A Project Report on

Master Time-Table Generator

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Computer Engineering

by

Suraj Thakkar(16102057)
Neel Dhruva(17202003) Adesh
Thosani(17202001)

Under the Guidance of

Dr. Rahul Ambekar

Department of Branch Name
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615 UNIVERSITY OF MUMBAI

Academic Year 2019-2020

Approval Sheet

This Project Report entitled *“Master Time-Table Generator”* Submitted by *“Suraj Thakkar”(16102057),“Neel Dhruva”(17202003),“Adesh Thosani”(17202001)*is approved for the partial fulfillment of the requirenment for the award of the degree of *Bachelor of Engineering* in *Computer Engineering* from *University of Mumbai*.

(Dr. Rahul Ambekar) Guide

Prof. Sachin Malve
Head Department of Computer Engineering

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled *"Title of project"* submitted by *"Suraj Thakar" (16102057),"Neel Dhruva" (17202003),"Adesh Thosani" (17202001),* for the partial fulfillment of the requirement for award of a degree *Bachelor of Engineering* in *Computer Engineering*,to the University of Mumbai,is a bonafide work carried out during academic year 2019-2020.

(Dr. Rahul Ambekar)	(Prof. Sachin Malave)
Guide	Head of Computer Department

Dr. Uttam Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Suraj Thakkar 16102057)
(Neel Dhruva 17202003)
(Adesh Thosani 17202001)

Date:

Abstract

Many colleges use manual process of preparing academic timetables with large number of constraints and this is very time consuming. This usually ends up with various courses clashing, this may be either at same room or with same teachers having more than one course at a time. These are just due to common human errors which are very difficult to prevent and also a very hectic and cumbersome activity. To overcome all these problems we propose this application of Master Time Table Generator. There are many departments in a college/institute and each department has its own time table with sharing some common resources like rooms, infrastructure and faculty also , the master time table will generate the copy of time table integrating all time tables of various departments considering the needs of the said departments. It will be implemented with the information required such as Faculty Name, Faculty Load,Preference of the Time slot, Required hours for the faculty. The proposed application is intended to be used in any institutes irrespective of their discipline and number of departments.

Contents

List of Abbreviations

GA:	Genetic Algorithms

Chapter 1

Introduction

Most colleges have a number of different courses and each course has a number of subjects. Now there are limited faculties, each faculty teaching more than one subjects. So now the time table needed to schedule the faculty at provided time slots in such a way that their timings do not overlap and the time table schedule makes best use of all faculty subject demands. Timetable generator automatically schedules timetable for students and faculty which reduces the manual work.Once the inputs like faculty with their respective subjects are given it will generate the period slots for the given semester.

Objectives

1. The final system should be able to generate time table in completely automated way which will save lot of time and effort of an institute.

2. User defined constraints handling.

3. Ease of use of user of system so that he/she can make automated time table.

4. Focus on optimization of resources.

5. Generate multiple useful views from the timetable.

6. Managing all the information about semester course, faculty, and classrooms.

6. Building the application to reduce the manual handwork load.

Benefits for Society

1. It is complicated task that to handle many Faculty's and allocating subjects for them ata time physically.

2. So our proposed system will help to overcome this disadvantage. Thus we can produce timetable for any number of courses and multiple semesters.

3. Separate timetable for the individual class, faculty and labs are generated automatically by this system.

4. The project reduces time consumption and the pain in framing the timetable manually.

5. The project is developed in such a way that, no slot clashes occur providing features to tailor the timetable as of wish.

6. It will save time and efforts. It will help to reduce errors. After Successful Completion of the project, The Master Timetable can be used in many different Institutes.

The main goal and solution for the Problem

The main goal is to build the Master timetable that creates the time table without overlapping of lectures. This will be solved with he help of master time table generator project by using different technologies and algorithms.

Chapter 2

Literature Review

Automated Time Table Generation Using Multiple Context Reasoning for University Modules

.

Finding a feasible lecture/tutorial timetable in a large university department is a challenging problem faced continually in educational establishments. This paper presents an evolutionary algorithm (EA) based approach to solving a heavily constrained university timetabling problem. The approach uses a problem-specific chromosome representation. Heuristics and context-based reasoning have been used for obtaining feasible timetables in a reasonable computing time.

Automatic Time Table Generator1Saritha M, 2Pranav Kiran Vaze, 3Pradeep, 4Mahesh N R1Assistant Professor,2, 3, 4 UG Scholar1, 2, 3, 4 Department

of CSE, SDMIT Ujire, Karnataka, India

The manual system of preparing time table in colleges with large number of students is very time consuming and usually ends up with various classes clashing either at same room or with same teachers having more than one class at a time.to overcome all these problems , propose to make an automated system.The system will take various inputs like details of students,subjects and class rooms and teachers available,depending upon these inputs it will generate a possible time tabble,making optimal utilization of all resources in a way that will best suit any of constraints or college rules.List of subjects may include electives as well as core subjects.

Automatic Timetable Generation System1Deeksha C S, 2A Kavya Reddy, 3Nagambika A, 4Akash Castelino, 5K Panimozhi1,2,3,4UG Student, 5Assistant Professor

This paper discusses the various approaches thatcan be taken to solve the timetable generation problem. Timetable generation is basically a constraint satisfaction problem. The methods discussed in this paper are capable of handling both soft and hard constraints

A time table is an organised list, usually set out in tabular form, providing information about the series of arranged events in a particular, the time at which it is planned this events will take place. They are applicable to any institute where the activitied have been carried out by various individuals in a specified time frame. From the time institutes became organised environments, time tables have been the framework for all institutes activities. As a result institute has devoted time, energy and human capital to the implementation of nearly optimisation time tables which must be to satisfy all required constraints as specified by faculties. The lecture timetabling problem is a tyupical scheduling problem that appears to be a tedious job in every academic institute twice a year. The problem involves the scheduling of classes, Students, Teachers and rooms at fixed number of timeslots, Subject to a certain number of constraints. An effective time table is crusial for the satisfaction of educational requirments and efficient utilisation of human resourses which make an optimisation problem.

Chapter 3

Flow of Modules

1. Department Level Module
2. Faculty and Resource allocation Module
3. Admin Module
4. Integration of department Module
5. Master Time-Table Module

Department Level Module

Firstly the timetable will be generated for one department. There will be different inputes taken such as semester name, Faculty name, different Time slots.

Faculty and resource allocation Module

The faculty gives all of their details to the admin In the case, at times the faculty could take a leave as well In times like these, the facility is responsible to send the reason, date and on which period the leave is to be taken. The substitute faculty gets the request. The substitute faculty has the facility to either accept or reject the substitute hour. Then this is sent back to the faculty informing about the request. According to the timetable is modified

Admin Module

The admin is responsible for taking all the details of the faculty, course, subject, semester and how many hours a day the classes last. The admin generates the timetable according to all these factors.

Integration Module

Above all the created department level timetables will be integrated to form a master timetable.

Master Time-Table Module

In this module, generation is done by considering the maximum and minimum workload for each faculty. This will be generated by the admin and viewed by the faculty who are the users of this system.

Chapter 4

Project Design

Proposed System

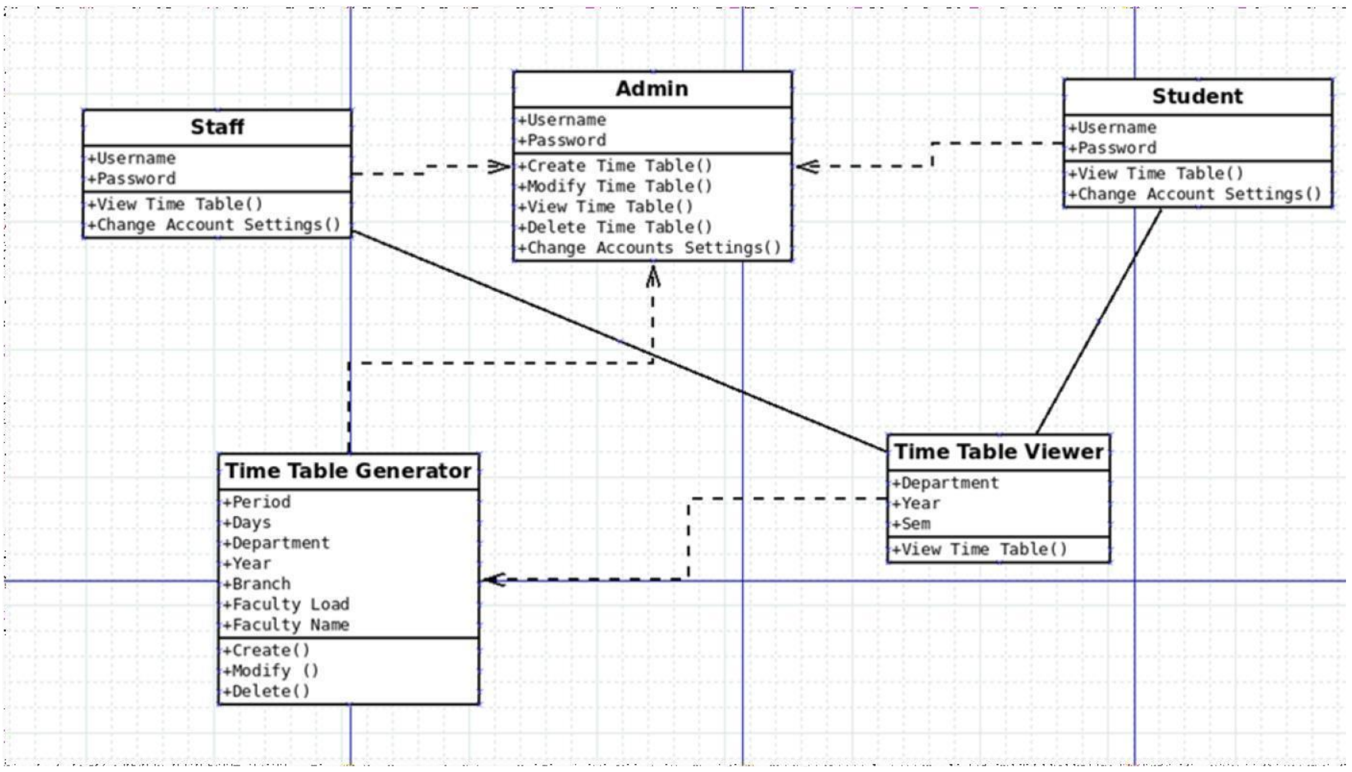
Proposed System

The proposed system is designed to be more efficient than the actual manual system. Most colleges have a number of different courses and each course has ‘n’ number of subjects. Now there are limited faculties, and each faculty might be teaching more than one subjects. So now the time table needed to schedule all the faculty at provided time slots in such a way that their timings do not overlap and the time table schedule will make the best use of all faculty subject demands.

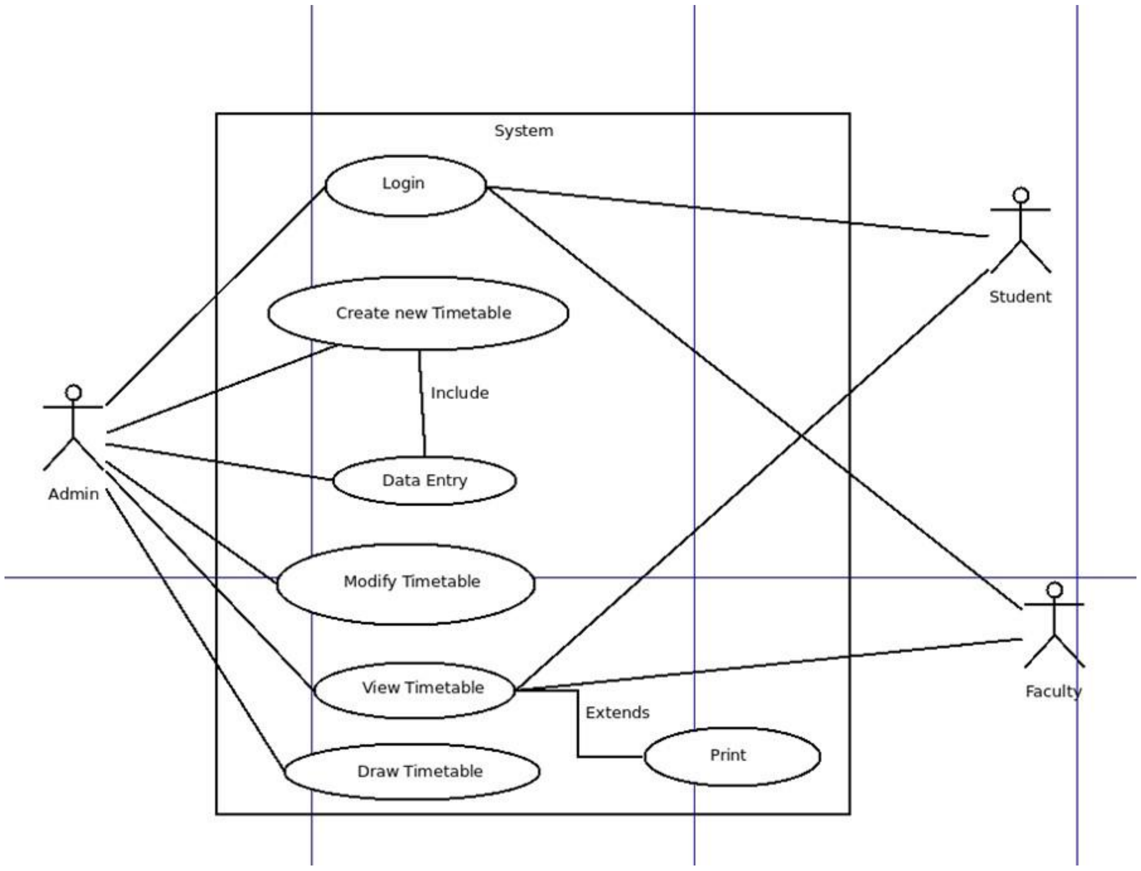
Description of Class Diagram

Description of Class Diagram The diagram represents Class Diagram for Time Table Generator system. Here student , faculty, timetable generator were depend on administrator because the administrator can able to modify the details of the students, Faculty and timetable. So dependency relationship is exist between students and administrator, Faculty and administrator. This dependency is also exist between the timetable generator and timetable viewer. since because we can view timetable only after it is generated.

Class Diagram



Use Case Diagram



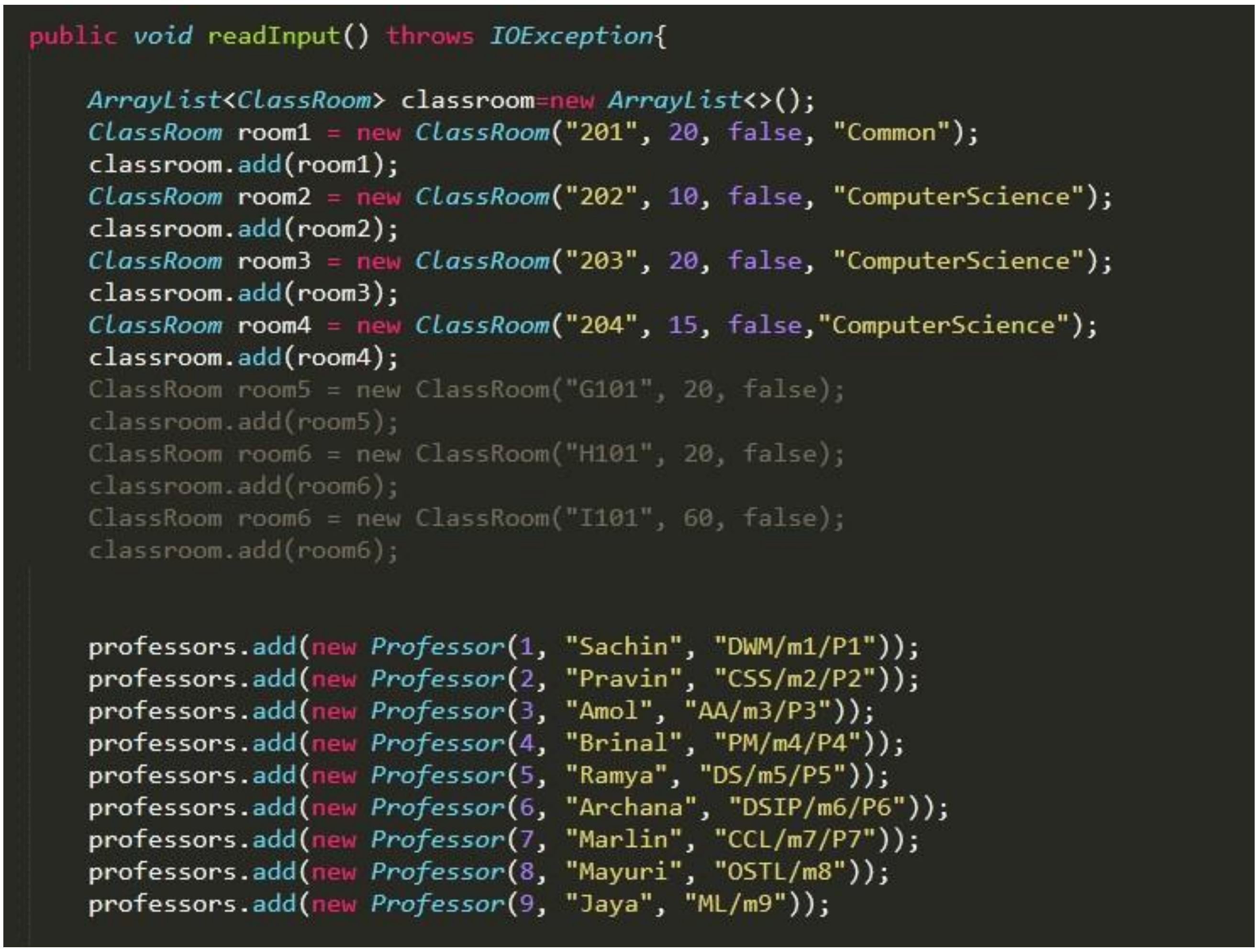
Chapter 5

Implementation

In this project we have used Genetic Algorithm. Genetic algorithm works as follows:-

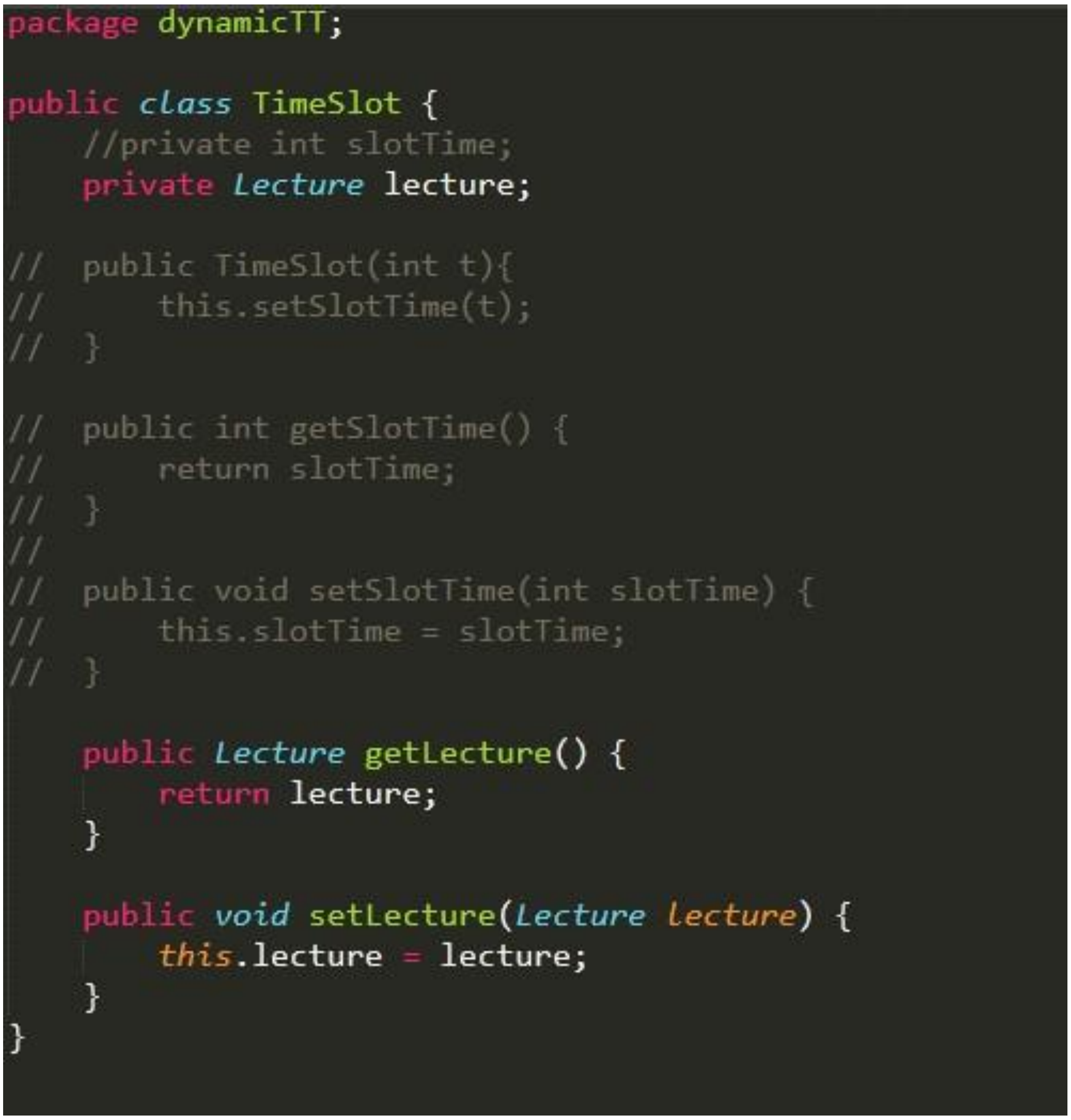
Fitness function finds number of lectures clashing between rooms.The timetable with less fitness score is the best. The crossover does nothing but shuffles the 'TimeSlots' for the 'Day'. The 'Day' is picked from a random 5 days of the week. These timeslots are maintained in a list and shuffling the timeslots of days in all the classroom produces a new child Time table.

In this we give Department,Room Number,Faculty Name,Subjects(Along with Faculty name),And Number of Lectures required in a week as a input and we get the generated output for each class.



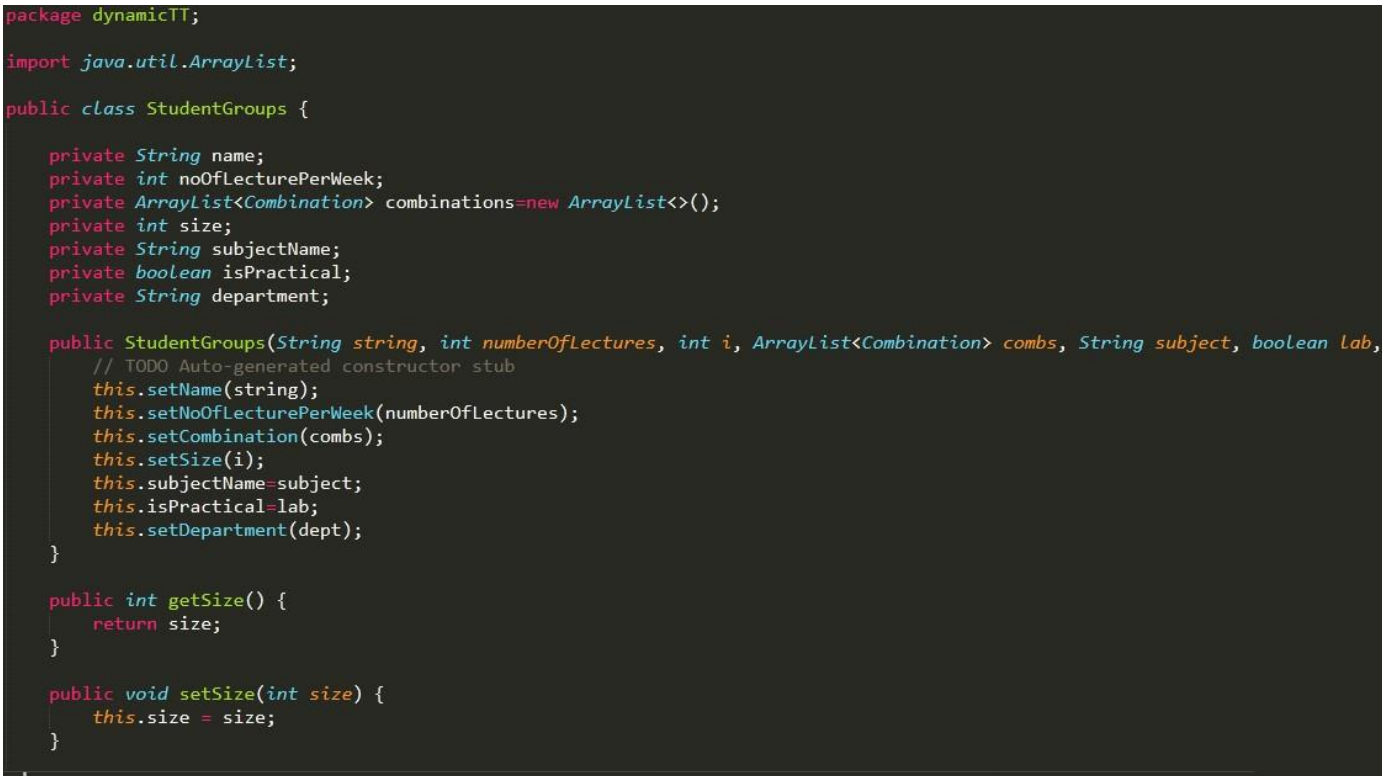
```
public void readInput() throws IOException{

    ArrayList<ClassRoom> classroom=new ArrayList<>();
    ClassRoom room1 = new ClassRoom("201", 20, false, "Common");
    classroom.add(room1);
    ClassRoom room2 = new ClassRoom("202", 10, false, "ComputerScience");
    classroom.add(room2);
    ClassRoom room3 = new ClassRoom("203", 20, false, "ComputerScience");
    classroom.add(room3);
    ClassRoom room4 = new ClassRoom("204", 15, false,"ComputerScience");
    classroom.add(room4);
    ClassRoom room5 = new ClassRoom("G101", 20, false);
    classroom.add(room5);
    ClassRoom room6 = new ClassRoom("H101", 20, false);
    classroom.add(room6);
    ClassRoom room6 = new ClassRoom("I101", 60, false);
    classroom.add(room6);


    professors.add(new Professor(1, "Sachin", "DWM/m1/P1"));
    professors.add(new Professor(2, "Pravin", "CSS/m2/P2"));
    professors.add(new Professor(3, "Amol", "AA/m3/P3"));
    professors.add(new Professor(4, "Brinal", "PM/m4/P4"));
    professors.add(new Professor(5, "Ramya", "DS/m5/P5"));
    professors.add(new Professor(6, "Archana", "DSIP/m6/P6"));
    professors.add(new Professor(7, "Marlin", "CCL/m7/P7"));
    professors.add(new Professor(8, "Mayuri", "OSTL/m8"));
    professors.add(new Professor(9, "Jaya", "ML/m9"));
```

Time Slots

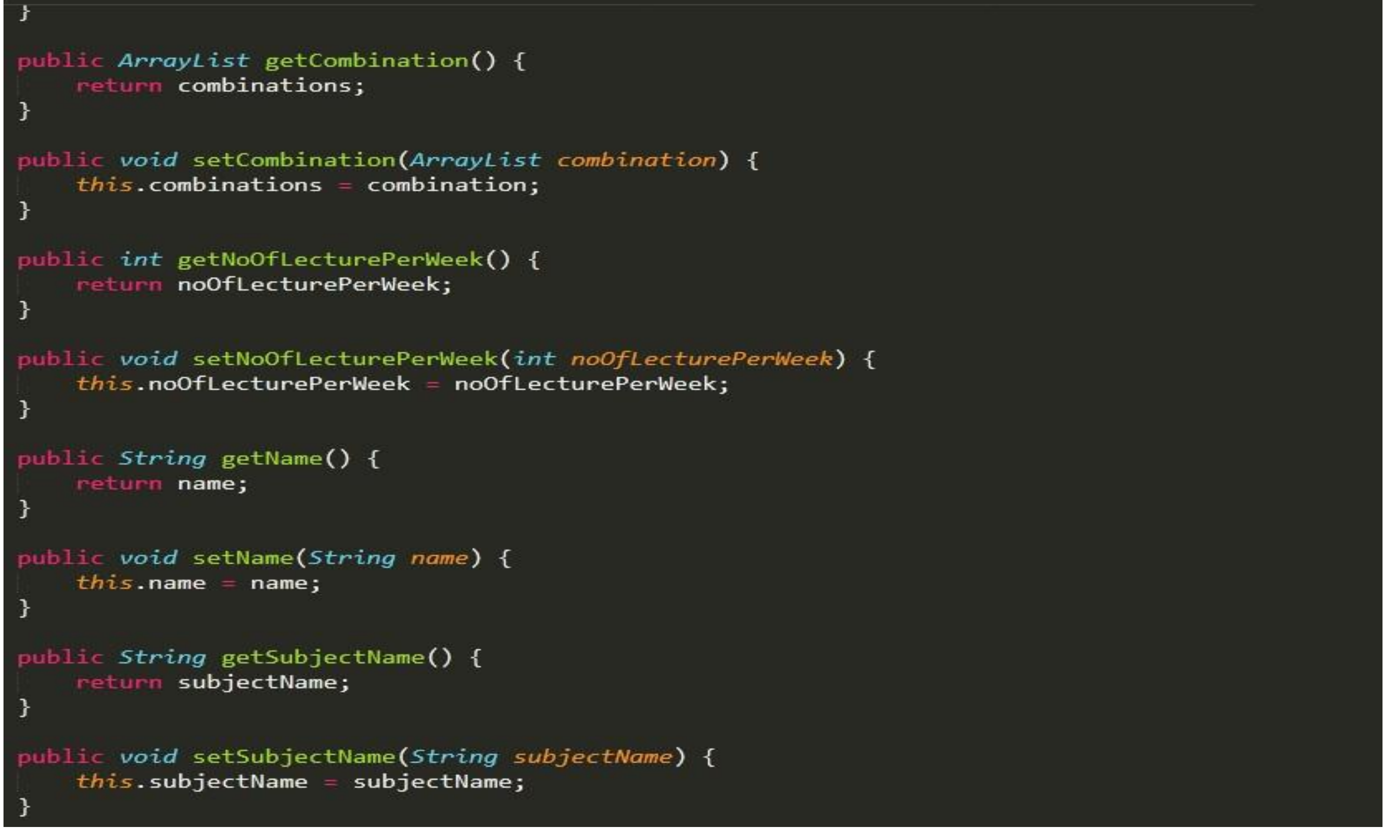


```java
package dynamicTT;

public class TimeSlot {
    //private int slotTime;
    private Lecture lecture;

//  public TimeSlot(int t){
//      this.setSlotTime(t);
//  }

//  public int getSlotTime() {
//      return slotTime;
//  }
//
//  public void setSlotTime(int slotTime) {
//      this.slotTime = slotTime;
//  }

    public Lecture getLecture() {
        return lecture;
    }

    public void setLecture(Lecture lecture) {
        this.lecture = lecture;
    }
}
```

Student Group

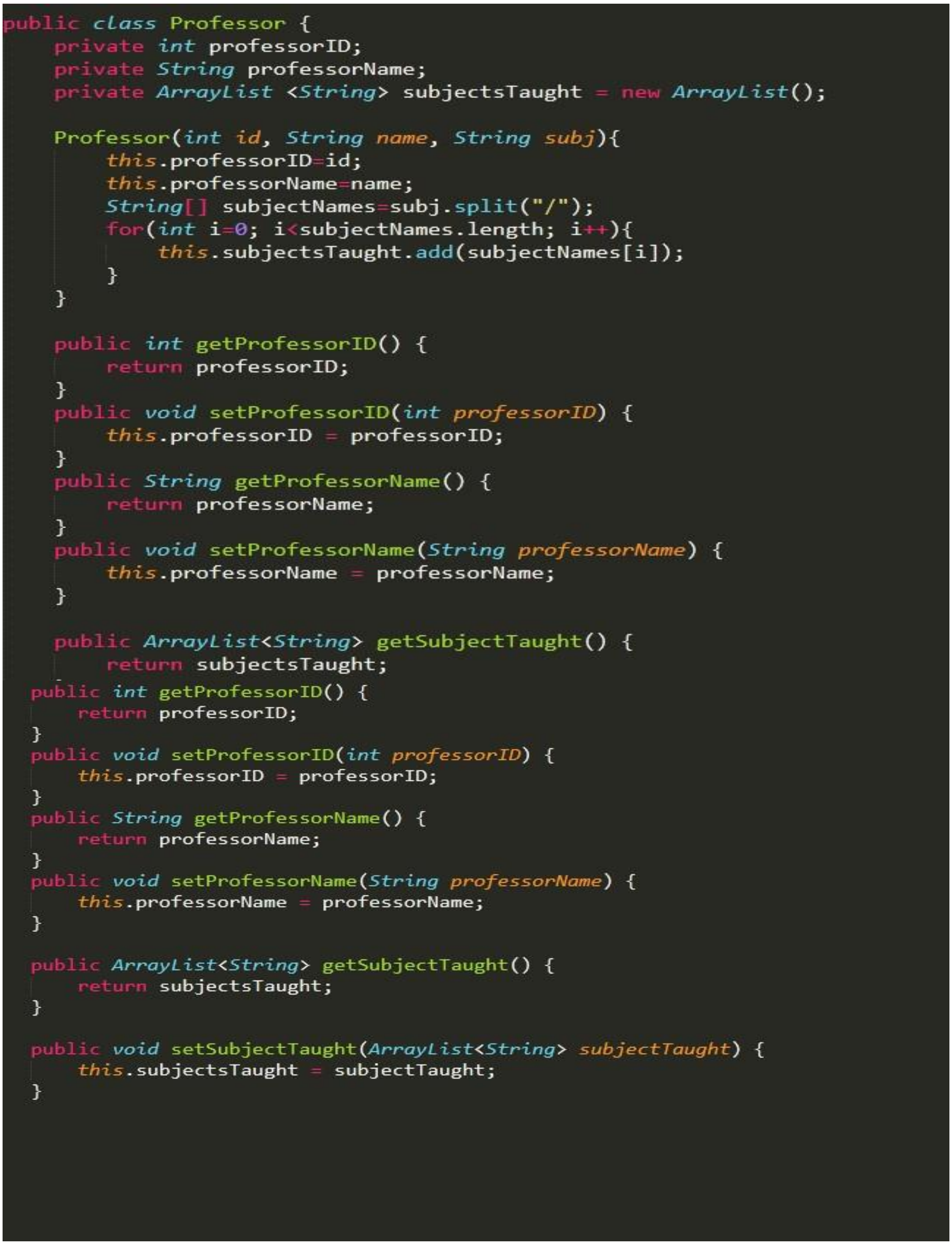


```java
package dynamicTT;

import java.util.ArrayList;

public class StudentGroups {

    private String name;
    private int noOfLecturePerWeek;
    private ArrayList<Combination> combinations=new ArrayList<>();
    private int size;
    private String subjectName;
    private boolean isPractical;
    private String department;

    public StudentGroups(String string, int numberOfLectures, int i, ArrayList<Combination> combs, String subject, boolean lab,
        // TODO Auto-generated constructor stub
        this.setName(string);
        this.setNoOfLecturePerWeek(numberOfLectures);
        this.setCombination(combs);
        this.setSize(i);
        this.subjectName=subject;
        this.isPractical=lab;
        this.setDepartment(dept);
    }

    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }
```



```java
}

public ArrayList getCombination() {
    return combinations;
}

public void setCombination(ArrayList combination) {
    this.combinations = combination;
}

public int getNoOfLecturePerWeek() {
    return noOfLecturePerWeek;
}

public void setNoOfLecturePerWeek(int noOfLecturePerWeek) {
    this.noOfLecturePerWeek = noOfLecturePerWeek;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getSubjectName() {
    return subjectName;
}

public void setSubjectName(String subjectName) {
    this.subjectName = subjectName;
}
```

Professor

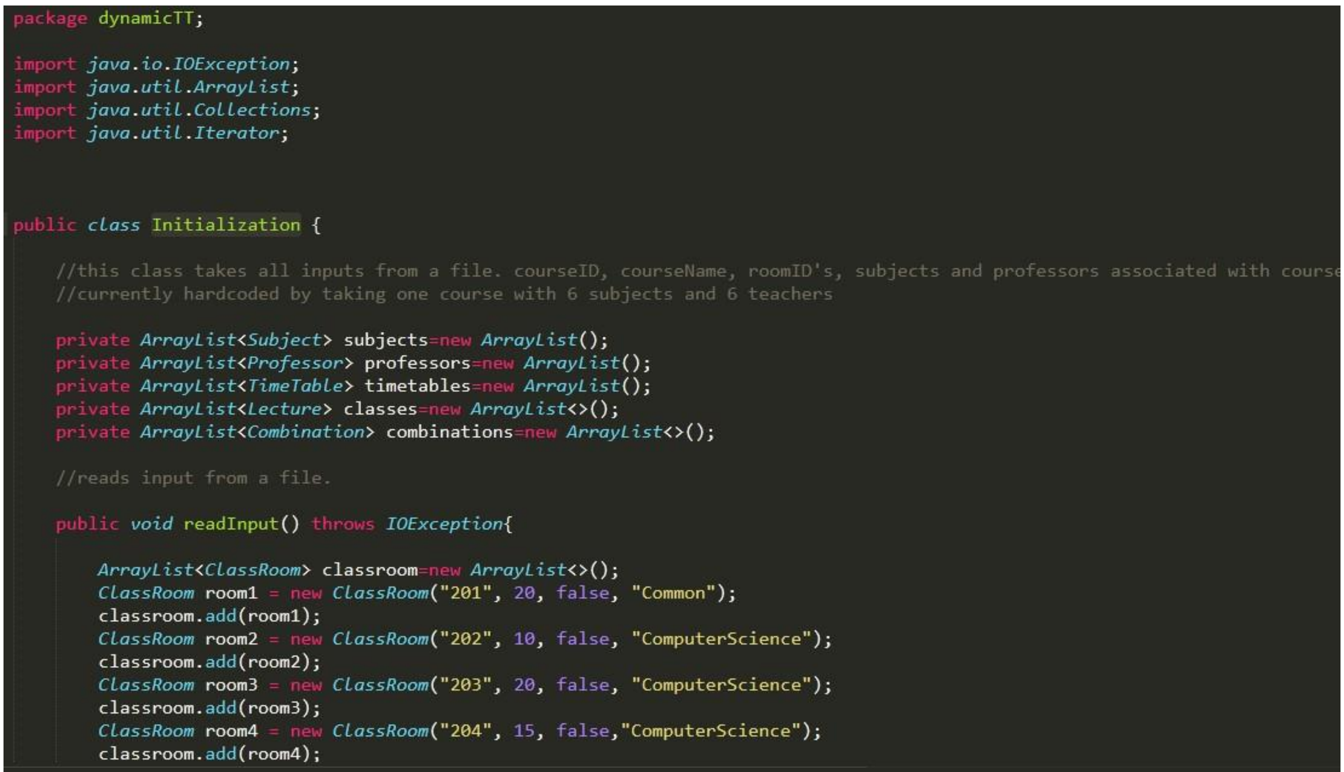


```
public class Professor {
    private int professorID;
    private String professorName;
    private ArrayList <String> subjectsTaught = new ArrayList();

    Professor(int id, String name, String subj){
        this.professorID=id;
        this.professorName=name;
        String[] subjectNames=subj.split("/");
        for(int i=0; i<subjectNames.length; i++){
            this.subjectsTaught.add(subjectNames[i]);
        }
    }

    public int getProfessorID() {
        return professorID;
    }
    public void setProfessorID(int professorID) {
        this.professorID = professorID;
    }
    public String getProfessorName() {
        return professorName;
    }
    public void setProfessorName(String professorName) {
        this.professorName = professorName;
    }

    public ArrayList<String> getSubjectTaught() {
        return subjectsTaught;
```

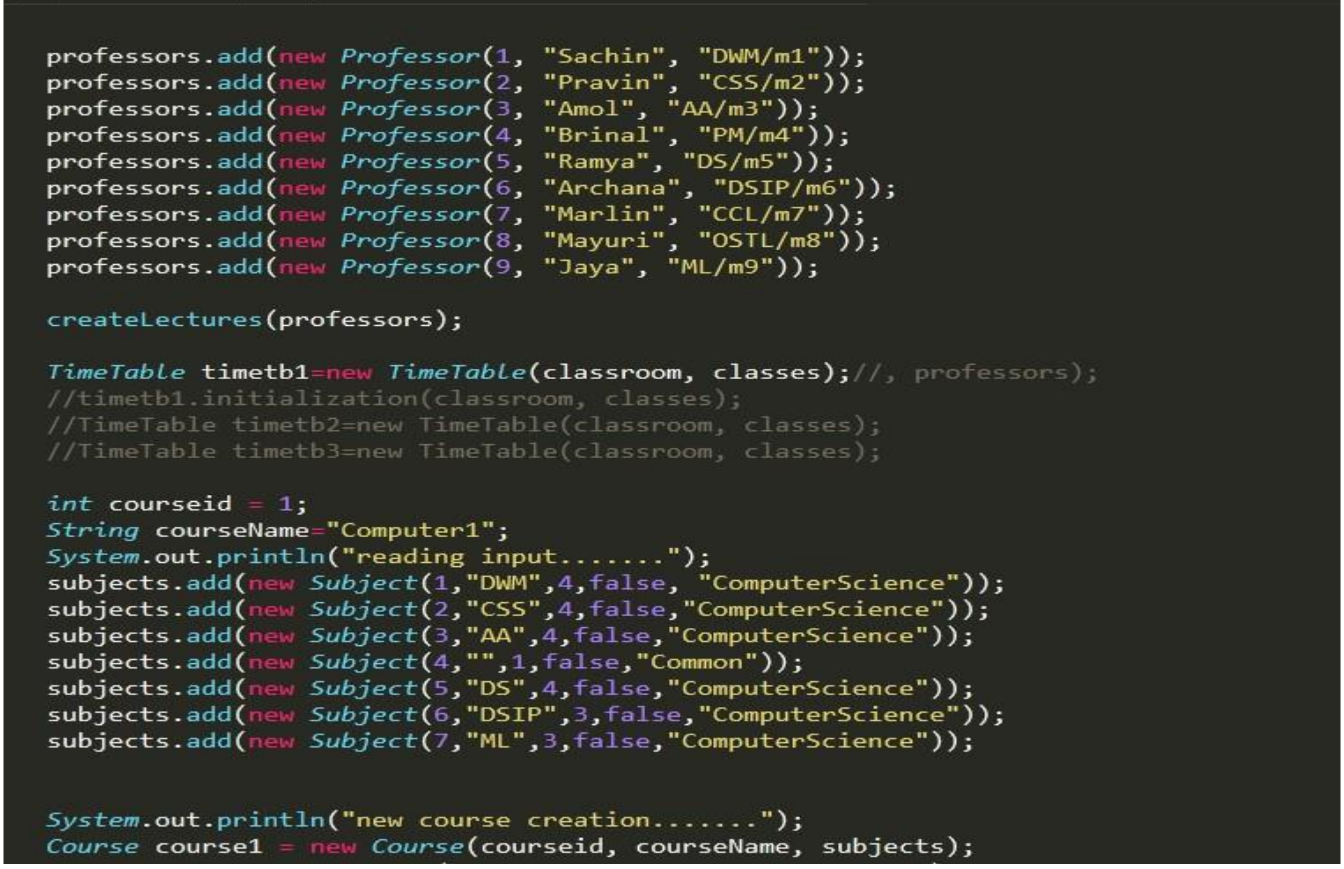
```
    public int getProfessorID() {
        return professorID;
    }
    public void setProfessorID(int professorID) {
        this.professorID = professorID;
    }
    public String getProfessorName() {
        return professorName;
    }
    public void setProfessorName(String professorName) {
        this.professorName = professorName;
    }

    public ArrayList<String> getSubjectTaught() {
        return subjectsTaught;
    }
```

```
    public void setSubjectTaught(ArrayList<String> subjectTaught) {
        this.subjectsTaught = subjectTaught;
    }
```

Initialization

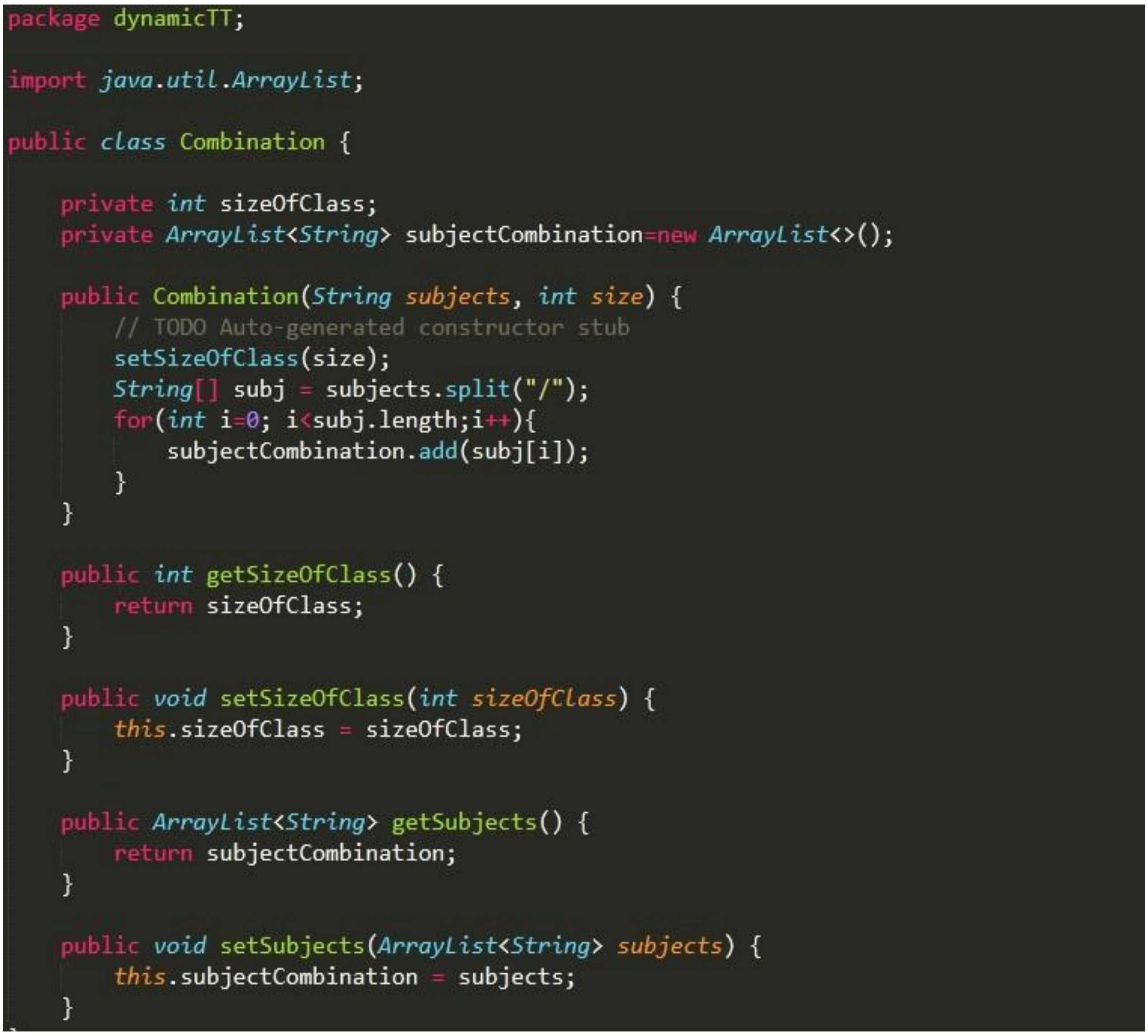


```java
package dynamicTT;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;



public class Initialization {

    //this class takes all inputs from a file. courseID, courseName, roomID's, subjects and professors associated with course
    //currently hardcoded by taking one course with 6 subjects and 6 teachers

    private ArrayList<Subject> subjects=new ArrayList();
    private ArrayList<Professor> professors=new ArrayList();
    private ArrayList<TimeTable> timetables=new ArrayList();
    private ArrayList<Lecture> classes=new ArrayList<>();
    private ArrayList<Combination> combinations=new ArrayList<>();

    //reads input from a file.

    public void readInput() throws IOException{

        ArrayList<ClassRoom> classroom=new ArrayList<>();
        ClassRoom room1 = new ClassRoom("201", 20, false, "Common");
        classroom.add(room1);
        ClassRoom room2 = new ClassRoom("202", 10, false, "ComputerScience");
        classroom.add(room2);
        ClassRoom room3 = new ClassRoom("203", 20, false, "ComputerScience");
        classroom.add(room3);
        ClassRoom room4 = new ClassRoom("204", 15, false,"ComputerScience");
        classroom.add(room4);
```



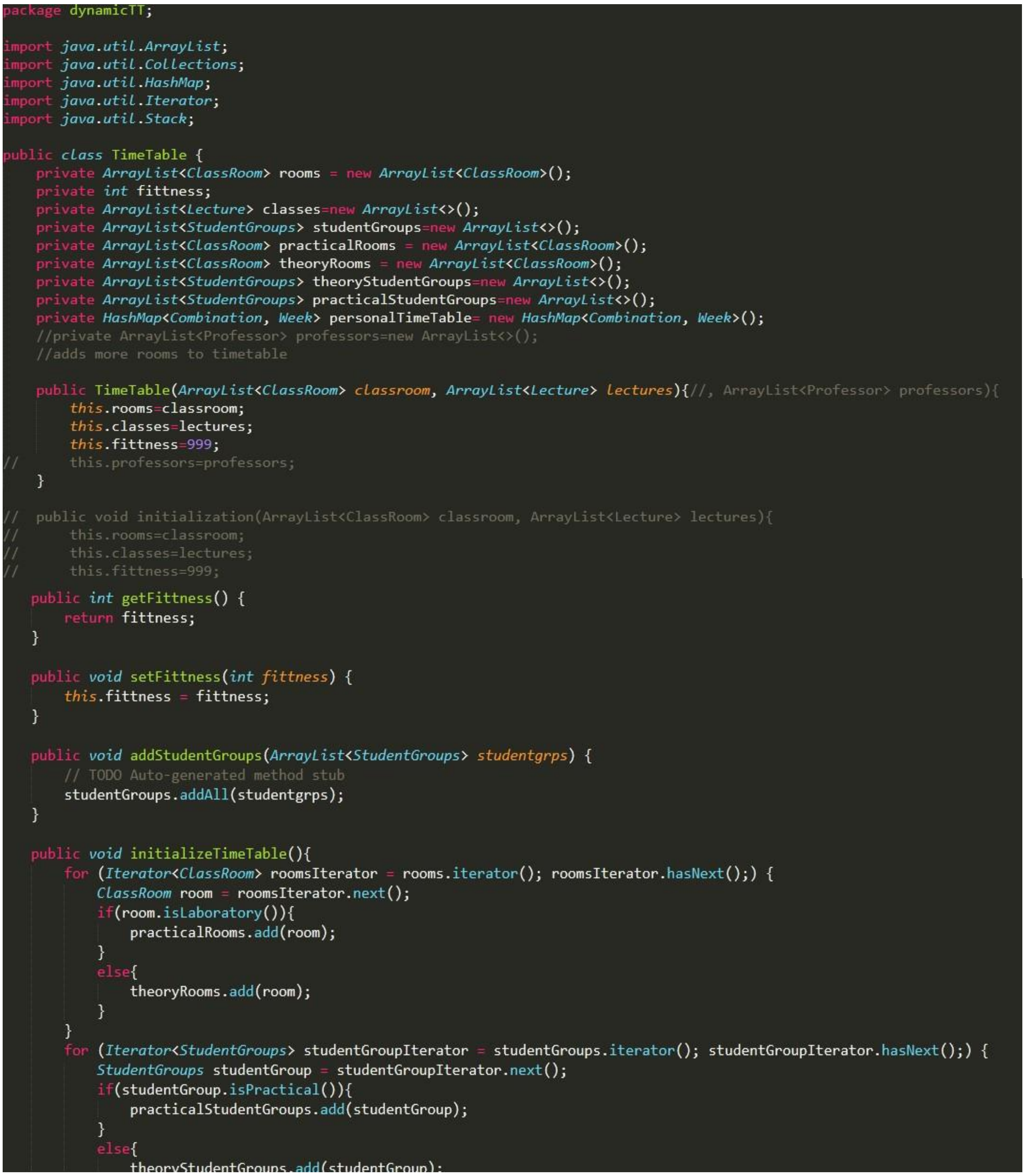
```java
        professors.add(new Professor(1, "Sachin", "DWM/m1"));
        professors.add(new Professor(2, "Pravin", "CSS/m2"));
        professors.add(new Professor(3, "Amol", "AA/m3"));
        professors.add(new Professor(4, "Brinal", "PM/m4"));
        professors.add(new Professor(5, "Ramya", "DS/m5"));
        professors.add(new Professor(6, "Archana", "DSIP/m6"));
        professors.add(new Professor(7, "Marlin", "CCL/m7"));
        professors.add(new Professor(8, "Mayuri", "OSTL/m8"));
        professors.add(new Professor(9, "Jaya", "ML/m9"));

        createLectures(professors);

        TimeTable timetb1=new TimeTable(classroom, classes);//, professors);
        //timetb1.initialization(classroom, classes);
        //TimeTable timetb2=new TimeTable(classroom, classes);
        //TimeTable timetb3=new TimeTable(classroom, classes);

        int courseid = 1;
        String courseName="Computer1";
        System.out.println("reading input.......");
        subjects.add(new Subject(1,"DWM",4,false, "ComputerScience"));
        subjects.add(new Subject(2,"CSS",4,false,"ComputerScience"));
        subjects.add(new Subject(3,"AA",4,false,"ComputerScience"));
        subjects.add(new Subject(4,"",1,false,"Common"));
        subjects.add(new Subject(5,"DS",4,false,"ComputerScience"));
        subjects.add(new Subject(6,"DSIP",3,false,"ComputerScience"));
        subjects.add(new Subject(7,"ML",3,false,"ComputerScience"));


        System.out.println("new course creation.......");
        Course course1 = new Course(courseid, courseName, subjects);
```

Combination

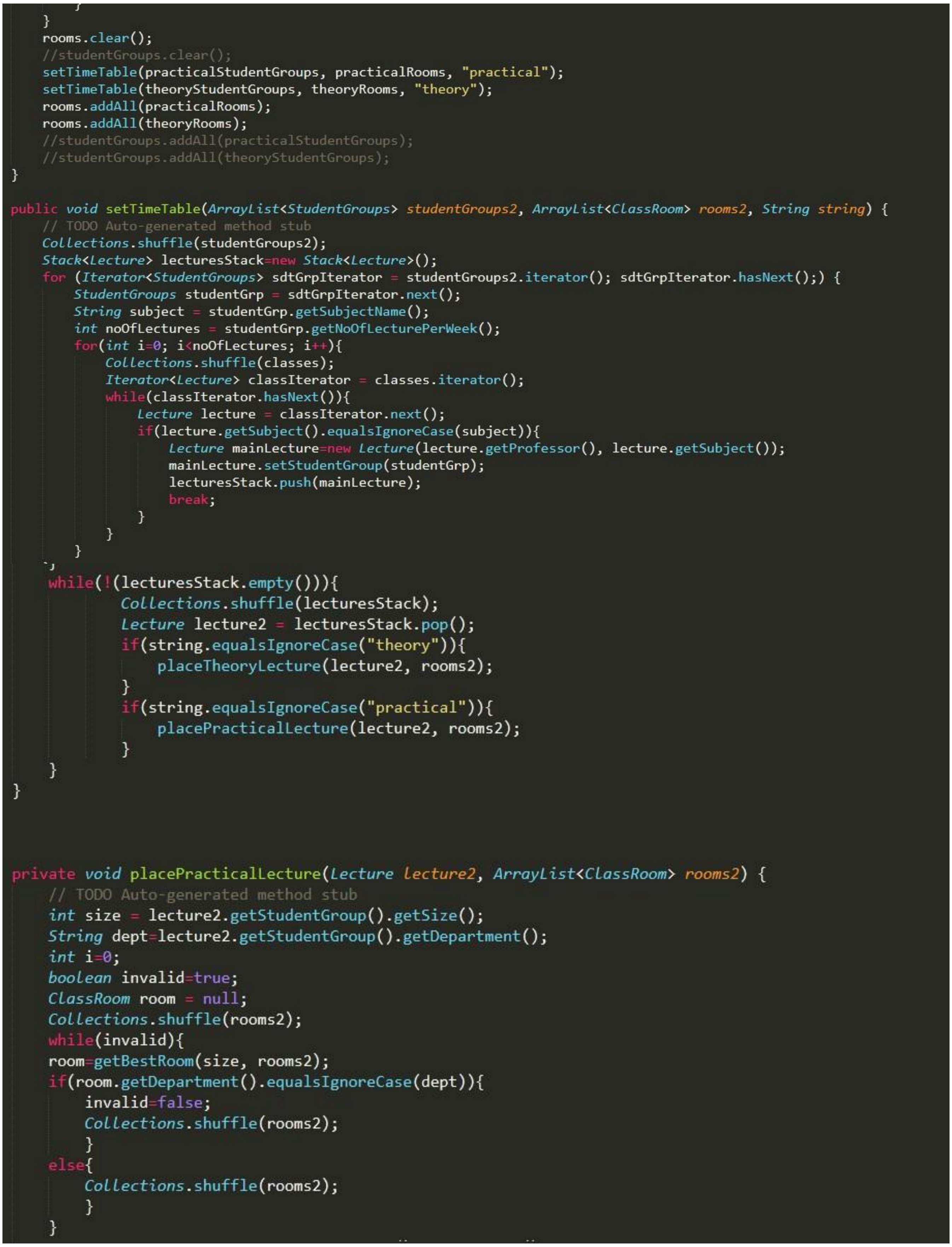


```java
package dynamicTT;

import java.util.ArrayList;

public class Combination {

    private int sizeOfClass;
    private ArrayList<String> subjectCombination=new ArrayList<>();

    public Combination(String subjects, int size) {
        // TODO Auto-generated constructor stub
        setSizeOfClass(size);
        String[] subj = subjects.split("/");
        for(int i=0; i<subj.length;i++){
            subjectCombination.add(subj[i]);
        }
    }

    public int getSizeOfClass() {
        return sizeOfClass;
    }

    public void setSizeOfClass(int sizeOfClass) {
        this.sizeOfClass = sizeOfClass;
    }

    public ArrayList<String> getSubjects() {
        return subjectCombination;
    }

    public void setSubjects(ArrayList<String> subjects) {
        this.subjectCombination = subjects;
    }
```

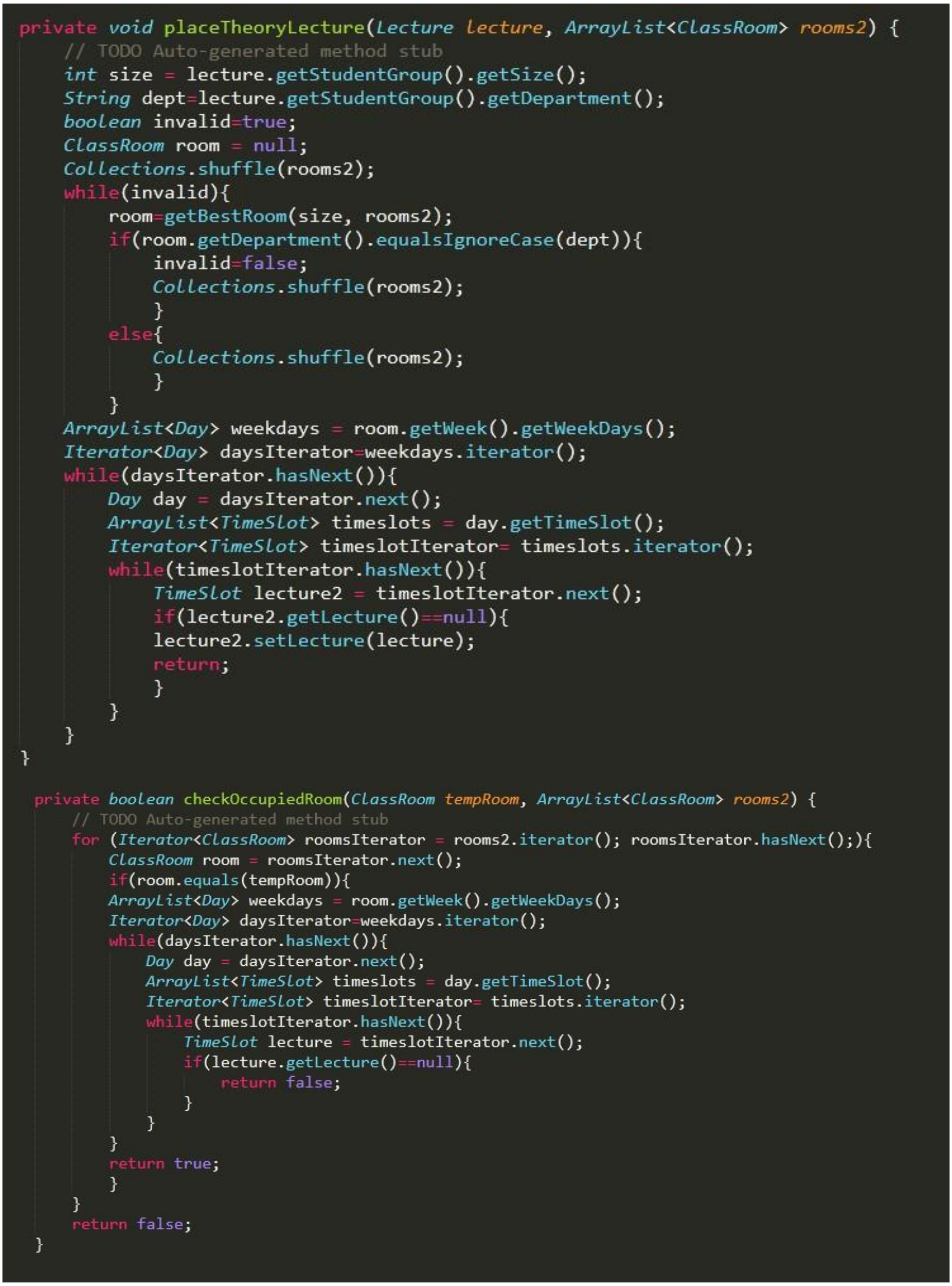
TimeTable



```
package dynamicTT;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Stack;

public class TimeTable {
    private ArrayList<ClassRoom> rooms = new ArrayList<ClassRoom>();
    private int fittness;
    private ArrayList<Lecture> classes=new ArrayList<>();
    private ArrayList<StudentGroups> studentGroups=new ArrayList<>();
    private ArrayList<ClassRoom> practicalRooms = new ArrayList<ClassRoom>();
    private ArrayList<ClassRoom> theoryRooms = new ArrayList<ClassRoom>();
    private ArrayList<StudentGroups> theoryStudentGroups=new ArrayList<>();
    private ArrayList<StudentGroups> practicalStudentGroups=new ArrayList<>();
    private HashMap<Combination, Week> personalTimeTable= new HashMap<Combination, Week>();
    //private ArrayList<Professor> professors=new ArrayList<>();
    //adds more rooms to timetable

    public TimeTable(ArrayList<ClassRoom> classroom, ArrayList<Lecture> lectures){//, ArrayList<Professor> professors){
        this.rooms=classroom;
        this.classes=lectures;
        this.fittness=999;
//      this.professors=professors;
    }

//  public void initialization(ArrayList<ClassRoom> classroom, ArrayList<Lecture> lectures){
//      this.rooms=classroom;
//      this.classes=lectures;
//      this.fittness=999;
    public int getFittness() {
        return fittness;
    }

    public void setFittness(int fittness) {
        this.fittness = fittness;
    }

    public void addStudentGroups(ArrayList<StudentGroups> studentgrps) {
        // TODO Auto-generated method stub
        studentGroups.addAll(studentgrps);
    }

    public void initializeTimeTable(){
        for (Iterator<ClassRoom> roomsIterator = rooms.iterator(); roomsIterator.hasNext();) {
            ClassRoom room = roomsIterator.next();
            if(room.isLaboratory()){
                practicalRooms.add(room);
            }
            else{
                theoryRooms.add(room);
            }
        }
        for (Iterator<StudentGroups> studentGroupIterator = studentGroups.iterator(); studentGroupIterator.hasNext();) {
            StudentGroups studentGroup = studentGroupIterator.next();
            if(studentGroup.isPractical()){
                practicalStudentGroups.add(studentGroup);
            }
            else{
                theoryStudentGroups.add(studentGroup);
```



```
        }
        rooms.clear();
        //studentGroups.clear();
        setTimeTable(practicalStudentGroups, practicalRooms, "practical");
        setTimeTable(theoryStudentGroups, theoryRooms, "theory");
        rooms.addAll(practicalRooms);
        rooms.addAll(theoryRooms);
        //studentGroups.addAll(practicalStudentGroups);
        //studentGroups.addAll(theoryStudentGroups);
    }

    public void setTimeTable(ArrayList<StudentGroups> studentGroups2, ArrayList<ClassRoom> rooms2, String string) {
        // TODO Auto-generated method stub
        Collections.shuffle(studentGroups2);
        Stack<Lecture> lecturesStack=new Stack<Lecture>();
        for (Iterator<StudentGroups> sdtGrpIterator = studentGroups2.iterator(); sdtGrpIterator.hasNext();) {
            StudentGroups studentGrp = sdtGrpIterator.next();
            String subject = studentGrp.getSubjectName();
            int noOfLectures = studentGrp.getNoOfLecturePerWeek();
            for(int i=0; i<noOfLectures; i++){
                Collections.shuffle(classes);
                Iterator<Lecture> classIterator = classes.iterator();
                while(classIterator.hasNext()){
                    Lecture lecture = classIterator.next();
                    if(lecture.getSubject().equalsIgnoreCase(subject)){
                        Lecture mainLecture=new Lecture(lecture.getProfessor(), lecture.getSubject());
                        mainLecture.setStudentGroup(studentGrp);
                        lecturesStack.push(mainLecture);
                        break;
                    }
                }
            }
        }
        while(!(lecturesStack.empty())){
                Collections.shuffle(lecturesStack);
                Lecture lecture2 = lecturesStack.pop();
                if(string.equalsIgnoreCase("theory")){
                    placeTheoryLecture(lecture2, rooms2);
                }
                if(string.equalsIgnoreCase("practical")){
                    placePracticalLecture(lecture2, rooms2);
                }
        }
    }


    private void placePracticalLecture(Lecture lecture2, ArrayList<ClassRoom> rooms2) {
        // TODO Auto-generated method stub
        int size = lecture2.getStudentGroup().getSize();
        String dept=lecture2.getStudentGroup().getDepartment();
        int i=0;
        boolean invalid=true;
        ClassRoom room = null;
        Collections.shuffle(rooms2);
        while(invalid){
        room=getBestRoom(size, rooms2);
        if(room.getDepartment().equalsIgnoreCase(dept)){
            invalid=false;
            Collections.shuffle(rooms2);
            }
        else{
            Collections.shuffle(rooms2);
            }
        }
```

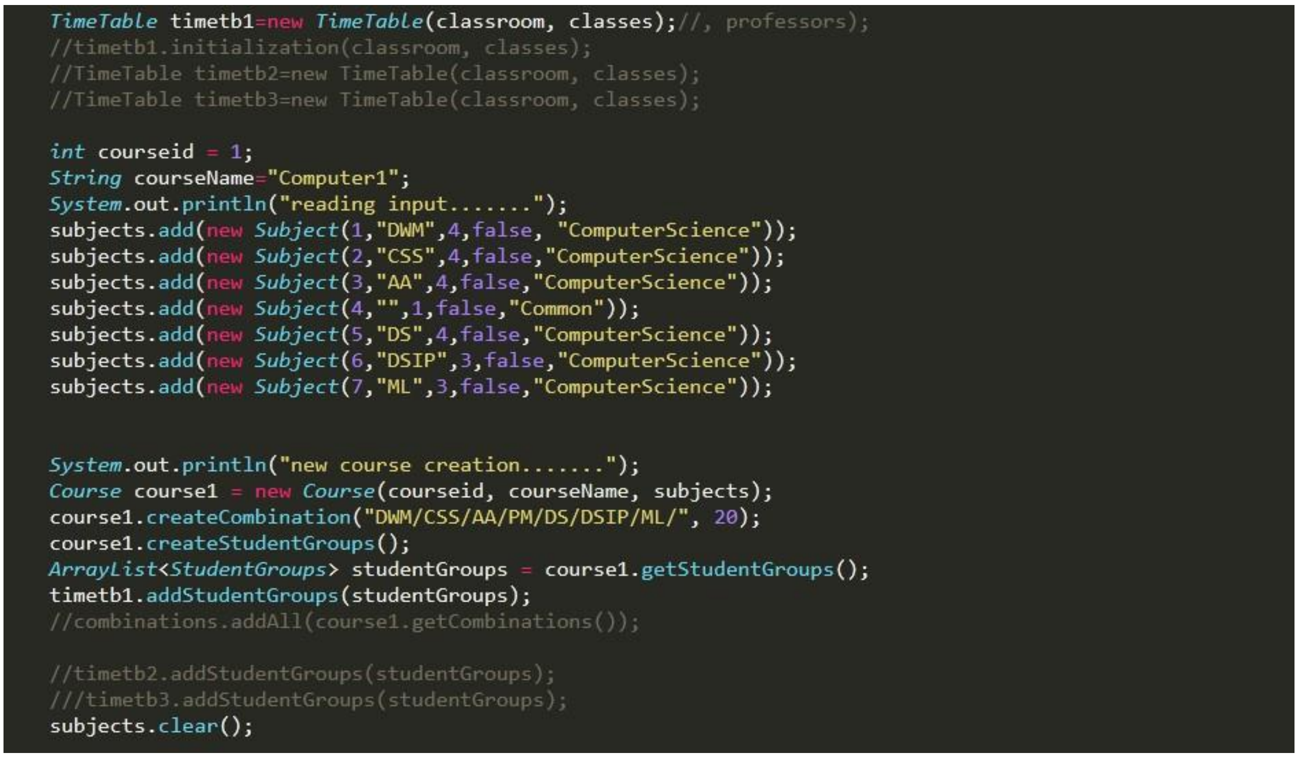


```java
private void placeTheoryLecture(Lecture lecture, ArrayList<ClassRoom> rooms2) {
    // TODO Auto-generated method stub
    int size = lecture.getStudentGroup().getSize();
    String dept=lecture.getStudentGroup().getDepartment();
    boolean invalid=true;
    ClassRoom room = null;
    Collections.shuffle(rooms2);
    while(invalid){
        room=getBestRoom(size, rooms2);
        if(room.getDepartment().equalsIgnoreCase(dept)){
            invalid=false;
            Collections.shuffle(rooms2);
            }
        else{
            Collections.shuffle(rooms2);
            }
        }
    ArrayList<Day> weekdays = room.getWeek().getWeekDays();
    Iterator<Day> daysIterator=weekdays.iterator();
    while(daysIterator.hasNext()){
        Day day = daysIterator.next();
        ArrayList<TimeSlot> timeslots = day.getTimeSlot();
        Iterator<TimeSlot> timeslotIterator= timeslots.iterator();
        while(timeslotIterator.hasNext()){
            TimeSlot lecture2 = timeslotIterator.next();
            if(lecture2.getLecture()==null){
            lecture2.setLecture(lecture);
            return;
            }
        }
    }
}

 private boolean checkOccupiedRoom(ClassRoom tempRoom, ArrayList<ClassRoom> rooms2) {
    // TODO Auto-generated method stub
    for (Iterator<ClassRoom> roomsIterator = rooms2.iterator(); roomsIterator.hasNext();){
        ClassRoom room = roomsIterator.next();
        if(room.equals(tempRoom)){
        ArrayList<Day> weekdays = room.getWeek().getWeekDays();
        Iterator<Day> daysIterator=weekdays.iterator();
        while(daysIterator.hasNext()){
            Day day = daysIterator.next();
            ArrayList<TimeSlot> timeslots = day.getTimeSlot();
            Iterator<TimeSlot> timeslotIterator= timeslots.iterator();
            while(timeslotIterator.hasNext()){
                TimeSlot lecture = timeslotIterator.next();
                if(lecture.getLecture()==null){
                    return false;
                }
            }
        }
        return true;
        }
    }
    return false;
 }
```

Chapter 6

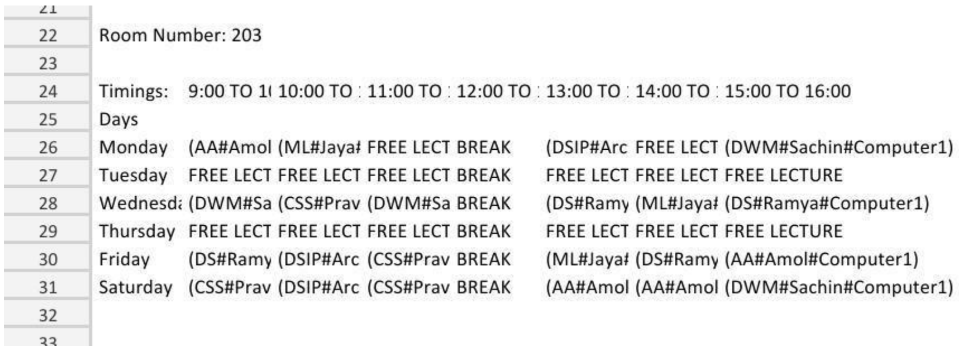
Testing

In our project we had tried different test cases to aviod collision. here are some of the testing results:

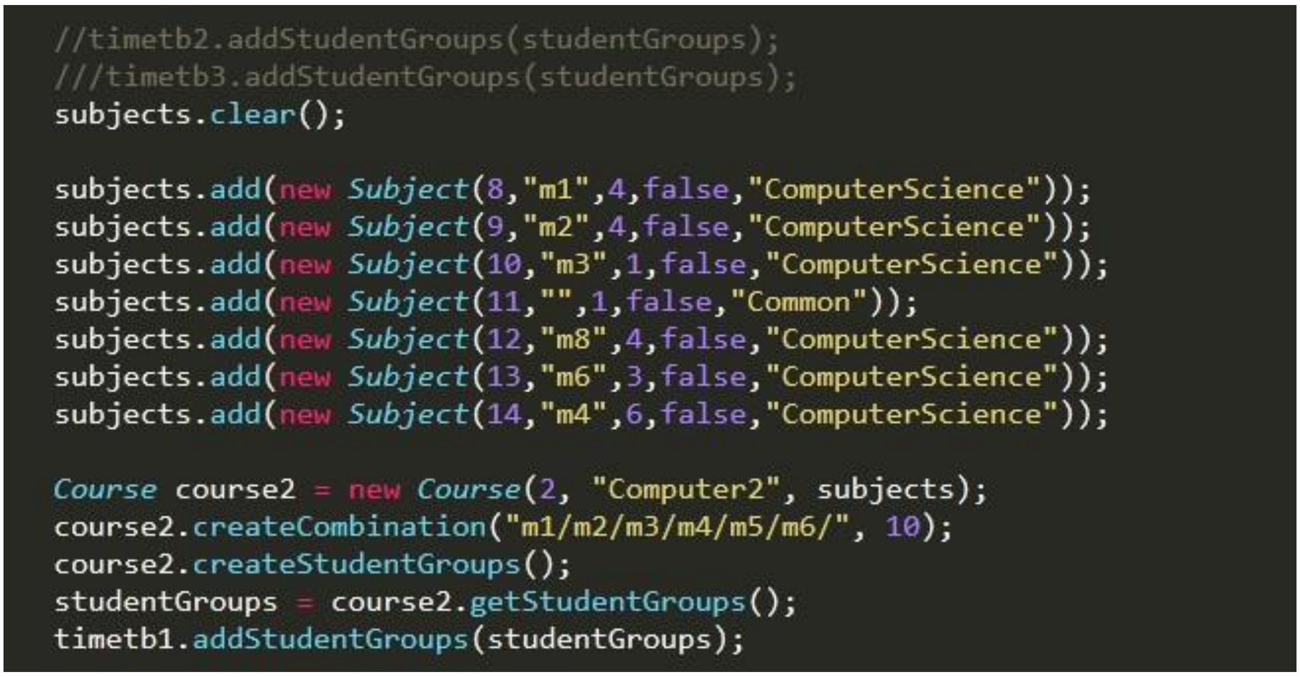


```java
TimeTable timetb1=new TimeTable(classroom, classes);//, professors);
//timetb1.initialization(classroom, classes);
//TimeTable timetb2=new TimeTable(classroom, classes);
//TimeTable timetb3=new TimeTable(classroom, classes);

int courseid = 1;
String courseName="Computer1";
System.out.println("reading input.......");
subjects.add(new Subject(1,"DWM",4,false, "ComputerScience"));
subjects.add(new Subject(2,"CSS",4,false,"ComputerScience"));
subjects.add(new Subject(3,"AA",4,false,"ComputerScience"));
subjects.add(new Subject(4,"",1,false,"Common"));
subjects.add(new Subject(5,"DS",4,false,"ComputerScience"));
subjects.add(new Subject(6,"DSIP",3,false,"ComputerScience"));
subjects.add(new Subject(7,"ML",3,false,"ComputerScience"));


System.out.println("new course creation.......");
Course course1 = new Course(courseid, courseName, subjects);
course1.createCombination("DWM/CSS/AA/PM/DS/DSIP/ML/", 20);
course1.createStudentGroups();
ArrayList<StudentGroups> studentGroups = course1.getStudentGroups();
timetb1.addStudentGroups(studentGroups);
//combinations.addAll(course1.getCombinations());

//timetb2.addStudentGroups(studentGroups);
///timetb3.addStudentGroups(studentGroups);
subjects.clear();
```

For this Input the desired output is:-

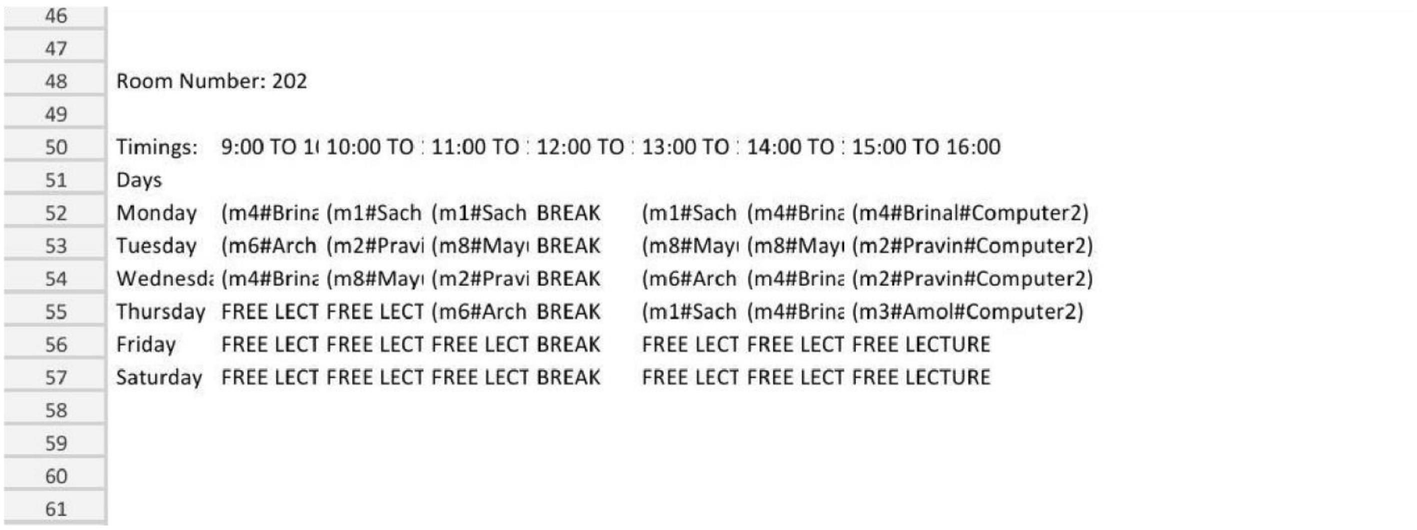


```
Room Number: 203

Timings:   9:00 TO 1( 10:00 TO : 11:00 TO : 12:00 TO : 13:00 TO : 14:00 TO : 15:00 TO 16:00
Days
Monday     (AA#Amol (ML#Jaya# FREE LECT BREAK      (DSIP#Arc FREE LECT (DWM#Sachin#Computer1)
Tuesday    FREE LECT FREE LECT FREE LECT BREAK     FREE LECT FREE LECT FREE LECTURE
Wednesd:   (DWM#Sa (CSS#Prav (DWM#Sa BREAK         (DS#Ramy (ML#Jaya# (DS#Ramya#Computer1)
Thursday   FREE LECT FREE LECT FREE LECT BREAK     FREE LECT FREE LECT FREE LECTURE
Friday     (DS#Ramy (DSIP#Arc (CSS#Prav BREAK      (ML#Jaya# (DS#Ramy (AA#Amol#Computer1)
Saturday   (CSS#Prav (DSIP#Arc (CSS#Prav BREAK     (AA#Amol (AA#Amol (DWM#Sachin#Computer1)
```

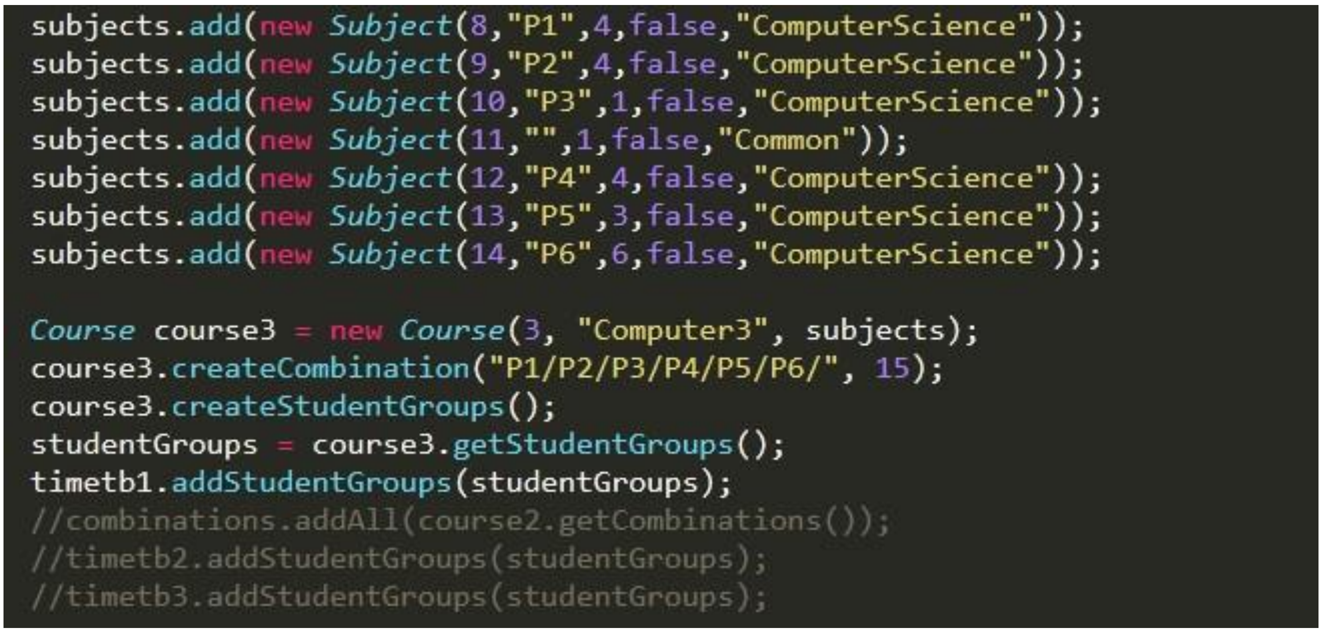


```java
//timetb2.addStudentGroups(studentGroups);
///timetb3.addStudentGroups(studentGroups);
subjects.clear();

subjects.add(new Subject(8,"m1",4,false,"ComputerScience"));
subjects.add(new Subject(9,"m2",4,false,"ComputerScience"));
subjects.add(new Subject(10,"m3",1,false,"ComputerScience"));
subjects.add(new Subject(11,"",1,false,"Common"));
subjects.add(new Subject(12,"m8",4,false,"ComputerScience"));
subjects.add(new Subject(13,"m6",3,false,"ComputerScience"));
subjects.add(new Subject(14,"m4",6,false,"ComputerScience"));

Course course2 = new Course(2, "Computer2", subjects);
course2.createCombination("m1/m2/m3/m4/m5/m6/", 10);
course2.createStudentGroups();
studentGroups = course2.getStudentGroups();
timetb1.addStudentGroups(studentGroups);
```

For this Input The desired Output is:-

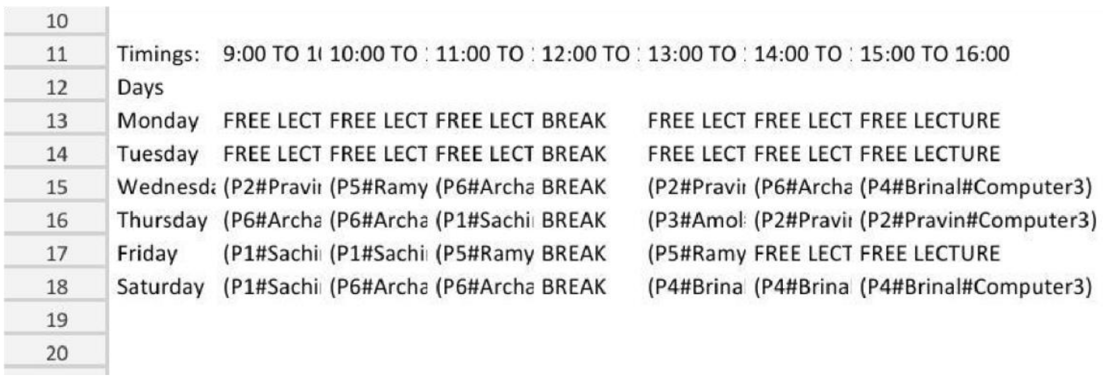


Room Number: 202							
Timings:	9:00 TO 1(	10:00 TO :	11:00 TO :	12:00 TO :	13:00 TO :	14:00 TO :	15:00 TO 16:00
Days							
Monday	(m4#Brina	(m1#Sach	(m1#Sach	BREAK	(m1#Sach	(m4#Brina	(m4#Brinal#Computer2)
Tuesday	(m6#Arch	(m2#Pravi	(m8#Mayı	BREAK	(m8#Mayı	(m8#Mayı	(m2#Pravin#Computer2)
Wednesda	(m4#Brina	(m8#Mayı	(m2#Pravi	BREAK	(m6#Arch	(m4#Brina	(m2#Pravin#Computer2)
Thursday	FREE LECT	FREE LECT	(m6#Arch	BREAK	(m1#Sach	(m4#Brina	(m3#Amol#Computer2)
Friday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Saturday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE



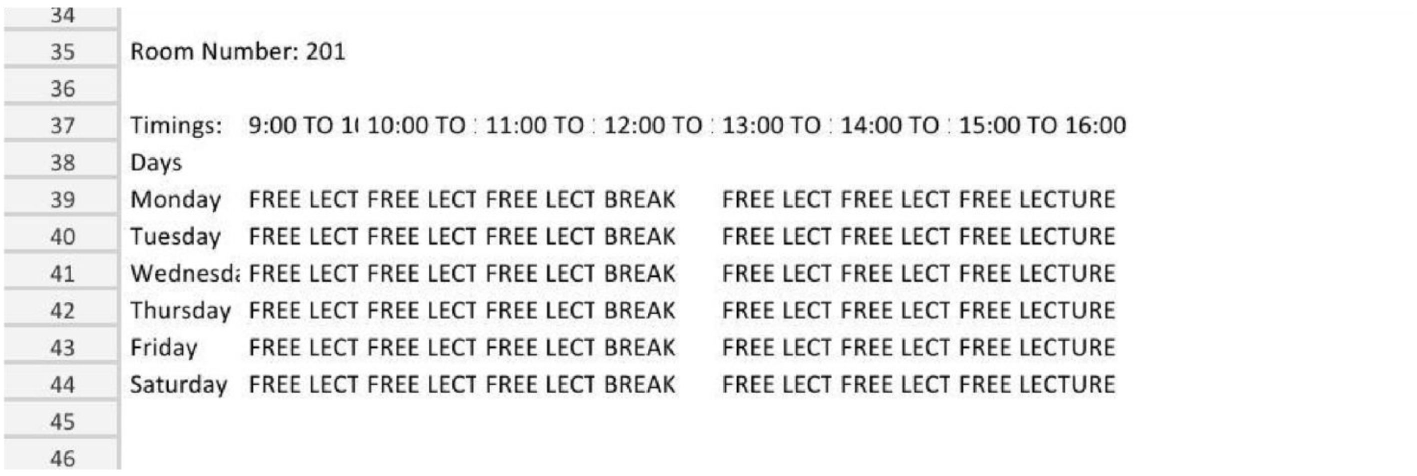
```java
subjects.add(new Subject(8,"P1",4,false,"ComputerScience"));
subjects.add(new Subject(9,"P2",4,false,"ComputerScience"));
subjects.add(new Subject(10,"P3",1,false,"ComputerScience"));
subjects.add(new Subject(11,"",1,false,"Common"));
subjects.add(new Subject(12,"P4",4,false,"ComputerScience"));
subjects.add(new Subject(13,"P5",3,false,"ComputerScience"));
subjects.add(new Subject(14,"P6",6,false,"ComputerScience"));

Course course3 = new Course(3, "Computer3", subjects);
course3.createCombination("P1/P2/P3/P4/P5/P6/", 15);
course3.createStudentGroups();
studentGroups = course3.getStudentGroups();
timetb1.addStudentGroups(studentGroups);
//combinations.addAll(course2.getCombinations());
//timetb2.addStudentGroups(studentGroups);
//timetb3.addStudentGroups(studentGroups);
```

For this input the desired output is:-



Timings:	9:00 TO 1	10:00 TO	11:00 TO	12:00 TO	13:00 TO	14:00 TO	15:00 TO 16:00
Days							
Monday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Tuesday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Wednesda	(P2#Pravi	(P5#Ramy	(P6#Archa	BREAK	(P2#Pravi	(P6#Archa	(P4#Brinal#Computer3)
Thursday	(P6#Archa	(P6#Archa	(P1#Sachi	BREAK	(P3#Amol	(P2#Pravi	(P2#Pravin#Computer3)
Friday	(P1#Sachi	(P1#Sachi	(P5#Ramy	BREAK	(P5#Ramy	FREE LECT	FREE LECTURE
Saturday	(P1#Sachi	(P6#Archa	(P6#Archa	BREAK	(P4#Brina	(P4#Brina	(P4#Brinal#Computer3)

For this class classroom we have not assigned any lectures so this is showing free lectures.



Room Number: 201							
Timings:	9:00 TO 1(	10:00 TO :	11:00 TO :	12:00 TO :	13:00 TO :	14:00 TO :	15:00 TO 16:00
Days							
Monday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Tuesday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Wednesda	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Thursday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Friday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE
Saturday	FREE LECT	FREE LECT	FREE LECT	BREAK	FREE LECT	FREE LECT	FREE LECTURE

Chapter 7

Result

These results immediately induce new directionsof research for finding new approaches on representation, in which instead of looking for a particular solution for a specific problem, GAS are used for searching for a good algorithm that solve the problem.

Chapter 8

Conclusions and Future Scope

A Time-Table management system involves everything that is required for the efficient working of a Institute. Managing timetables,routing,Planning and alloting Load of the Faculties. Creating timetables are one of the biggest headaches for Faculties. It can be confusing and timeconsuming when done manually.Timetable management software is the best solution in such situations TimeTable Generation System generates timetable for each class and teacher,in keeping with the availability calendar of teachers, availability and capacity of physical resources (such as classrooms,laboratories,computer rooom) and rules applicable at different classes,semesters, teachers and subject level. Best of all,this timetable generation system tremendously improves resource utilization and optimization. This will reduce load from the faculties.

Bibliography

[1] Meysam Shahvali Kohshori, Mohammad saniee abadeh,Hedieh Sajedi “A fuzzy genetic algorithm with local search for university course timetabling” 2008 20th IEEE International conference [1].

[2] Abramson D., Abeh, A Parallel genelic algorithm for Solving the school Timetabling Problem., Royal Melbourne Institute of technology, 1991 [2] .

[3] MlLENA KAROVA Drparliiieiit of Cantpuler Scierice Studenrska I Trcliniral Uiiiversily Vama [3].

Appendices

Detailed information, lengthy derivations, raw experimental observations etc. are to be presented in the separate appendices, which shall be numbered in Roman Capitals (e.g. “Appendix I”). Since reference can be drawn to published/unpublished literature in the appendices these should precede the “Literature Cited” section.

Appendix-A: NS2 Download and Installation

1. Download wamp from https://sourceforge.net/projects/wampserver//

2. Go to terminal and do as following commands**sudo apt-get update sudo apt-get install mysql-server**

3. For acess of mysql **$sudo ufw enable $sudo ufw allow mysql**

4. Start Mysql service

$ sudo systemctl start mysql

Acknowledgement

We have great pleasure in presenting the report on **Master Time-Table Generator.** We take this opportunity to express our sincere thanks towards our guide **Dr. Rahul Ambekar** & CoGuide **Co-Guide Name** Department of computer Engineering, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Sachin Malve** Head of Department,Comps, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Amol Kalugade** BE project co-ordinator, Department of Computer Engineering, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Suraj Thakkar
16102057

Neel Dhruva
17202003

Adesh Thosani
17202001

Reference

- Meysam Shahvali Kohshori, Mohammad saniee abadeh,Hedieh Sajedi “A fuzzy genetic algorithm with local search for university course timetabling” 2008 20th IEEE International conference [1].
- Abramson D., Abeh, A Parallel genelic algorithm for Solving the school Timetabling Problem., Royal Melbourne Institute of technology, 1991 [2] .
- MlLENA KAROVA Drparliiieiit of Cantpuler Scierice Studenrska I Trcliniral Uiiiversily Vama [3].