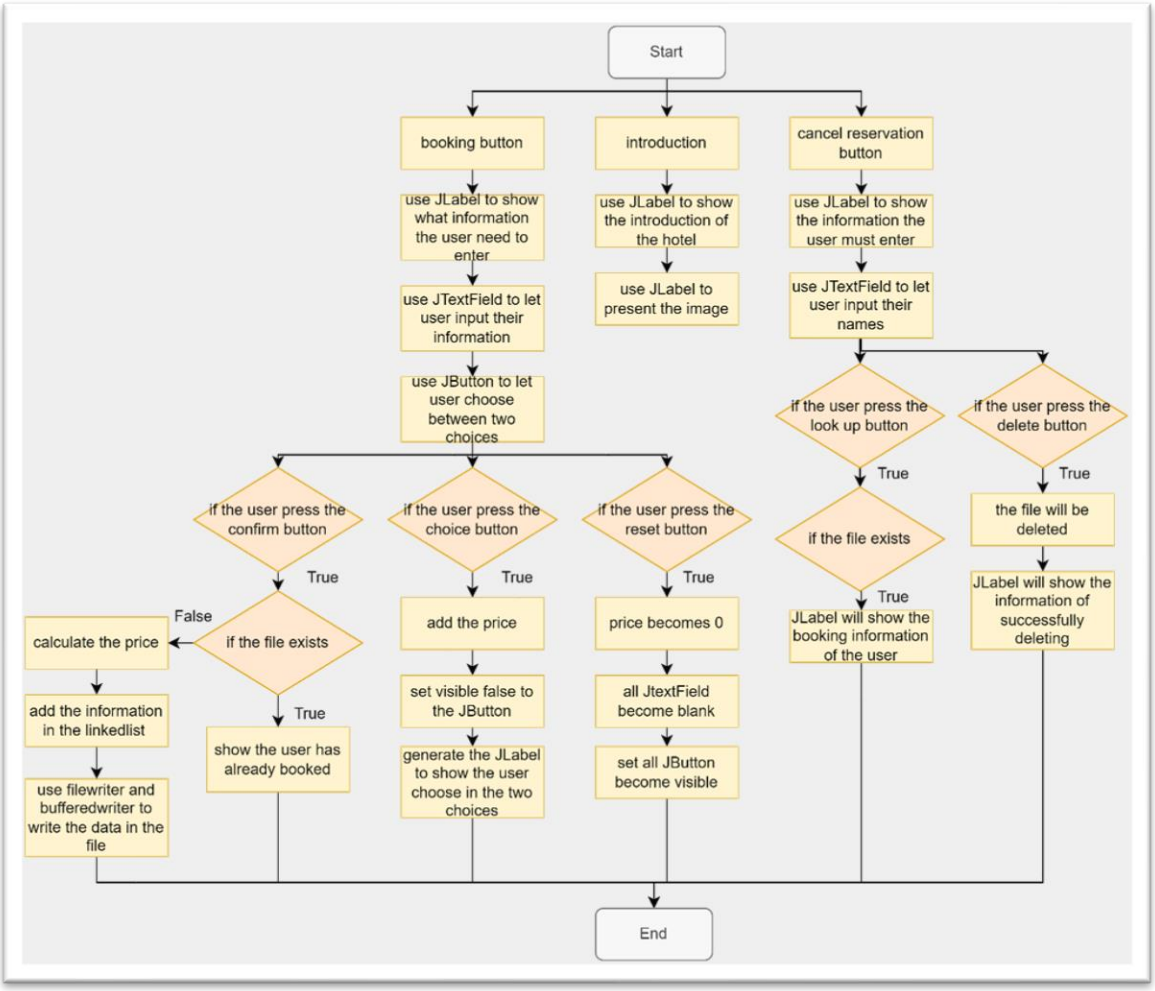
	LinkedList add()	**Stack push()**	**LinkedList remove()**	**Stack pop()**
Time complexity	**O(1)**	**O(1)**	**O(1)**	**O(1)**
Space complexity	**O(n)**	**O(n)**	**O(n)**	**O(n)**
Nanotime	**22900**	**50500**	**52600**	**723400**

3.3 Results

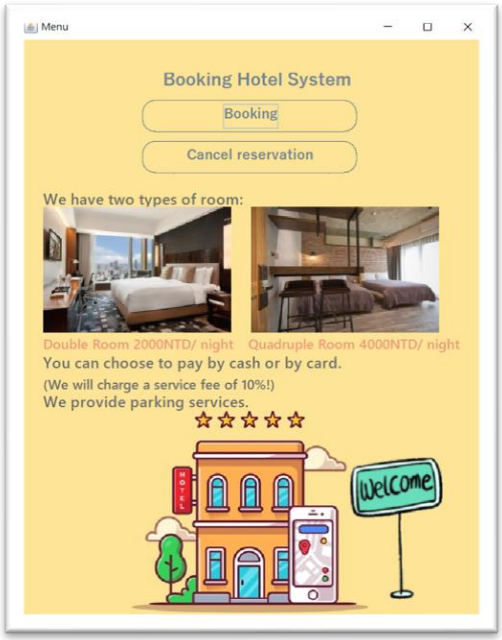
Compared to the complexity, we can find out that the time complexity and space complexity are the same, but when we talk about run time, linked list is much faster than Stack. So, from the perspective for the running time, it is much better to choose linked list to store the information in our booking system program.

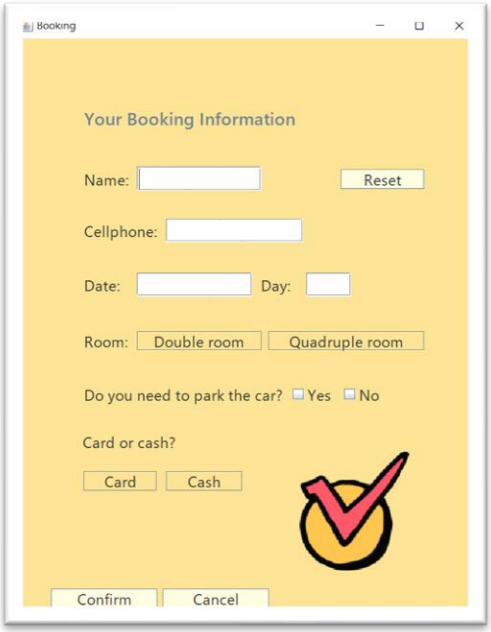
4. Certification Service

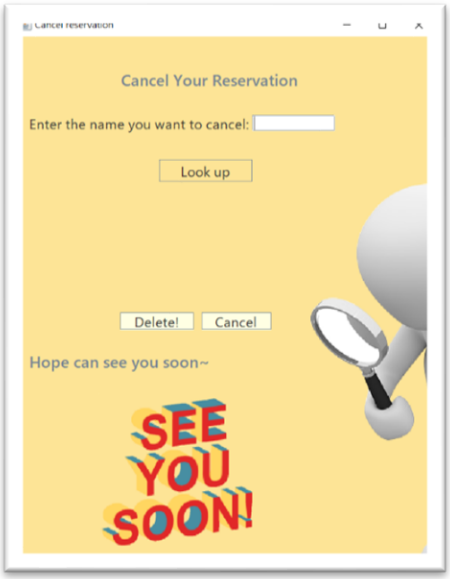
4.1 Flowchart of program



4.2 Picture of interface:


Menu
Booking Hotel System
Booking
Cancel reservation
We have two types of room:
Double Room 2000NTD/ night
Quadruple Room 4000NTD/ night
You can choose to pay by cash or by card.
(We will charge a service fee of 10%!)
We provide parking services.
Welcome


Booking
Your Booking Information
Name:
Reset
Cellphone:
Date:
Day:
Room:
Double room
Quadruple room
Do you need to park the car?
Yes
No
Card or cash?
Card
Cash
Confirm
Cancel


Cancel Your Reservation
Enter the name you want to cancel:
Look up
Delete!
Cancel
Hope can see you soon~
SEE YOU SOON!


Hello~ A
You book the Quadruple room Sucessfully!
You need to pay 2250 dollars by cash~
Customer information: A
Cellphone: 0912345678
Date: 1/2
OK


A.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
A
0912345678
1/2
5
Quadruple room
cash

4.3 Conclusion

Through this course, we have learned a lot of data structure, from the most basic how to write pseudo-code, judge O Notation, to the logic of linked list and Stack. We also applied these techniques and concepts to our final report and understood the differences between linked list and Stack in O-notation, compared to the time and space complexity of two programs. At the same time, we also recorded the output and input time of the two programs. It was found that the linked lists can run faster than Stack. And the linked list is also more convenient in storing data. so in the end we found that link list is much useful then stack in our booking system program.

4.4 References

Team, P. (n.d.). Question - Pretag. Pretag Development Team. https://pretagteam.com/question/how-to-create-rounded-jbutton-in-java

UIManager (Java Platform SE 7). (2020, June 24). S. https://docs.oracle.com/javase/7/docs/api/javax/swing/UIManager.html

Java.io.BufferedWriter Class. (n.d.). Www.Tutorialspoint.Com. https://www.tutorialspoint.com/java/io/java_io_bufferedwriter.htm

tutorialspoint.com. (n.d.). Java.io.FileWriter Class. Www.Tutorialspoint.Com. https://www.tutorialspoint.com/java/io/java_io_filewriter.htm

Java JOptionPane - javatpoint. (n.d.). Www.Javatpoint.Com. https://www.javatpoint.com/java-joptionpane

Java Stack - Javatpoint. (n.d.). Www.Javatpoint.Com. https://www.javatpoint.com/java-stack

GeeksforGeeks. (2021c, November 10). Linked List | Set 1 (Introduction). https://www.geeksforgeeks.org/linked-list-set-1-introduction/

I, I. (2016). i (i ed.) [E-book]. Alianza. https://wpollock.com/Java/JOptionPaneNote.pdf