Techniques used

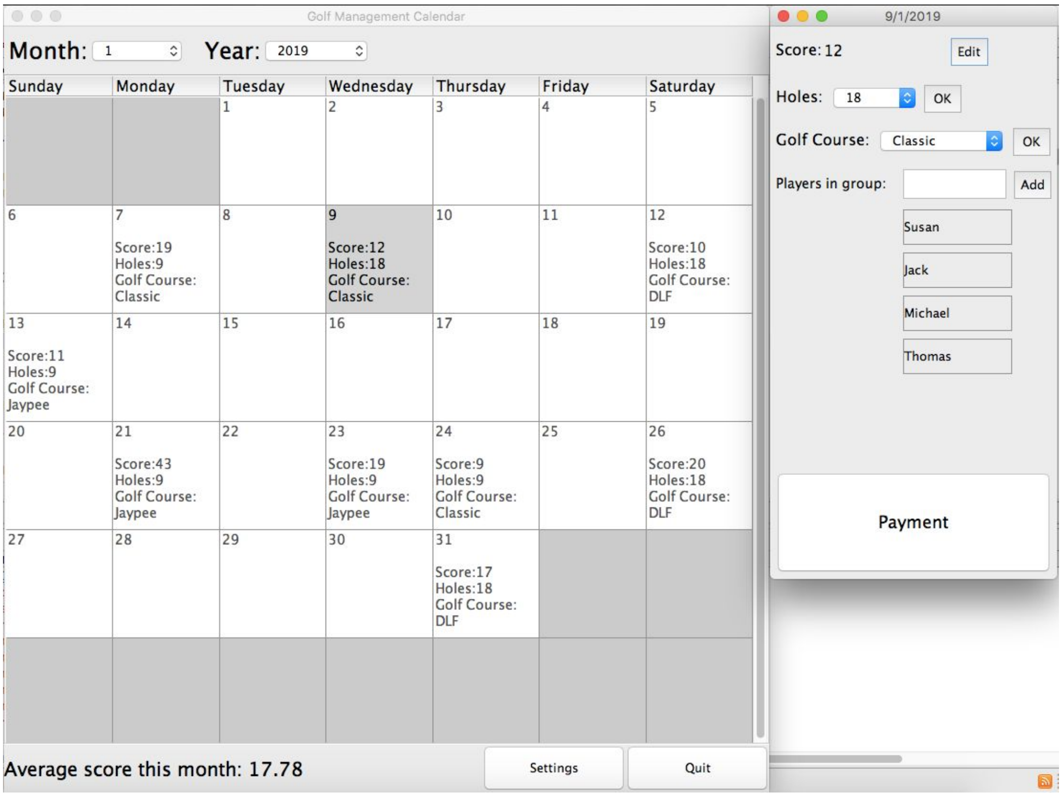
- Object Oriented Programming
- Graphical User Interface
- Algorithmic thinking
- Parameter passing
- Applying Promotion Sales
- Saving data

Object Oriented Programming

I used Object Oriented Programming to define various objects such as a Month object, Year Object, DateCal Object, Person Object, Golf Course Object, etc. I created classes for each Object, which included its own methods, getters and setters, as well as parameters. This provided a clear modular structure. In addition, by creating different classes, objects are maintained separately, making it easier to locate bugs and problems.

Graphic User Interface (GUI)

With the Swing Designer in Eclipse IDE, I was able to use tools provided by the Swing Designer, such as the Window Builder plugin. I utilized GUI components such as JButton, JTable, JComboBox, and JPanel. I used methods that allow the user to input values into a cell in a JTable, and updated the display of the table as values were being input. This allowed me to create an interactive relationship between the frames within the program.



Month: 1 Year: 2019

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1	2	3	4	5
6	7 Score:19 Holes:9 Golf Course: Classic	8	9 Score:12 Holes:18 Golf Course: Classic	10	11	12 Score:10 Holes:18 Golf Course: DLF
13 Score:11 Holes:9 Golf Course: Jaypee	14	15	16	17	18	19
20	21 Score:43 Holes:9 Golf Course: Jaypee	22	23 Score:19 Holes:9 Golf Course: Jaypee	24 Score:9 Holes:9 Golf Course: Classic	25	26 Score:20 Holes:18 Golf Course: DLF
27	28	29	30	31 Score:17 Holes:18 Golf Course: DLF		

Average score this month: 17.78

9/1/2019

Score: 12

Holes: 18

Golf Course: Classic

Players in group:

Susan

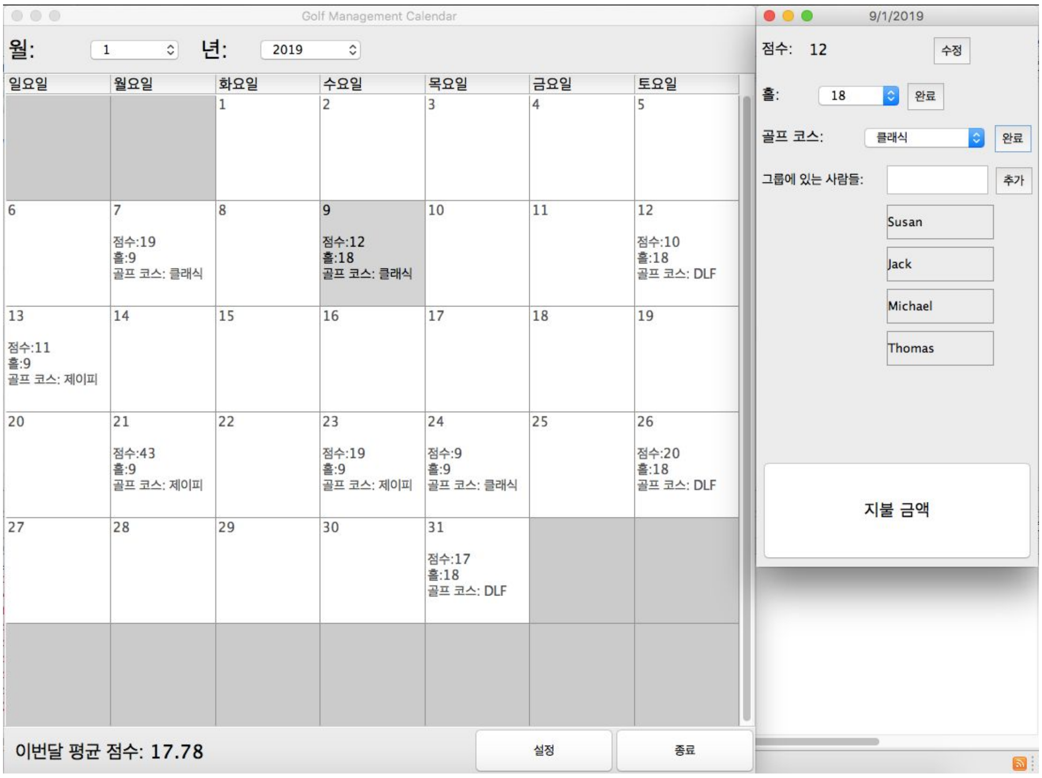
Jack

Michael

Thomas

↳*Use of GUI frame in eclipse, and interactive relation between the frames*

As my program is designed for my mother, whose first language is Korean, my program can be used in Korean as well. In the settings frame, under a combo box labelled "Language", the user can select "Korean", which will dispose the English Golf Calendar frame, then open the Korean one. The Korean Golf Calendar functions the same way as the English one and stores the same information.



Golf Management Calendar

월: 1 년: 2019

일요일	월요일	화요일	수요일	목요일	금요일	토요일
		1	2	3	4	5
6	7 점수:19 홀:9 골프 코스: 클래식	8	9 점수:12 홀:18 골프 코스: 클래식	10	11	12 점수:10 홀:18 골프 코스: DLF
13 점수:11 홀:9 골프 코스: 제이피	14	15	16	17	18	19
20	21 점수:43 홀:9 골프 코스: 제이피	22	23 점수:19 홀:9 골프 코스: 제이피	24 점수:9 홀:9 골프 코스: 클래식	25	26 점수:20 홀:18 골프 코스: DLF
27	28	29	30	31 점수:17 홀:18 골프 코스: DLF		

이번달 평균 점수: 17.78

9/1/2019

점수: 12

홀: 18

골프 코스: 클래식

그룹에 있는 사람들:

Susan

Jack

Michael

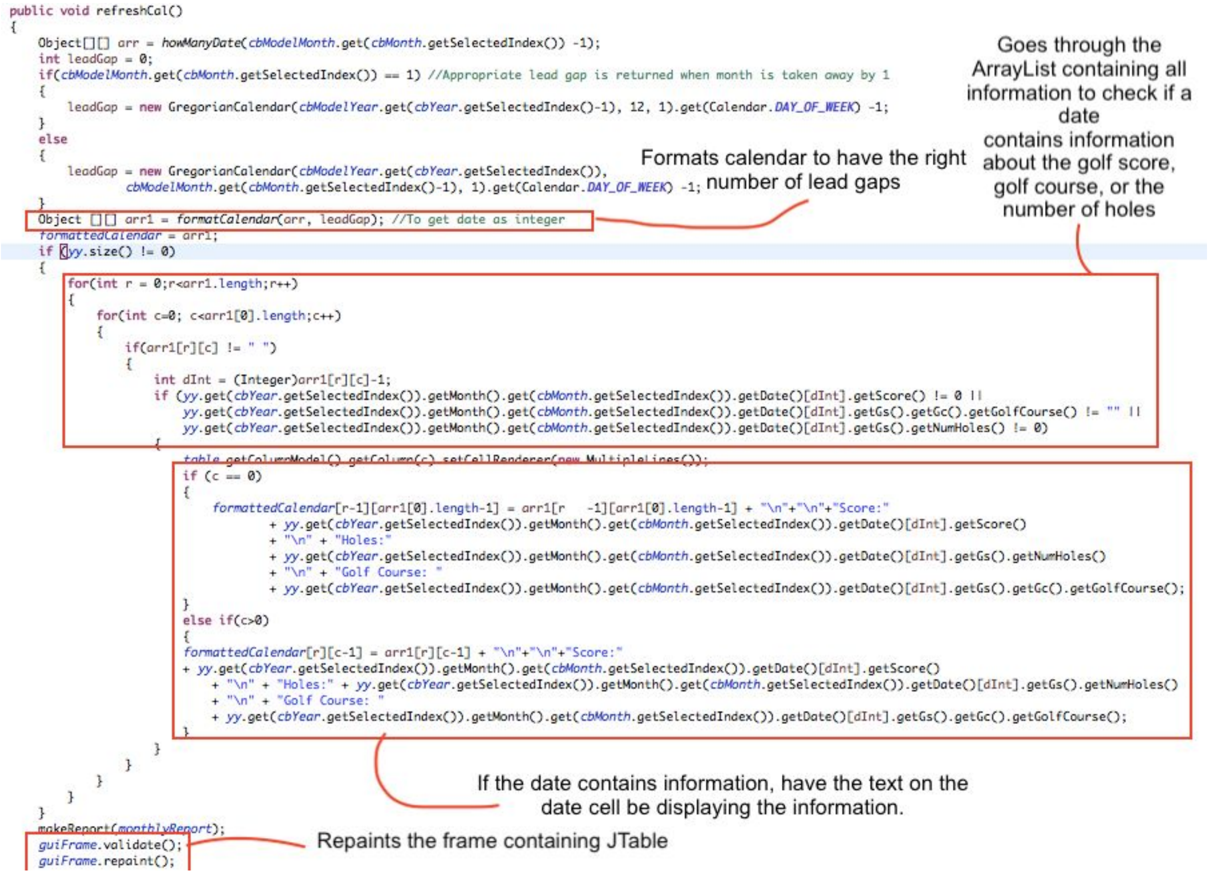
Thomas

↳*GUI frame of the Korean Golf Calendar*

Algorithmic Thinking

- Hierarchical data structure (nested arraylist) used in refreshing calendar
- Button that edits score. Score can be freely manipulated by the user.

Hierarchical data structure:



```
public void refreshCal()
{
    Object[][] arr = howManyDate(cbModelMonth.get(cbMonth.getSelectedIndex()) -1);
    int leadGap = 0;
    if(cbModelMonth.get(cbMonth.getSelectedIndex()) == 1) //Appropriate lead gap is returned when month is taken away by 1
    {
        leadGap = new GregorianCalendar(cbModelYear.get(cbYear.getSelectedIndex()-1), 12, 1).get(Calendar.DAY_OF_WEEK) -1;
    }
    else
    {
        leadGap = new GregorianCalendar(cbModelYear.get(cbYear.getSelectedIndex()),
                cbModelMonth.get(cbMonth.getSelectedIndex()-1), 1).get(Calendar.DAY_OF_WEEK) -1;
    }
    Object [][] arr1 = formatCalendar(arr, leadGap); //To get date as integer
    formattedCalendar = arr1;
    if (yy.size() != 0)
    {
        for(int r = 0;r<arr1.length;r++)
        {
            for(int c=0; c<arr1[0].length;c++)
            {
                if(arr1[r][c] != " ")
                {
                    int dInt = (Integer)arr1[r][c]-1;
                    if (yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getScore() != 0 ||
                        yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getGs().getGc().getGolfCourse() != "" ||
                        yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getGs().getNumHoles() != 0)
                    {
                        table.getColumnModel().getColumn(c).setCellRenderer(new MultipleLines());
                        if (c == 0)
                        {
                            formattedCalendar[r-1][arr1[0].length-1] = arr1[r   -1][arr1[0].length-1] + "\n"+"\n"+"Score:"
                                    + yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getScore()
                                    + "\n" + "Holes:"
                                    + yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getGs().getNumHoles()
                                    + "\n" + "Golf Course: "
                                    + yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getGs().getGc().getGolfCourse();
                        }
                        else if(c>0)
                        {
                        formattedCalendar[r][c-1] = arr1[r][c-1] + "\n"+"\n"+"Score:"
                        + yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getScore()
                            + "\n" + "Holes:" + yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getGs().getNumHoles()
                            + "\n" + "Golf Course: "
                            + yy.get(cbYear.getSelectedIndex()).getMonth().get(cbMonth.getSelectedIndex()).getDate()[dInt].getGs().getGc().getGolfCourse();
                        }
                    }
                }
            }
        }
    }
    makeReport(monthlyReport);
    guiFrame.validate();
    guiFrame.repaint();
```

Goes through the ArrayList containing all information to check if a date contains information about the golf score, golf course, or the number of holes

Formats calendar to have the right number of lead gaps

If the date contains information, have the text on the date cell be displaying the information.

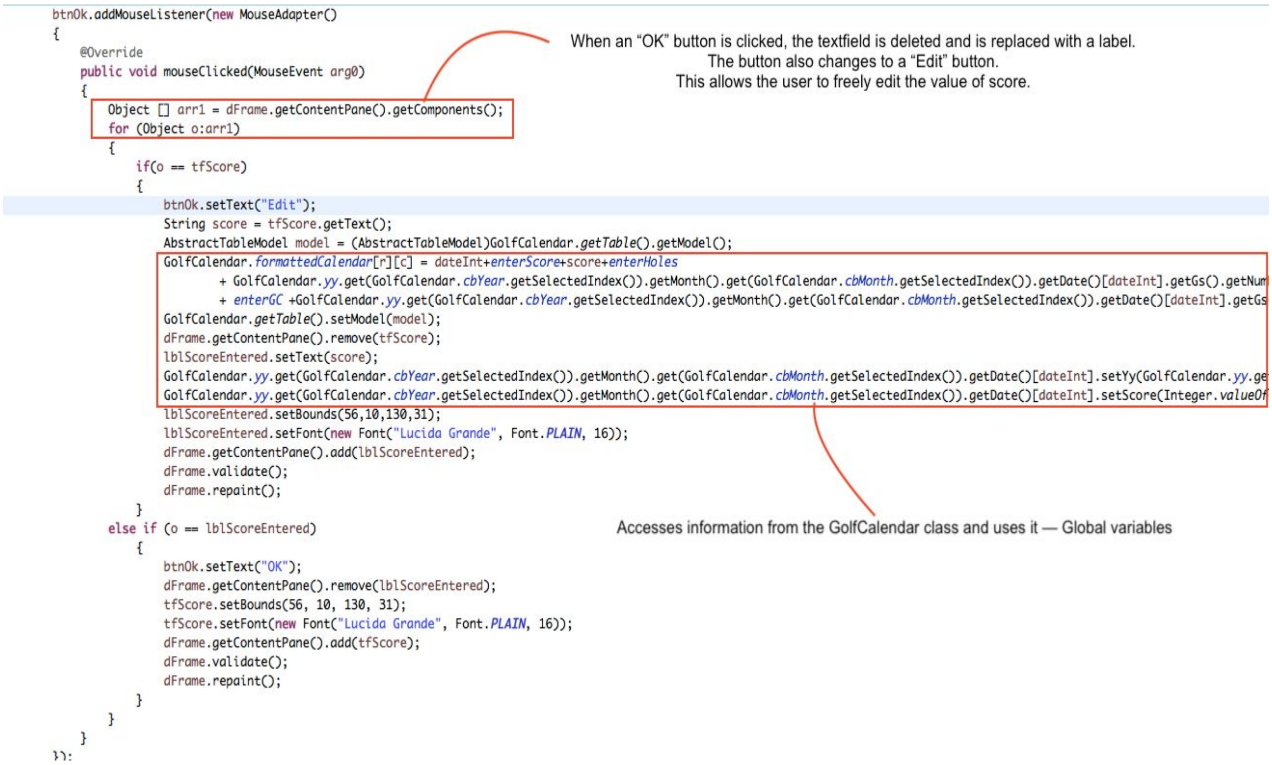
Repaints the frame containing JTable

I used 2D arrays, ArrayLists, and arrays as the data structures in my program. 2D arrays were used in my code to create the calendar (formattedCalendar[][]), and ArrayLists were used to store details of a golf session (yy). The lists were populated with the objects I created, which allowed me to extend my code.

I created a formatCalendar method that formats the calendar to have the appropriate amount of lead gaps corresponding to the month and year (A lead gap is the number of “empty” days before the first day of the month).

The hierarchy that I created goes like this: ArrayLxist of Year Objects, named yy, contains years from 1968 to 2019. Each Year object contains an ArrayList of Month Objects. Inside each year from 1968 to 2019, there is an ArrayList of Months, from January to December. Inside each Month Object, there is an Array of 31 DateCal Objects. Each DateCal Object consists of integer values of year, month, date, score, price, and a GolfSession Object. This way of hierarchical coding was inspired from the fact that each date in a month in a year contains different information.

Button that edits score:



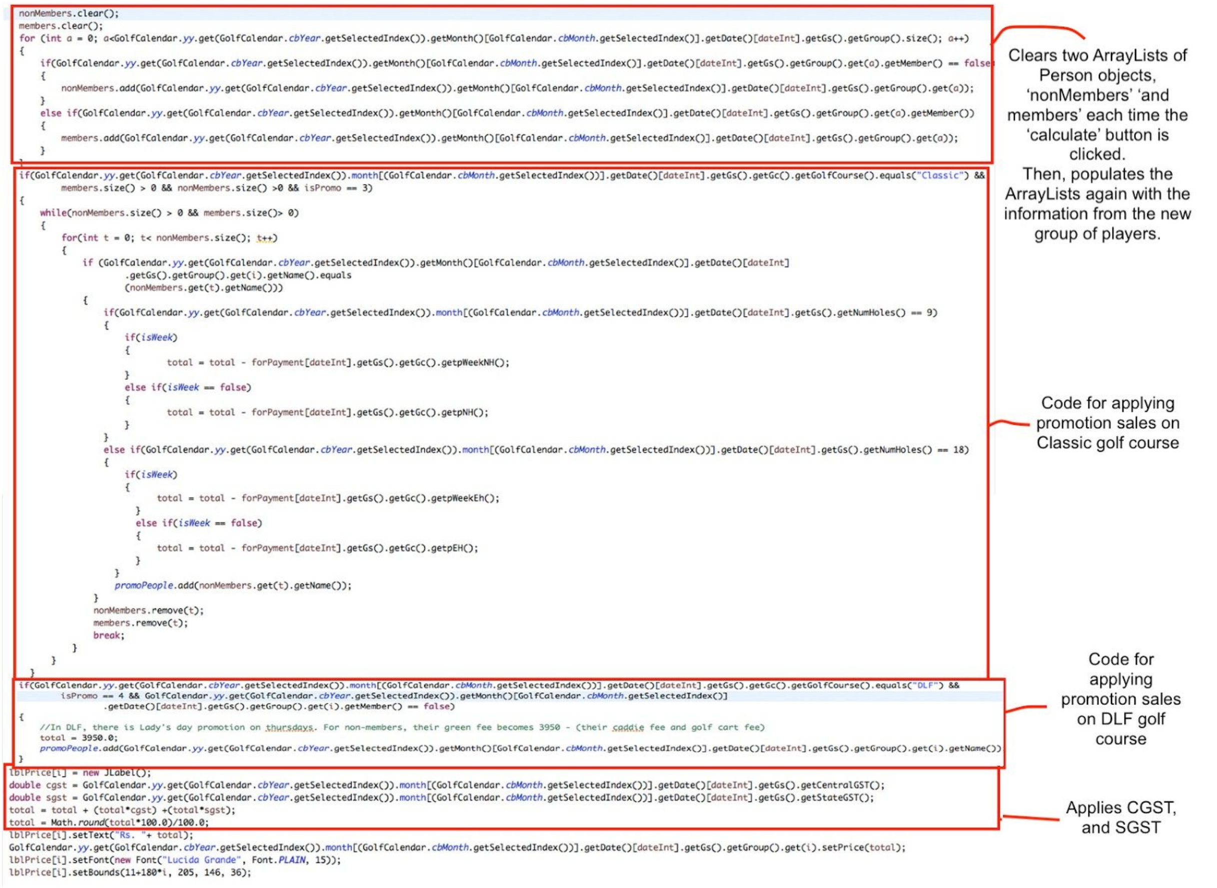
```
btnOk.addMouseListener(new MouseAdapter()
{
    @Override
    public void mouseClicked(MouseEvent arg0)
    {
        Object [] arr1 = dFrame.getContentPane().getComponents();
        for (Object o:arr1)
        {
            if(o == tfScore)
            {
                btnOk.setText("Edit");
                String score = tfScore.getText();
                AbstractTableModel model = (AbstractTableModel)GolfCalendar.getTable().getModel();
                GolfCalendar.formattedCalendar[r][c] = dateInt+enterScore+score+enterHoles
                        + GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth().get(GolfCalendar.cbMonth.getSelectedIndex()).getDate()[dateInt].getGs().getNum
                        + enterGC +GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth().get(GolfCalendar.cbMonth.getSelectedIndex()).getDate()[dateInt].getGs
                GolfCalendar.getTable().setModel(model);
                dFrame.getContentPane().remove(tfScore);
                lblScoreEntered.setText(score);
                GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth().get(GolfCalendar.cbMonth.getSelectedIndex()).getDate()[dateInt].setYy(GolfCalendar.yy.ge
                GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth().get(GolfCalendar.cbMonth.getSelectedIndex()).getDate()[dateInt].setScore(Integer.valueOf
                lblScoreEntered.setBounds(56,10,130,31);
                lblScoreEntered.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
                dFrame.getContentPane().add(lblScoreEntered);
                dFrame.validate();
                dFrame.repaint();
            }
        else if (o == lblScoreEntered)
            {
                btnOk.setText("OK");
                dFrame.getContentPane().remove(lblScoreEntered);
                tfScore.setBounds(56, 10, 130, 31);
                tfScore.setFont(new Font("Lucida Grande", Font.PLAIN, 16));
                dFrame.getContentPane().add(tfScore);
                dFrame.validate();
                dFrame.repaint();
            }
        }
    }
});
```

When an "OK" button is clicked, the textfield is deleted and is replaced with a label.
The button also changes to a "Edit" button.
This allows the user to freely edit the value of score.

Accesses information from the GolfCalendar class and uses it — Global variables

The button allows the user to edit the value of score at will, and in order to do that, the buttonClicked method utilizes Global values from the GolfCalendar class.

Applying Promotion Sales



```
nonMembers.clear();
members.clear();
for (int a = 0; a<GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()].getDate()[dateInt].getGs().getGroup().size(); a++)
{
    if(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()].getDate()[dateInt].getGs().getGroup().get(a).getMember() == false
    {
        nonMembers.add(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()].getDate()[dateInt].getGs().getGroup().get(a));
    }
    else if(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()].getDate()[dateInt].getGs().getGroup().get(a).getMember())
    {
        members.add(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()].getDate()[dateInt].getGs().getGroup().get(a));
    }
}
if(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).month[(GolfCalendar.cbMonth.getSelectedIndex())].getDate()[dateInt].getGs().getGc().getGolfCourse().equals("Classic") &&
        members.size() > 0 && nonMembers.size() >0 && isPromo == 3)
{
    while(nonMembers.size() > 0 && members.size()> 0)
    {
        for(int t = 0; t< nonMembers.size(); t++)
        {
            if (GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()].getDate()[dateInt]
                    .getGs().getGroup().get(i).getName().equals
                    (nonMembers.get(t).getName()))
            {
                if(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).month[(GolfCalendar.cbMonth.getSelectedIndex())].getDate()[dateInt].getGs().getNumHoles() == 9)
                {
                    if(isWeek)
                    {
                            total = total - forPayment[dateInt].getGs().getGc().getpWeekNH();
                    }
                    else if(isWeek == false)
                    {
                            total = total - forPayment[dateInt].getGs().getGc().getpNH();
                    }
                }
                else if(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).month[(GolfCalendar.cbMonth.getSelectedIndex())].getDate()[dateInt].getGs().getNumHoles() == 18)
                {
                    if(isWeek)
                    {
                           total = total - forPayment[dateInt].getGs().getGc().getpWeekEh();
                      }
                      else if(isWeek == false)
                      {
                           total = total - forPayment[dateInt].getGs().getGc().getpEH();
                      }
                  }
                  promoPeople.add(nonMembers.get(t).getName());
            }
            nonMembers.remove(t);
            members.remove(t);
            break;
        }
    }
}
if(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).month[(GolfCalendar.cbMonth.getSelectedIndex())].getDate()[dateInt].getGs().getGc().getGolfCourse().equals("DLF") &&
        isPromo == 4 && GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()]
                .getDate()[dateInt].getGs().getGroup().get(i).getMember() == false)
{
    //In DLF, there is Lady's day promotion on thursdays. For non-members, their green fee becomes 3950 - (their caddie fee and golf cart fee)
    total = 3950.0;
    promoPeople.add(GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).getMonth()[GolfCalendar.cbMonth.getSelectedIndex()].getDate()[dateInt].getGs().getGroup().get(i).getName())
}
lblPrice[i] = new JLabel();
double cgst = GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).month[(GolfCalendar.cbMonth.getSelectedIndex())].getDate()[dateInt].getGs().getCentralGST();
double sgst = GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).month[(GolfCalendar.cbMonth.getSelectedIndex())].getDate()[dateInt].getGs().getStateGST();
total = total + (total*cgst) +(total*sgst);
total = Math.round(total*100.0)/100.0;
lblPrice[i].setText("Rs. "+ total);
GolfCalendar.yy.get(GolfCalendar.cbYear.getSelectedIndex()).month[(GolfCalendar.cbMonth.getSelectedIndex())].getDate()[dateInt].getGs().getGroup().get(i).setPrice(total);
lblPrice[i].setFont(new Font("Lucida Grande", Font.PLAIN, 15));
lblPrice[i].setBounds(11+180*i, 205, 146, 36);
```

Clears two ArrayLists of Person objects, 'nonMembers' 'and members' each time the 'calculate' button is clicked. Then, populates the ArrayLists again with the information from the new group of players.

Code for applying promotion sales on Classic golf course

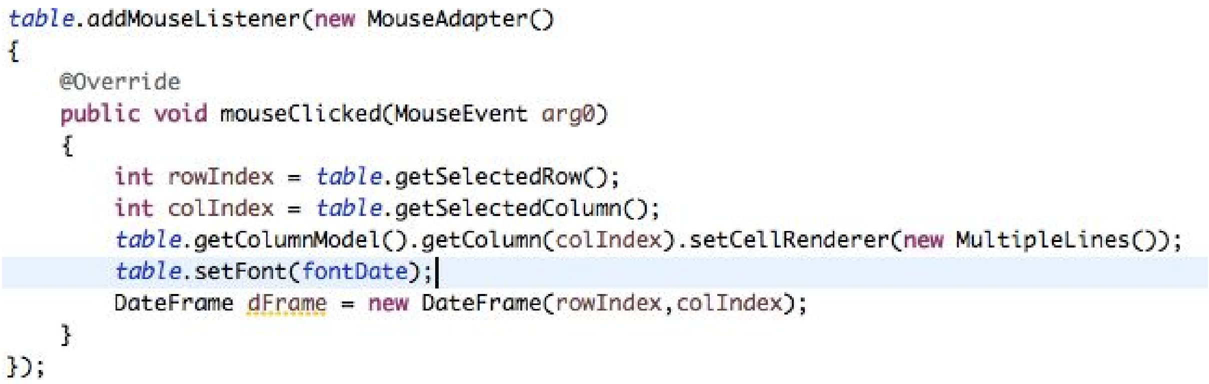
Code for applying promotion sales on DLF golf course

Applies CGST, and SGST

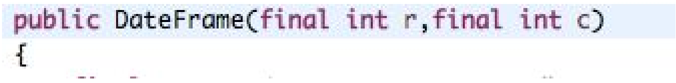
↳*Part of code from buttonClicked method for the "Calculate" button which calculates how much money each player in a group has to pay.*

When calculating how much a player has to pay, the program checks if the date at which a golf session took place, is applicable for a promotion sale. For the Classic golf course, there is a promotion every wednesday where for every member in a group, a non-member player pays the green fee as the member price, which is 0. For example, if there are 2 members and 2 non-members, all players will pay no green fee. In the code, the nonMember ArrayList is populated with non-member players, and member ArrayList with member players. Once the ArrayLists are populated, it checks if the name of the golf course in that golf session is equal to "Classic", if the size of the two ArrayLists are more than 0, and if the date is Wednesday. If the conditions are met, then a while loop runs and subtracts the green fees that would have been previously added to the total price of a non-member. The while loop stops as each person from nonMember and members is removed. As for the DLF golf course, the promotion is on thursday, which sets the total price of non-member players to 3950. Lastly, CGST (Central Goods and Services Tax) and SGST (States Goods and Service Tax) are applied. The values of tax are set in another part of code.

Parameter Passing



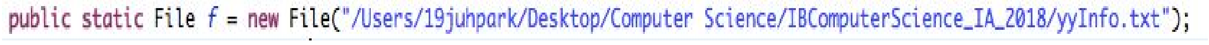
```
table.addMouseListener(new MouseAdapter()
{
    @Override
    public void mouseClicked(MouseEvent arg0)
    {
        int rowIndex = table.getSelectedRow();
        int colIndex = table.getSelectedColumn();
        table.getColumnModel().getColumn(colIndex).setCellRenderer(new MultipleLines());
        table.setFont(fontDate);
        DateFrame dFrame = new DateFrame(rowIndex,colIndex);
    }
});
```



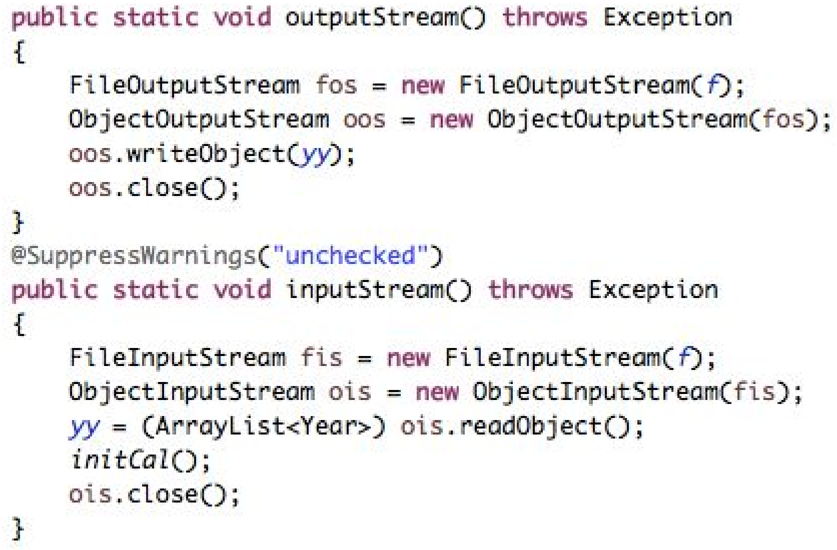
```
public DateFrame(final int r,final int c)
{
```

When a date cell in the JTable is clicked, the GolfCalendar class passes on integer values “rowIndex” and “colIndex” to the DateFrame class. This is done in order to pass on information about which date is clicked. The row and column values are utilized everywhere in the DateFrame class.

Saving



```
public static File f = new File("/Users/19juhpark/Desktop/Computer Science/IBComputerScience_IA_2018/yyInfo.txt");
```



```
public static void outputStream() throws Exception
{
    FileOutputStream fos = new FileOutputStream(f);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(yy);
    oos.close();
}
@SuppressWarnings("unchecked")
public static void inputStream() throws Exception
{
    FileInputStream fis = new FileInputStream(f);
    ObjectInputStream ois = new ObjectInputStream(fis);
    yy = (ArrayList<Year>) ois.readObject();
    initCal();
    ois.close();
}
```

I save information about the yy ArrayList in a text file, File f. The outputStream method uses File f to output the information of the ArrayLists, and the inputStream method calls upon the file and reads information from the file. This way, even if the program is closed, the information that might have been added by the user is saved in a file.

Word Count: 865