```
public class Module {

	//instance variables
	private String moduleCode, moduleTitle, timeSlot;
	private char room;
	private int classSize;

	/**
	 * constructor to initialize instance variables
	 * @param mc parameter to represent module code
	 * @param mt parameter to represent module title
	 * @param ts parameter to represent module time slot
	 * @param cs parameter to represent class size
	 * @param r parameter to represent room
	 */
	public Module(String mc, String mt, String ts, int cs, char r){
		moduleCode = mc;
		moduleTitle = mt;
		timeSlot = ts;
		room = r;
		classSize = cs;
	}

	/**
	 * accessor method to return module code
	 * @return module code
	 */
	public String getModuleCode(){
		return moduleCode;
	}

	/**
	 * accessor method to return module title
	 * @return module title
	 */
	public String getModuleTitle(){
		return moduleTitle;
	}

	/**
	 * accessor method to return module time slot
	 * @return module time slot
	 */
	public String getTimeSlot(){
		return timeSlot;
	}

	/**
	 * accessor method to return room assigned to module
	 * @return room
	 */
	public char getRoom(){
		return room;
	}

	/**
	 * accessor method to return class size
	 * @return classSize
	 */
	public int getClassSize(){
		return classSize;
```



```
	}

	/**
	 *method to get what level a module belongs to
	 * @return module level
	 */
	public char getLevel(){
		return moduleCode.charAt(2); //the level is represented by the t
hird character in the module title
	}

	/**
	 * method to return the subject that the course belongs to
	 * @return the module subject
	 */
	public String getSubject(){
		return moduleCode.substring(0, 2); //the subject is represented
by the first two characters of the module title
	}

	/**
	 * mutator method to modify module code
	 * @param s
	 */
	public void setModuleCode(String s){
		moduleCode = s;
	}

	/**
	 * mutator method to modify module title
	 * @param s
	 */
	public void setModuleTitle(String s){
		moduleTitle = s;
	}

	/**
	 * mutator method to modify module time slot
	 * @param s
	 */
	public void setTimeSlot(String s){
		timeSlot = s;
	}

	/**
	 * mutator method to modify room assigned to module
	 * @param c
	 */
	public void setRoom(char c){
		room = c;
	}

	/**
	 * mutator method to modify class size
	 * @param i
	 */
	public void setClassSize(int i){
		classSize = i;
	}
}
```

Jan 26, 16 15:27 **TeamProject.java** Page 1/1

```
import javax.swing.JFrame;

public class TeamProject {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		JFrame gui = new TimetableGUI();
		gui.setVisible(true);

	}
}
```

Feb 05, 16 3:42 **Timetable.java** Page 1/3

```
import java.util.ArrayList;

public class Timetable {

	//instance variables
	private ArrayList<Module> programme;
	private int moduleCount;

	//constructor to initialize the instance variables
	public Timetable(){
		programme = new ArrayList<Module>();
		moduleCount = 0;
	}

	/**
	 * method to create and add a module to the timetable
	 * @param line is a string containing a line read from the ModulesIn.txt
 file
	 */
	public void addModule(String line){
		String [] token = line.split("[]+"); //split the line into an ar
ray of strings
		String code = token[0]; //element at index 0 of token will be th
e module code
		String title = token[1]; //element at index 1 of token will be t
he module title
		String time = token[2]; //element at index 2 of token will be th
e time slot for module
		char room = token[3].charAt(0); //element at index 3 of token wi
ll be a string containing just one letter to represent room.

//this string is converted to a char type
		int capacity = Integer.parseInt(token[4]); //element at index 4
of token will be the number of people enrolled in the module.

//it is converted to an integer
		Module m = new Module(code,title,time,capacity,room); //create a
 module object from the variables read
		programme.add(m); //add the module object to the arraylist
		moduleCount++; //increment the number of modules currently prese
nt in the timetable
	}

	/**
	 * method to assign a module a time slot
	 * @param tt is the code of the module
	 * @param ts is the time slot to be assigned to the module
	 */
	public void assignModuleToTimeSlot(String tt, String ts){
		Module m = getModuleByCode(tt); //gets the module with the speci
fied title
		m.setTimeSlot(ts); //sets the time slot of that module
	}

	/**
	 * method to remove a module from a time slot
	 * @param tt is the code of the module
	 * @param ts is the time slot to be removed from the module
	 */
	public void removeModuleFromTimeSlot(String tt){
		Module m = getModuleByCode(tt); //gets the module with the speci
fied title
```

```
                m.setTimeSlot("?????"); //sets the timeslot of the module to "???
??" indicating that the module has no time slot
        }

        /**
         * method to assign a room to a module
         * @param tt is the code of the module
         * @param r is the room to be assigned to the module
         */
        public void addModuleToRoom(String tt, char r){
                Module m = getModuleByCode(tt); //gets the module with the speci
fied title
                m.setRoom(r);
        }

        /**
         * method to remove a room from a module
         * @param tt is the code of the module
         * @param r is the room to be removed
         */
        public void removeModuleFromRoom(String tt){
                Module m = getModuleByCode(tt); //gets the module with the speci
fied title
                m.setRoom('?'); //sets the room assigned to the module to '?' in
dicating that the module is not assigned to any room
        }

        /**
         * method to search for a module by its code
         * @param s is the code to be searched for
         * @return the module required
         */
        public Module getModuleByCode(String s){
                for(Module m: programme){
                        if(m != null && m.getModuleCode() == s) //if a module wi
th the specified title is found,
                                return m; //return that module
                }
                return null; //else module not found. Return null
        }

        /**
         * method to search for module(s) assigned to a particular time slot
         * @param t is the time to be searched for
         * @return the modules if any
         */
        public ArrayList<Module> getModuleByTime(String time)
        {
                //uses a list to store the module since a time slot can have mul
tiple modules assigned to it but with different rooms
                ArrayList<Module> list = new ArrayList<Module>();
                for(Module m: programme)
                {
                        if(m != null && m.getTimeSlot().equals(time)) //if a mod
ule is assigned to the specified time slot,
                                list.add(m); //add the module to the list
                }
                return list; //return the list
        }

        /**
         * accessor method to get all the modules in the timetable
```



```
         * @return an array containing all modules
         */
        public ArrayList<Module> getModules(){
                return programme;
        }

        /**
         * @return the moduleCount
         */
        public int getModuleCount() {
                return moduleCount;
        }
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.FileNotFoundException;
import java.io.*;
import java.util.*;

import javax.swing.*;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;

public class TimetableGUI extends JFrame implements ActionListener, WindowListene
r{

	//instance variables
	private Timetable tt;
	private ArrayList<Module> m;
	private JTable table;
	private JTextArea ta1;
	private JComboBox<String> cb1;
	private JComboBox<String> cb2;
	private JComboBox<String> cb3;
	private JButton b1,b2,b3,b4; //b1 is remove, b2 is assign, b3 is save, b
4 is exit
	private String [][] rowData;

	//room capacities
	private final int ROOM_A_SIZE = 100;
	private final int ROOM_B_SIZE = 100;
	private final int ROOM_C_SIZE = 60;
	private final int ROOM_D_SIZE = 60;
	private final int ROOM_E_SIZE = 60;
	private final int ROOM_F_SIZE = 30;
	private final int ROOM_G_SIZE = 30;
	private final int ROOM_H_SIZE = 30;

	private final String inputFile = "ModulesIn.txt";
	private final String outputFile = "ModulesOut.txt";

	/**
	 * constructor to create a TimetableGUI object
	 */
	public TimetableGUI(){
		tt = new Timetable();
		m = tt.getModules();

		if(readFile()) //if ModulesIn.txt file can be read and opened, l
ay out components
		{
			layoutGUIComponents();
			displayCourses();
			fillTable();
		}
		else //otherwise terminate program
		{
			JOptionPane.showMessageDialog(null, "No text file to read. Exiting
program...", "Error: No Input File", JOptionPane.ERROR_MESSAGE);
			System.exit(0);
		}
```



```
	}

	private void layoutGUIComponents()
	{
		setTitle("MIT Timetabling Assistant"); //set the title of the window
		setSize(925,585); //sets the window dimensions
		setLocation(350,100); //sets the window location
		setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
		addWindowListener(this); //to enable a customized operation when
 the close button of the JFrame is clicked
		setResizable(false); //disables resizing of window

		String [] columnNames = {" ","A(100)","B(100)","C(60)","D(60)","E(60)"
,"F(30)","G(30)","H(30)"}; //correspond to the names of columns.
                                                        //The numbers in bracket
s represent room capacities

		rowData = new String[10][9]; //creates a String array object to
hold table data
		//fills the first column of each row of the table with the speci
fied time slots
		rowData[0][0] = "MonAM";	rowData[1][0] = "MonPM";
		rowData[2][0] = "TueAM";	rowData[3][0] = "TuePM";
		rowData[4][0] = "WedAM";	rowData[5][0] = "WedPM";
		rowData[6][0] = "ThuAM";	rowData[7][0] = "ThuPM";
		rowData[8][0] = "FriAM";	rowData[9][0] = "FriPM";

		//panels to be used for component placement
		JPanel p1 = new JPanel();
		JPanel p2 = new JPanel();
		JPanel p3 = new JPanel();
		JPanel p4 = new JPanel();
		JPanel p5 = new JPanel();
		JPanel p6 = new JPanel();


		table = new JTable(rowData,columnNames)
				{
					//prevent cell from being editable
					@Override public boolean isCellEditable(
int row, int col)
					{
						return false;
					}
				};
		table.setRowHeight(25); //sets the height of each table row
		//Prevents column headers from being rearranged or resized
		JTableHeader th = table.getTableHeader();
		th.setReorderingAllowed(false);
		th.setResizingAllowed(false);
		table.setPreferredScrollableViewportSize(new Dimension(600, 250)
);

		JScrollPane sP = new JScrollPane(table); //adds the table to a s
croll pane

		GridLayout grid2 = new GridLayout(2,1);
		p1.setLayout(grid2);

		p1.add(sP, BorderLayout.NORTH);
```

```
                GridLayout grid3 = new GridLayout(5,2);
                p3.setLayout(grid3);
                JLabel l1 = new JLabel("Module Code");
                JLabel l2 = new JLabel("Time Slot");
                JLabel l3 = new JLabel("Room");

                //combobox to hold the module classes. Its contents will be popu
lated by the displayCourses method
                cb1 = new JComboBox<String>();

                //combobox to hold the timeslots
                cb2 = new JComboBox<String>();
                cb2.addItem("MonAM");
                cb2.addItem("MonPM");
                cb2.addItem("TueAM");
                cb2.addItem("TuePM");
                cb2.addItem("WedAM");
                cb2.addItem("WedPM");
                cb2.addItem("ThuAM");
                cb2.addItem("ThuPM");
                cb2.addItem("FriAM");
                cb2.addItem("FriPM");

                //combobox to hold the rooms
                cb3 = new JComboBox<String>();
                cb3.addItem("A");
                cb3.addItem("B");
                cb3.addItem("C");
                cb3.addItem("D");
                cb3.addItem("E");
                cb3.addItem("F");
                cb3.addItem("G");
                cb3.addItem("H");

                //buttons to facilitate program operation
                b1 = new JButton("Remove");
                b2 = new JButton("Assign");
                b3 = new JButton("Save Timetable");
                b4 = new JButton("Quit");
                b1.addActionListener(this);
                b2.addActionListener(this);
                b3.addActionListener(this);
                b4.addActionListener(this);

                p3.add(l1); p3.add(cb1);
                p3.add(l2); p3.add(cb2);
                p3.add(l3); p3.add(cb3);
                p3.add(b1); p3.add(b2);
                p3.add(b3); p3.add(b4);

                //Panel p4 will have p2 and p3 added onto it
                GridLayout grid4 = new GridLayout(1,2);
                p4.setLayout(grid4);
                p4.add(p3);

                GridLayout grid6 = new GridLayout(1,2);
                p6.setLayout(grid6);
                p6.add(p4);
                p6.add(p5);

                p1.add(p6, BorderLayout.SOUTH);
```



```
                ta1 = new JTextArea(14,36);
                ta1.setFont(new Font("Courier", Font.PLAIN, 14));
                ta1.setEditable(false);
                //Adds a scroll pane to the text area
                JScrollPane textPane = new JScrollPane(ta1);
                p2.setLayout(new BorderLayout());
                p2.add(textPane);
                p4.add(p2);

                add(p1, BorderLayout.WEST);
                add(p2, BorderLayout.EAST);
        }

        /**
         * method to read the content of the ModulesIn.txt file
         */
        private boolean readFile(){

                try {
                        FileReader reader = new FileReader(inputFile); //open th
e ModulesIn.txt file
                        Scanner scanner = new Scanner(reader);
                        while(scanner.hasNextLine()){ //while there is still a l
ine to be read
                                String line = scanner.nextLine(); //read a line
                                        tt.addModule(line); //add the module to
the timetable
                        }

                        //close file
                        reader.close();
                        scanner.close();
                        return true;
                } catch (FileNotFoundException e) {
                        JOptionPane.showMessageDialog(null, "The file was not found",
"Error", JOptionPane.ERROR_MESSAGE); //notify user that file was not found
                        return false;
                } catch (IOException e){
                        JOptionPane.showMessageDialog(null, "The file could not be opene
d", "Error", JOptionPane.ERROR_MESSAGE); //notify user that file could not be ope
ned
                        return false;
                }

        }

        /**
         * method to display the courses in the text area
         */
        private void displayCourses()
        {
                String courses = "";
                courses += String.format("%-10s%-10s%-8s%-8s%n", "Code","Time","R
oom","Size");
                for(Module mo: m){
                        if(m != null){
                                courses += String.format("%-10s%-10s%-8s%-8s%n",
 mo.getModuleCode(), mo.getTimeSlot(),
                                                mo.getRoom(), mo.getClassSize())
;
                                cb1.addItem(mo.getModuleCode()); //populate the
module code JCombobox to contain the module codes read from the ModulesIn.txt fi
```

```
le
                        }
                }

                ta1.setText(courses); //display the courses in the text area
        }

        /**
         * method to update table with timetable contents
         */
        private void fillTable(){
                clearTable(); //clear the table contents

                //the table contents are then refilled after clearing to display
 the updated module assignments
                for(Module mo: m){
                        if(mo != null){
                                String time = mo.getTimeSlot();
                                char room = mo.getRoom();

                                int row = getIndexForTimeSlot(time);
                                int col = getIndexForRoom(room);

                                if(row != -1 && col != -1)
                                        rowData[row][col] = mo.getModuleCode();
                                AbstractTableModel tm = (AbstractTableModel)tabl
e.getModel();
                                tm.fireTableDataChanged(); //tells table to upda
te its display
                        }
                }
        }

        /**
         * method to clear table contents
         */
        private void clearTable(){

                for(int row=0; row<10; row++)
                        for(int col=1; col<9; col++)
                                rowData[row][col] = "";
        }

        /**
         * helper method to get what index of the table row a time slot belongs
to
         * @param ts is the time slot whose index is required
         * @return the required index
         */
        private int getIndexForTimeSlot(String ts)
        {
                int row;
                switch(ts){
                case "MonAM":
                        row = 0;
                        break;
                case "MonPM":
                        row = 1;
                        break;
                case "TueAM":
                        row = 2;
                        break;
```



```
                case "TuePM":
                        row = 3;
                        break;
                case "WedAM":
                        row = 4;
                        break;
                case "WedPM":
                        row = 5;
                        break;
                case "ThuAM":
                        row = 6;
                        break;
                case "ThuPM":
                        row = 7;
                        break;
                case "FriAM":
                        row = 8;
                        break;
                case "FriPM":
                        row = 9;
                        break;
                default:
                        row = -1;
                        break;
                }
                return row;
        }

        /**
         * gets the index of the table column that a room belongs to
         * @param r is the room whose index is required
         * @return the required index
         */
        private int getIndexForRoom(char r)
        {
                int col;
                switch(r){
                case 'A':
                        col = 1;
                        break;
                case 'B':
                        col = 2;
                        break;
                case 'C':
                        col = 3;
                        break;
                case 'D':
                        col = 4;
                        break;
                case 'E':
                        col = 5;
                        break;
                case 'F':
                        col = 6;
                        break;
                case 'G':
                        col = 7;
                        break;
                case 'H':
                        col = 8;
                        break;
                default:
```

```
                        col = -1;
                        break;
                }
                return col;
        }

        /**
         * method to handle event generated when any of the buttons is clicked
         */
        public void actionPerformed(ActionEvent e)
        {
                //get the selected entries from the combo boxes
                String code = (String)cb1.getSelectedItem();
                String time = (String)cb2.getSelectedItem();
                String room = (String)cb3.getSelectedItem();

                Object source = (JButton)e.getSource();

                //if assign button is clicked, module is to be assigned a time s
lot and a room
                if(source == b2)
                {

                        if(validateInput(code, time, room)) //check if the selec
ted entries passes the required validations
                        {
                                Module m = tt.getModuleByCode(code);
                                m.setRoom(room.charAt(0));
                                m.setTimeSlot(time);
                                fillTable();
                                displayCourses();
                        }
                }

                //If the save button is clicked, calls the saveOutput() method t
o save the timetable data to a text file
                else if(source == b3)
                        saveOutput();

                /*
                  If the quit button is pressed, will ask the user if they have
saved the timetable first.
                  If they click YES, quit the program. Else if NO is pressed, wi
ll ask the user if they wish to save the timetable.
                  If Yes is clicked, saves the timetable and quits the program.
Else, quits the program without saving
                */
                else if(source == b4)
                {
                        handleQuitButton();
                }

                //if the remove button is clicked, remove the room and time slot
 assigned to the module
                else if(source == b1)
                {
                        removeModule(code);
                }
        }

        /**
         * method that removes the room and time slot assigned to a module
```



```
         * @param code
         */
        private void removeModule(String code){

                tt.removeModuleFromRoom(code);
                tt.removeModuleFromTimeSlot(code);
                fillTable();
                displayCourses();
        }

        /**
         * method that handles what happens when the quit button or the window c
lose button is clicked.
         */
        private void handleQuitButton(){
                int confirm = JOptionPane.showConfirmDialog(null, "Are you sure you
want to exit the program?", "Exit Program?",  JOptionPane.YES_NO_OPTION);

                if(confirm == JOptionPane.YES_OPTION){
                        saveOutput();
                        System.exit(0);
                }
        }

        /**
         * Method to process saving the data from the timetable to a text file c
alled ModulesOut.txt
         */
        private void saveOutput()
        {

                String output = "";
                //loop to extract details of each module from the module array
                for(Module mo: m)
                {
                        if(mo != null)
                                output += String.format("%s %s %s %s %s%n", mo.ge
tModuleCode(), mo.getModuleTitle(), mo.getTimeSlot(), mo.getRoom(), mo.getClassS
ize());
                }

                try
                {
                        FileWriter writer = new FileWriter(outputFile); //create
 an output file
                        writer.write(output); //write the data to output file
                        writer.close(); //close the file
                }
                catch (IOException x)
                {
                        JOptionPane.showMessageDialog(null, "Output Error", "Error: Fil
e I/O Error", JOptionPane.ERROR_MESSAGE);
                }
        }

        /**
         * validates the inputs selected to check if the required criteria are m
et
         * @param module to be acted upon
         * @param time to be assigned to module
         * @param room to be assigned to module
         * @return validation result
```

```
         */
        private boolean validateInput(String code, String time, String room)
        {
                Module module = tt.getModuleByCode(code);
                if(validateRoomCapacity(module, room)) //checks if the room is b
ig enough to accommodate the class size of the module
                        if(validateCourseSubjectAndLevel(module, time))
//checks if there is another module with the same level already taking place on
the time slot to be assigned
                                if(validateFreeTimeSlotAndRoom(module, t
ime, room)) //checks if the selected time slot and room have another module assi
gned to it
                                        return true; //if all validation
s are passed, return true
                return false; //else return false
        }

        /**
         * method to get the capacity of a room
         * @param r is the room whose capacity is required
         * @return the required capacity
         */
        private int getRoomCapacity(String r)
        {
                int capacity;
                switch(r){
                case "A":
                        capacity = ROOM_A_SIZE;
                        break;
                case "B":
                        capacity = ROOM_B_SIZE;
                        break;
                case "C":
                        capacity = ROOM_C_SIZE;
                        break;
                case "D":
                        capacity = ROOM_D_SIZE;
                        break;
                case "E":
                        capacity = ROOM_E_SIZE;
                        break;
                case "F":
                        capacity = ROOM_F_SIZE;
                        break;
                case "G":
                        capacity = ROOM_G_SIZE;
                        break;
                case "H":
                        capacity = ROOM_H_SIZE;
                        break;
                default:
                        capacity = 0;
                }
                return capacity;
        }

        /**
         * method to check if the room to be assigned to a module can accommodat
e the class size
         * @param module that the room is to be assigned to
         * @param room that is to be assigned
         * @return result of validation
```



```
         */
        private boolean validateRoomCapacity(Module module, String room)
        {
                int classSize = module.getClassSize();
                int roomSize = getRoomCapacity(room);
                if(classSize>roomSize)
                {
                        JOptionPane.showMessageDialog(null, "Room is too small for class
", "Error",
                                                        JOptionPane.ERROR_MESSAGE);
                        return false;
                }
                return true;
        }

        /**
         * method to check if the time to be assigned to a module already has an
other module with the same subject and year
         * @param module that the time is to be assigned to
         * @param time that is to be assigned
         * @return result of validation
         */
        private boolean validateCourseSubjectAndLevel(Module module, String time
)
        {
                char level1 = module.getLevel(); //get the level of the course t
o be added
                String subj1 = module.getSubject(); //get the subject of the cou
rse to be added
                ArrayList<Module> list = tt.getModuleByTime(time); //get a list
containing modules taking place at the specified time
                for(Module mo: list){ //for each module,
                        char level2 = mo.getLevel(); //get the level of that mod
ule
                        String subj2 = mo.getSubject(); //get the subject of tha
t module

                        //if the module to be assigned is the same as what was a
lready there in the time slot, validation is passed. (The operation is just a ca
se of reassignment)
                        if(mo.equals(module))
                                return true;

                        //if a module with the same level and subject with the n
ew module to be assigned exists, validation fails
                        if(subj1.equals(subj2) && level1 == level2)
                        {
                                JOptionPane.showMessageDialog(null, "There is another
 module with the same subject and year taking place at this the selected time slot",

                        "Error", JOptionPane.ERROR_MESSAGE);
                                return false;
                        }
                }
                return true;
        }

        /**
         * method to check if the chosen time slot and room is not occupied
         * @param time to be checked
         * @param room to be checked
         * @return result of validation
```

```
         */
        private boolean validateFreeTimeSlotAndRoom(Module module, String time,
String room)
        {
                ArrayList<Module> list = tt.getModuleByTime(time); //get a list
containing modules taking place at the specified time
                if(!list.isEmpty()) //if the list is not empty
                {
                        //iterate through the list
                        for(Module mo: list){
                                //if the module to be assigned is the same as wh
at is already there in the time slot and room, validation is passed. (Nothing ha
ppens)
                                if(mo.equals(module))
                                        return true;

                                //if a module with that time and room exists, va
lidation fails
                                if(mo.getRoom() == room.charAt(0))
                                {
                                        JOptionPane.showMessageDialog(null, "The
re is another module at the selected time and room. Please remove that module first or choose another time or room.",

                                "Error", JOptionPane.ERROR_MESSAGE);
                                        return false;
                                }
                        }
                }
                return true;
        }

        @Override
        /**
         * method to trigger the checks to be done before the program can be clo
sed.
         */
        public void windowClosing(WindowEvent arg0) {
                handleQuitButton();
        }


        //the methods below were inherited from the WindowListener interface and
 were not needed
        @Override
        public void windowActivated(WindowEvent arg0) {}

        @Override
        public void windowClosed(WindowEvent arg0) {}

        @Override
        public void windowDeactivated(WindowEvent arg0) {}

        @Override
        public void windowDeiconified(WindowEvent arg0) {}

        @Override
        public void windowIconified(WindowEvent arg0) {}

        @Override
        public void windowOpened(WindowEvent arg0) {}

}
```