CS6308: JAVA PROGRAMMING

University Timetable Scheduler

Bhagavathi R	**- 2019103009**
Dhivyashri Ramesh	**- 2019103015**

Department of Computer Science and Engineering, College of Engineering Guindy Campus, Anna University, Chennai -25

Table of Contents

Abstract

The objective of this project is to create a university timetable scheduler using the Genetic Algorithm and the Java Framework. Every semester the process of manually creating and accommodating timetables without conflicts is a challenging task and requires a huge amount of time and effort. Our project aims to solve this problem and provide a viable solution by automating this process using the Genetic Algorithms. A Genetic Algorithm is a search heuristic that is inspired by Charles Darwin’s theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

Constraints

To generate the timetable, we have a set of hard and soft constraints, which are as follows.

Hard-constraints (rigid):
• There should not be any single instance of a faculty taking two classes simultaneously
• A class group must not have more than one lectures at the same time
• The minimum number of hours that is required by a course per week should be fulfilled

Soft-constraints (flexible):
• More or less equal load is given to all faculties
• Subjects that have lab may or may not have the same teacher for theory class and lab hours

In conclusion, we hope to achieve a timetable generator that satisfies these constraints and could be put into practical use to reduce manual labour for scheduling timetables every semester.

Tech Stack:

Front-end: HTML,CSS, JavaScript
Back-end: Java , JSP

Introduction

What is Genetic Algorithm?

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. Over successive generations, the population "evolves" toward an optimal solution.
The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

- Selection rules select the individuals, called parents, that contribute to the population at the next generation.
- Crossover rules combine two parents to form children for the next generation.
- Mutation rules apply random changes to individual parents to form children.

Algorithm

Outline of the Algorithm

The following outline summarizes how the genetic algorithm works:

1. The algorithm begins by creating a random initial population.
2. The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:
    1. Scores each member of the current population by computing its fitness value. These values are called the raw fitness scores.
    2. Scales the raw fitness scores to convert them into a more usable range of values. These scaled values are called expectation values.
    3. Selects members, called parents, based on their expectation. The algorithm usually selects individuals that have better fitness values as parents.
    4. Some of the individuals in the current population that have lower fitness are chosen as elite. These elite individuals are passed to the next population.
    5. Produces children from the parents. Children are produced either by making random changes to a single parent—mutation—or by combining the vector entries of a pair of parents—crossover.

6. Replaces the current population with the children to form the next generation.

The genetic algorithm creates three types of children for the next generation:

- Elite are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation.
- Crossover children are created by combining the vectors of a pair of parents.
- Mutation children are created by introducing random changes, or mutations, to a single parent.

Crossover Children

The algorithm creates crossover children by combining pairs of parents in the current population. At each coordinate of the child vector, the default crossover function randomly selects an entry, or gene, at the same coordinate from one of the two parents and assigns it to the child. For problems with linear constraints, the default crossover function creates the child as a random weighted average of the parents.

Mutation Children

The algorithm creates mutation children by randomly changing the genes of individual parents. By default, for unconstrained problems the algorithm adds a random vector from a Gaussian distribution to the parent. For bounded or linearly constrained problems, the child remains feasible.

Genetic Algorithm terminology

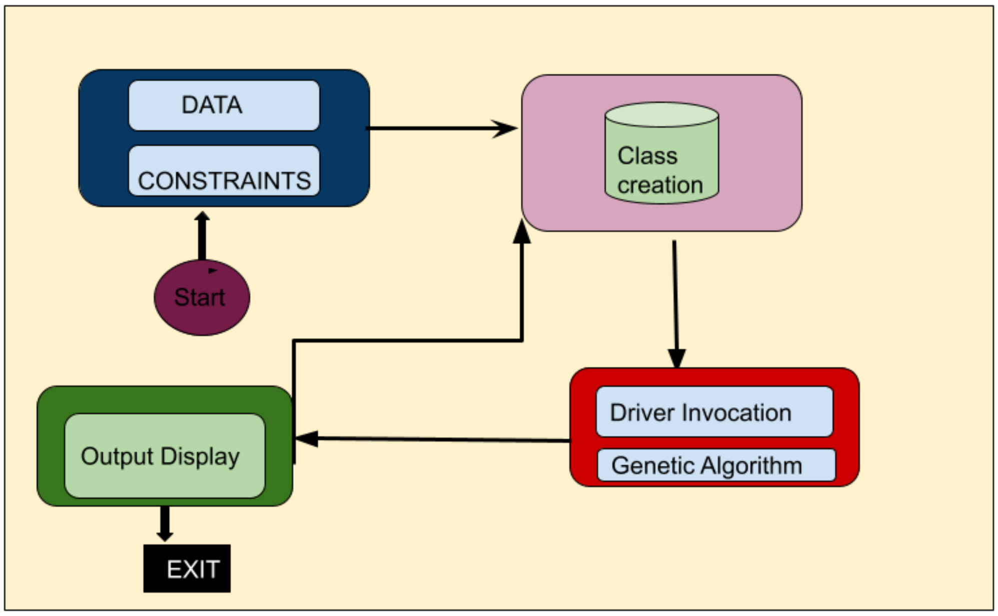
- Fitness Functions

The fitness function is the function you want to optimise. For standard optimization algorithms, this is known as the objective function. The toolbox software tries to find the minimum of the fitness function.

- Fitness Values and Best Fitness Values

The fitness value of an individual is the value of the fitness function for that individual. Because the toolbox software finds the minimum of the fitness function, the best fitness value for a population is the smallest fitness value for any individual in the population.

System Architecture:



Objects of the Scheduler

1. Class
2. Course
3. Department
4. Instructor
5. MeetingTIme
6. Room
7. Data
8. Population
9. Schedule
10. GeneticAlgorithm
11. Driver

Web development Modules

The web development half of this project has been done with the help of JSP.
A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

These implement the servlet concept without explicit declaration of servlets but with internal implementation of servlets. This project mainly consists of implementing the create.jsp and enter.jsp files calling the java objects.

Brief Split-Up of Modules:

1. The Data Module
2. The Genetic Algorithm Module
3. The Data Web Application Module
4. The Genetic Algorithm Web Module

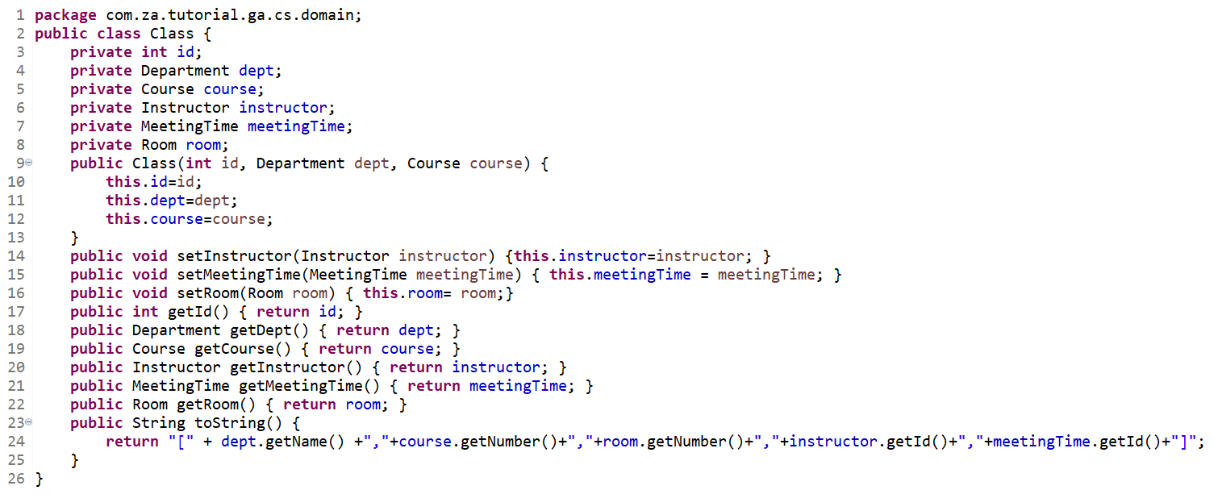
Module 1: Data Module

The data module includes the classes of Class, Course, Instructor, MeetingTime, Room and Department which form the building blocks of the processing data. It's these entities taken together as data that in the further modules will aid in generating the schedule.

Each of the classes have been explained in the below sequence:

Class.java

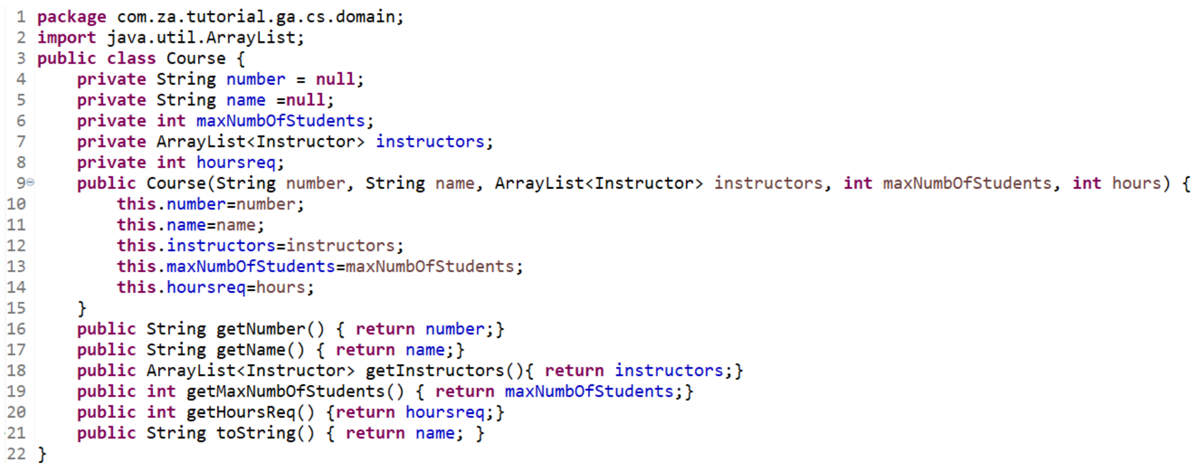
Class.java has the id, department, course, instructor, meeting time, and room of that particular class.



```java
package com.za.tutorial.ga.cs.domain;
public class Class {
    private int id;
    private Department dept;
    private Course course;
    private Instructor instructor;
    private MeetingTime meetingTime;
    private Room room;
    public Class(int id, Department dept, Course course) {
        this.id=id;
        this.dept=dept;
        this.course=course;
    }
    public void setInstructor(Instructor instructor) {this.instructor=instructor; }
    public void setMeetingTime(MeetingTime meetingTime) { this.meetingTime = meetingTime; }
    public void setRoom(Room room) { this.room= room;}
    public int getId() { return id; }
    public Department getDept() { return dept; }
    public Course getCourse() { return course; }
    public Instructor getInstructor() { return instructor; }
    public MeetingTime getMeetingTime() { return meetingTime; }
    public Room getRoom() { return room; }
    public String toString() {
        return "[" + dept.getName() +","+course.getNumber()+","+room.getNumber()+","+instructor.getId()+","+meetingTime.getId()+"]";
    }
}
```

Course.java

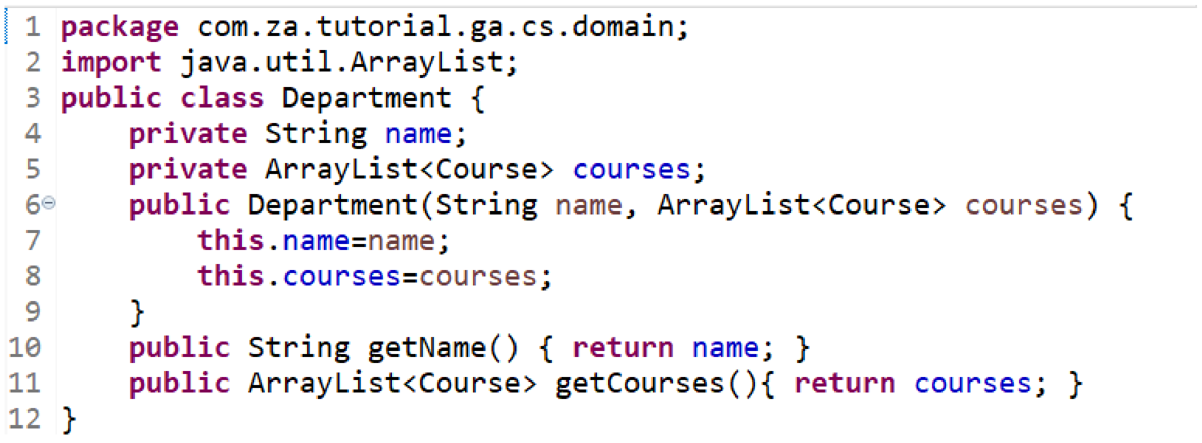
Course.java has the number and name of the course. It also has a list of instructors that are qualified to teach the course. It also has the maximum number of students and the hours required for the course.



```java
package com.za.tutorial.ga.cs.domain;
import java.util.ArrayList;
public class Course {
    private String number = null;
    private String name =null;
    private int maxNumbOfStudents;
    private ArrayList<Instructor> instructors;
    private int hoursreq;
    public Course(String number, String name, ArrayList<Instructor> instructors, int maxNumbOfStudents, int hours) {
        this.number=number;
        this.name=name;
        this.instructors=instructors;
        this.maxNumbOfStudents=maxNumbOfStudents;
        this.hoursreq=hours;
    }
    public String getNumber() { return number;}
    public String getName() { return name;}
    public ArrayList<Instructor> getInstructors(){ return instructors;}
    public int getMaxNumbOfStudents() { return maxNumbOfStudents;}
    public int getHoursReq() {return hoursreq;}
    public String toString() { return name; }
}
```

Department.java

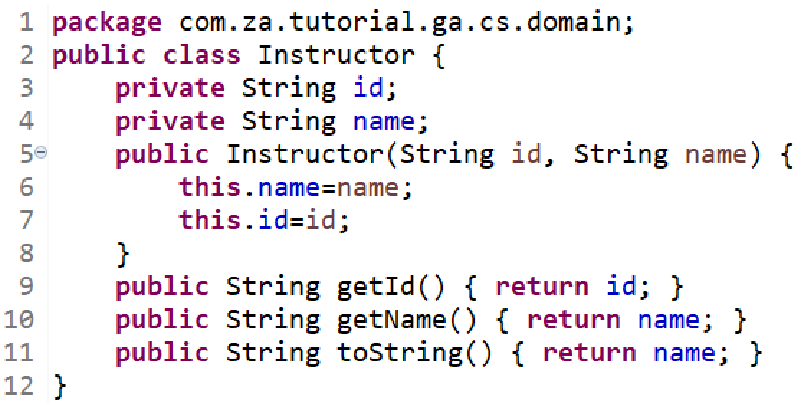
Department class has the name of the department and an array list of all the courses that come under this department.



```java
package com.za.tutorial.ga.cs.domain;
import java.util.ArrayList;
public class Department {
    private String name;
    private ArrayList<Course> courses;
    public Department(String name, ArrayList<Course> courses) {
        this.name=name;
        this.courses=courses;
    }
    public String getName() { return name; }
    public ArrayList<Course> getCourses(){ return courses; }
}
```

Instructor.java

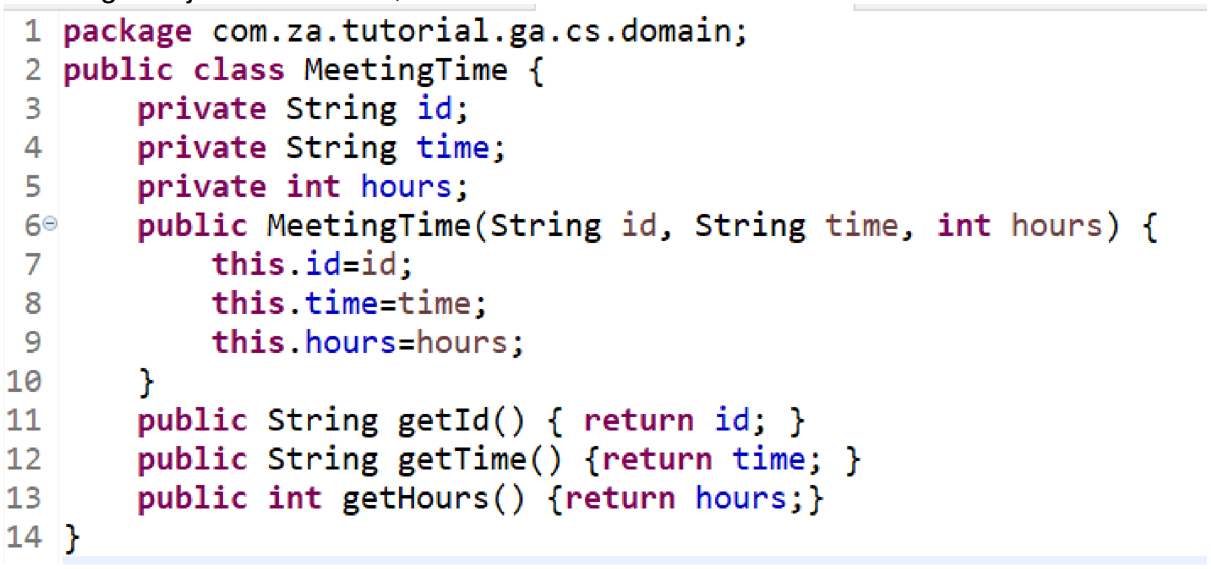
Instructor.java has the id, name of the instructor.



```java
package com.za.tutorial.ga.cs.domain;
public class Instructor {
    private String id;
    private String name;
    public Instructor(String id, String name) {
        this.name=name;
        this.id=id;
    }
    public String getId() { return id; }
    public String getName() { return name; }
    public String toString() { return name; }
}
```

MeetingTIme.java

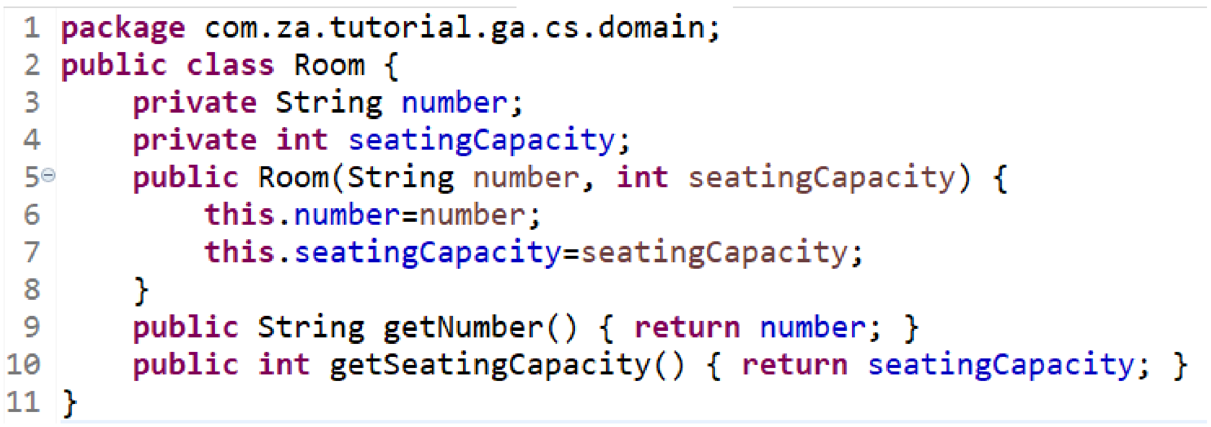
Meetingtime.java has the id, time and the time duration of that meet.



```java
package com.za.tutorial.ga.cs.domain;
public class MeetingTime {
    private String id;
    private String time;
    private int hours;
    public MeetingTime(String id, String time, int hours) {
        this.id=id;
        this.time=time;
        this.hours=hours;
    }
    public String getId() { return id; }
    public String getTime() {return time; }
    public int getHours() {return hours;}
}
```

Room.java

Room.java has the room number and the seating capacity of that room.



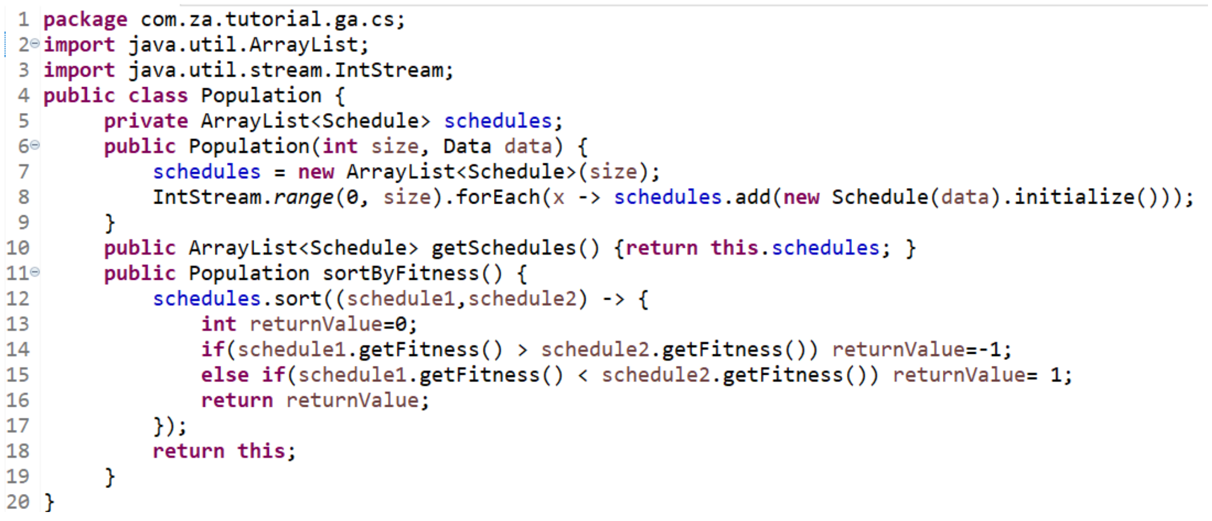
```java
package com.za.tutorial.ga.cs.domain;
public class Room {
    private String number;
    private int seatingCapacity;
    public Room(String number, int seatingCapacity) {
        this.number=number;
        this.seatingCapacity=seatingCapacity;
    }
    public String getNumber() { return number; }
    public int getSeatingCapacity() { return seatingCapacity; }
}
```

Module 2: The Genetic Algorithm Module

This module includes declaration of classes that perform the schedule generation. The algorithm here requires creation of population using the data already generated and running of schedules till the fitness mark has been reached. For this purpose, we need Population, Data, Driver, Schedule and Genetic Algorithm itself.
Each of these classes have been explained with their functionalities.

Population.java

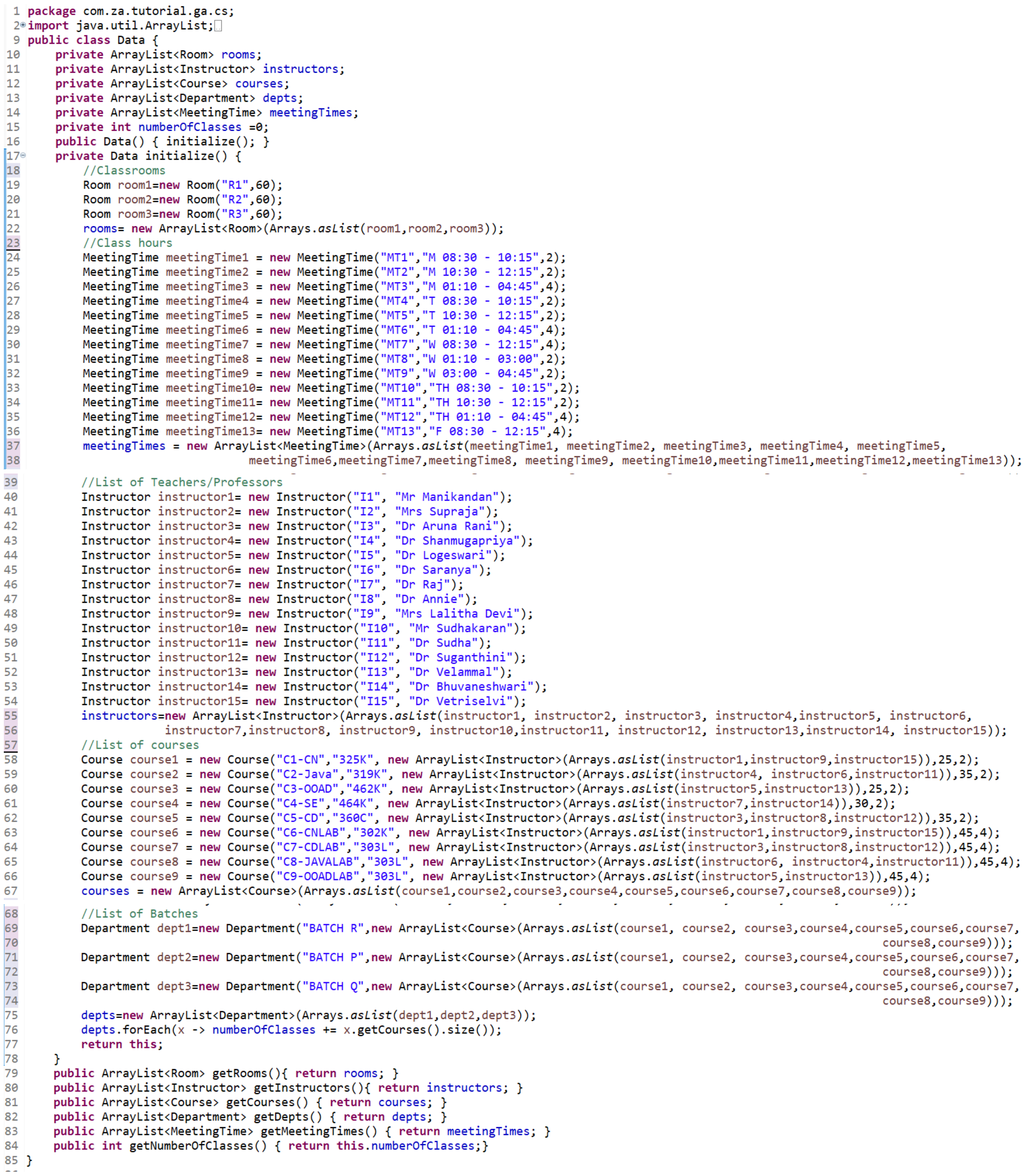
The Population class has an array list of Schedules. This class calls the initialize() function in the Schedule class to use the data from the Data class to make an initial Schedule. It takes the Schedules created and sorts them by fitness and returns the sorted Population.



```java
package com.za.tutorial.ga.cs;
import java.util.ArrayList;
import java.util.stream.IntStream;
public class Population {
     private ArrayList<Schedule> schedules;
     public Population(int size, Data data) {
         schedules = new ArrayList<Schedule>(size);
         IntStream.range(0, size).forEach(x -> schedules.add(new Schedule(data).initialize()));
     }
     public ArrayList<Schedule> getSchedules() {return this.schedules; }
     public Population sortByFitness() {
         schedules.sort((schedule1,schedule2) -> {
             int returnValue=0;
             if(schedule1.getFitness() > schedule2.getFitness()) returnValue=-1;
             else if(schedule1.getFitness() < schedule2.getFitness()) returnValue= 1;
             return returnValue;
         });
         return this;
     }
}
```

Data.java

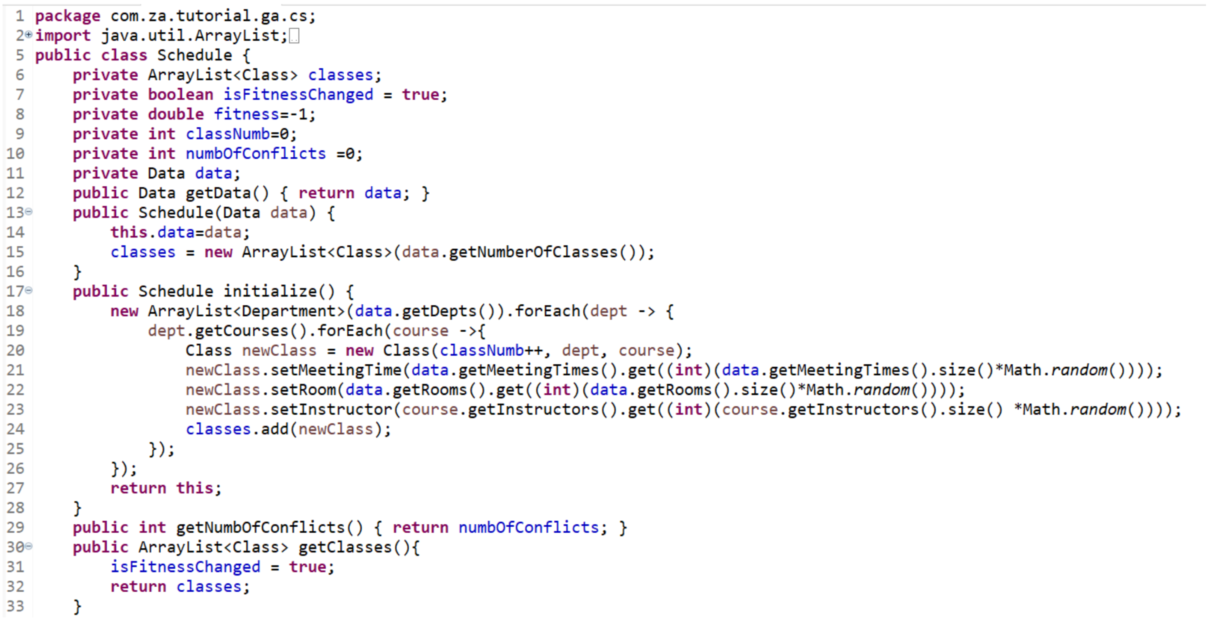
The data class has array lists for rooms, instructors, courses, departments, and meetingTimes. The initialize() function collects all the information required to make the schedule and returns the object.



```java
package com.za.tutorial.ga.cs;
import java.util.ArrayList;
public class Data {
    private ArrayList<Room> rooms;
    private ArrayList<Instructor> instructors;
    private ArrayList<Course> courses;
    private ArrayList<Department> depts;
    private ArrayList<MeetingTime> meetingTimes;
    private int numberOfClasses =0;
    public Data() { initialize(); }
    private Data initialize() {
        //Classrooms
        Room room1=new Room("R1",60);
        Room room2=new Room("R2",60);
        Room room3=new Room("R3",60);
        rooms= new ArrayList<Room>(Arrays.asList(room1,room2,room3));
        //Class hours
        MeetingTime meetingTime1 = new MeetingTime("MT1","M 08:30 - 10:15",2);
        MeetingTime meetingTime2 = new MeetingTime("MT2","M 10:30 - 12:15",2);
        MeetingTime meetingTime3 = new MeetingTime("MT3","M 01:10 - 04:45",4);
        MeetingTime meetingTime4 = new MeetingTime("MT4","T 08:30 - 10:15",2);
        MeetingTime meetingTime5 = new MeetingTime("MT5","T 10:30 - 12:15",2);
        MeetingTime meetingTime6 = new MeetingTime("MT6","T 01:10 - 04:45",4);
        MeetingTime meetingTime7 = new MeetingTime("MT7","W 08:30 - 12:15",4);
        MeetingTime meetingTime8 = new MeetingTime("MT8","W 01:10 - 03:00",2);
        MeetingTime meetingTime9 = new MeetingTime("MT9","W 03:00 - 04:45",2);
        MeetingTime meetingTime10= new MeetingTime("MT10","TH 08:30 - 10:15",2);
        MeetingTime meetingTime11= new MeetingTime("MT11","TH 10:30 - 12:15",2);
        MeetingTime meetingTime12= new MeetingTime("MT12","TH 01:10 - 04:45",4);
        MeetingTime meetingTime13= new MeetingTime("MT13","F 08:30 - 12:15",4);
        meetingTimes = new ArrayList<MeetingTime>(Arrays.asList(meetingTime1, meetingTime2, meetingTime3, meetingTime4, meetingTime5,
                        meetingTime6,meetingTime7,meetingTime8, meetingTime9, meetingTime10,meetingTime11,meetingTime12,meetingTime13));
        //List of Teachers/Professors
        Instructor instructor1= new Instructor("I1", "Mr Manikandan");
        Instructor instructor2= new Instructor("I2", "Mrs Supraja");
        Instructor instructor3= new Instructor("I3", "Dr Aruna Rani");
        Instructor instructor4= new Instructor("I4", "Dr Shanmugapriya");
        Instructor instructor5= new Instructor("I5", "Dr Logeswari");
        Instructor instructor6= new Instructor("I6", "Dr Saranya");
        Instructor instructor7= new Instructor("I7", "Dr Raj");
        Instructor instructor8= new Instructor("I8", "Dr Annie");
        Instructor instructor9= new Instructor("I9", "Mrs Lalitha Devi");
        Instructor instructor10= new Instructor("I10", "Mr Sudhakaran");
        Instructor instructor11= new Instructor("I11", "Dr Sudha");
        Instructor instructor12= new Instructor("I12", "Dr Suganthini");
        Instructor instructor13= new Instructor("I13", "Dr Velammal");
        Instructor instructor14= new Instructor("I14", "Dr Bhuvaneshwari");
        Instructor instructor15= new Instructor("I15", "Dr Vetriselvi");
        instructors=new ArrayList<Instructor>(Arrays.asList(instructor1, instructor2, instructor3, instructor4,instructor5, instructor6,
                instructor7,instructor8, instructor9, instructor10,instructor11, instructor12, instructor13,instructor14, instructor15));
        //List of courses
        Course course1 = new Course("C1-CN","325K", new ArrayList<Instructor>(Arrays.asList(instructor1,instructor9,instructor15)),25,2);
        Course course2 = new Course("C2-Java","319K", new ArrayList<Instructor>(Arrays.asList(instructor4, instructor6,instructor11)),35,2);
        Course course3 = new Course("C3-OOAD","462K", new ArrayList<Instructor>(Arrays.asList(instructor5,instructor13)),25,2);
        Course course4 = new Course("C4-SE","464K", new ArrayList<Instructor>(Arrays.asList(instructor7,instructor14)),30,2);
        Course course5 = new Course("C5-CD","360C", new ArrayList<Instructor>(Arrays.asList(instructor3,instructor8,instructor12)),35,2);
        Course course6 = new Course("C6-CNLAB","302K", new ArrayList<Instructor>(Arrays.asList(instructor1,instructor9,instructor15)),45,4);
        Course course7 = new Course("C7-CDLAB","303L", new ArrayList<Instructor>(Arrays.asList(instructor3,instructor8,instructor12)),45,4);
        Course course8 = new Course("C8-JAVALAB","303L", new ArrayList<Instructor>(Arrays.asList(instructor6, instructor4,instructor11)),45,4);
        Course course9 = new Course("C9-OOADLAB","303L", new ArrayList<Instructor>(Arrays.asList(instructor5,instructor13)),45,4);
        courses = new ArrayList<Course>(Arrays.asList(course1,course2,course3,course4,course5,course6,course7,course8,course9));
        //List of Batches
        Department dept1=new Department("BATCH R",new ArrayList<Course>(Arrays.asList(course1, course2, course3,course4,course5,course6,course7,
                                                                                         course8,course9)));
        Department dept2=new Department("BATCH P",new ArrayList<Course>(Arrays.asList(course1, course2, course3,course4,course5,course6,course7,
                                                                                         course8,course9)));
        Department dept3=new Department("BATCH Q",new ArrayList<Course>(Arrays.asList(course1, course2, course3,course4,course5,course6,course7,
                                                                                         course8,course9)));
        depts=new ArrayList<Department>(Arrays.asList(dept1,dept2,dept3));
        depts.forEach(x -> numberOfClasses += x.getCourses().size());
        return this;
    }
    public ArrayList<Room> getRooms(){ return rooms; }
    public ArrayList<Instructor> getInstructors(){ return instructors; }
    public ArrayList<Course> getCourses() { return courses; }
    public ArrayList<Department> getDepts() { return depts; }
    public ArrayList<MeetingTime> getMeetingTimes() { return meetingTimes; }
    public int getNumberOfClasses() { return this.numberOfClasses;}
}
```

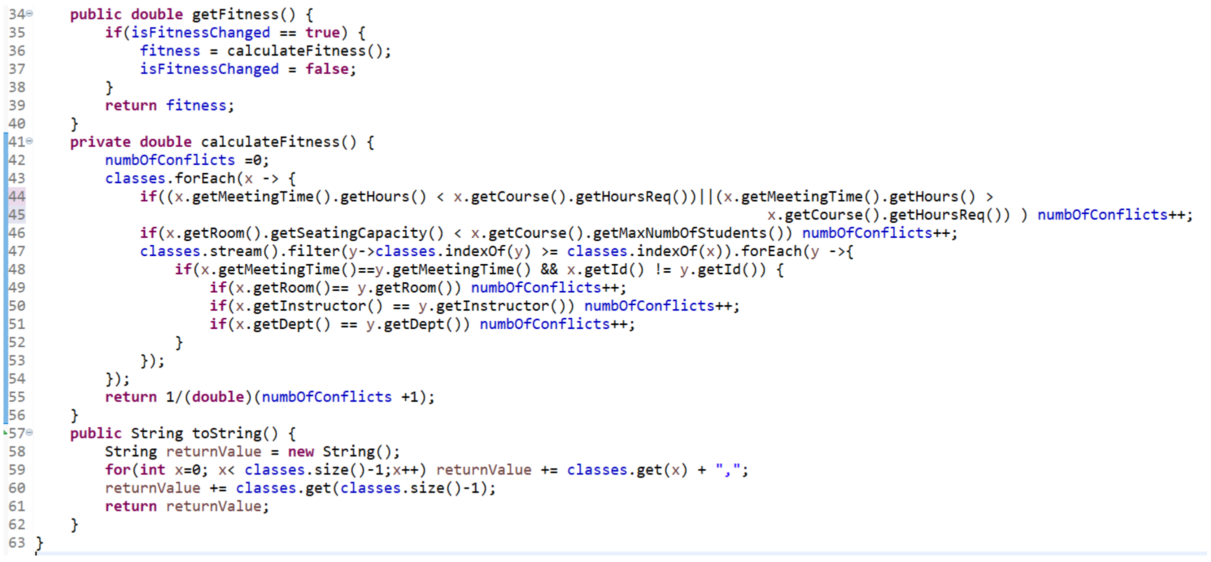
Schedule.java

This class initializes the first schedule by randomly picking meeting times, rooms, and professors for each course.



```java
package com.za.tutorial.ga.cs;
import java.util.ArrayList;
public class Schedule {
    private ArrayList<Class> classes;
    private boolean isFitnessChanged = true;
    private double fitness=-1;
    private int classNumb=0;
    private int numbOfConflicts =0;
    private Data data;
    public Data getData() { return data; }
    public Schedule(Data data) {
        this.data=data;
        classes = new ArrayList<Class>(data.getNumberOfClasses());
    }
    public Schedule initialize() {
        new ArrayList<Department>(data.getDepts()).forEach(dept -> {
            dept.getCourses().forEach(course ->{
                Class newClass = new Class(classNumb++, dept, course);
                newClass.setMeetingTime(data.getMeetingTimes().get((int)(data.getMeetingTimes().size()*Math.random())));
                newClass.setRoom(data.getRooms().get((int)(data.getRooms().size()*Math.random())));
                newClass.setInstructor(course.getInstructors().get((int)(course.getInstructors().size() *Math.random())));
                classes.add(newClass);
            });
        });
        return this;
    }
    public int getNumbOfConflicts() { return numbOfConflicts; }
    public ArrayList<Class> getClasses(){
        isFitnessChanged = true;
        return classes;
    }
```

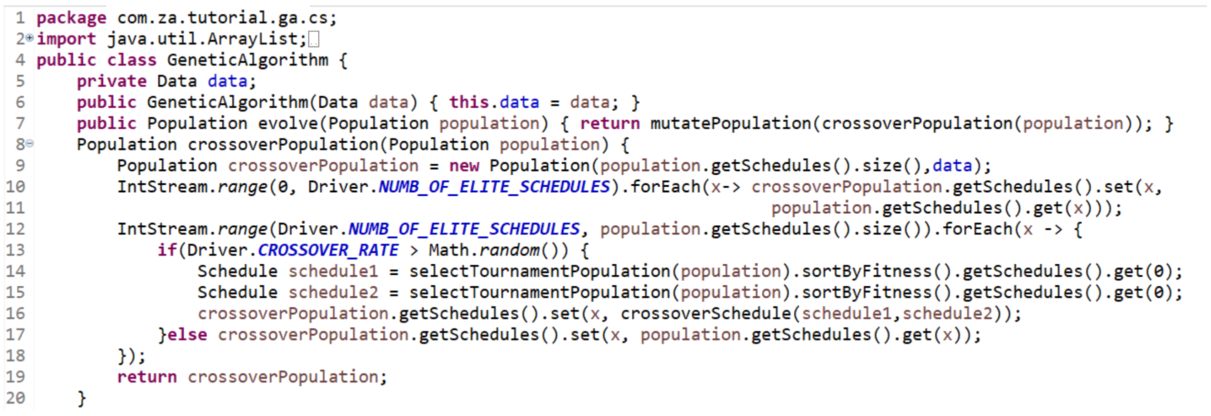
The calculateFitness method counts the number of conflicts that have arised in that particular schedule and returns 1/numbofconflicts as the fitness.



```java
    public double getFitness() {
        if(isFitnessChanged == true) {
            fitness = calculateFitness();
            isFitnessChanged = false;
        }
        return fitness;
    }
    private double calculateFitness() {
        numbOfConflicts =0;
        classes.forEach(x -> {
            if((x.getMeetingTime().getHours() < x.getCourse().getHoursReq())||(x.getMeetingTime().getHours() >
                                                                   x.getCourse().getHoursReq()) ) numbOfConflicts++;
            if(x.getRoom().getSeatingCapacity() < x.getCourse().getMaxNumbOfStudents()) numbOfConflicts++;
            classes.stream().filter(y->classes.indexOf(y) >= classes.indexOf(x)).forEach(y ->{
                if(x.getMeetingTime()==y.getMeetingTime() && x.getId() != y.getId()) {
                    if(x.getRoom()== y.getRoom()) numbOfConflicts++;
                    if(x.getInstructor() == y.getInstructor()) numbOfConflicts++;
                    if(x.getDept() == y.getDept()) numbOfConflicts++;
                }
            });
        });
        return 1/(double)(numbOfConflicts +1);
    }
    public String toString() {
        String returnValue = new String();
        for(int x=0; x< classes.size()-1;x++) returnValue += classes.get(x) + ",";
        returnValue += classes.get(classes.size()-1);
        return returnValue;
    }
}
```

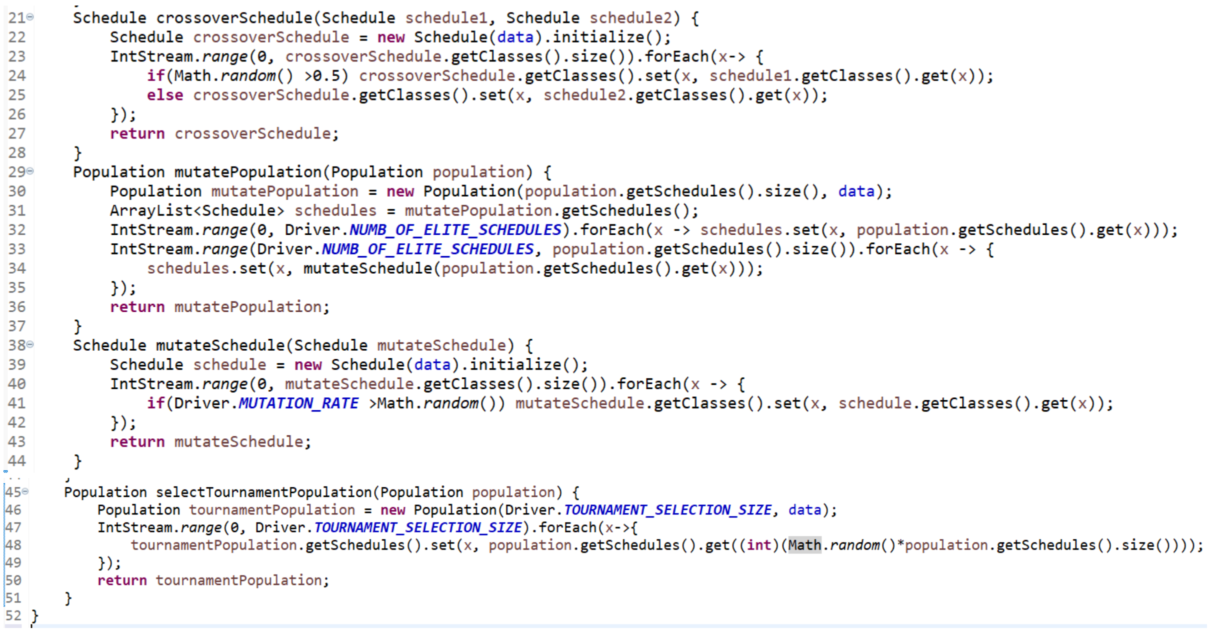
GeneticAlgorithm.java

This class runs the genetic algorithm. It has the crossover, and mutate methods. The crossoverPopulation method selects two Schedules to perform crossover.



```java
package com.za.tutorial.ga.cs;
import java.util.ArrayList;
public class GeneticAlgorithm {
    private Data data;
    public GeneticAlgorithm(Data data) { this.data = data; }
    public Population evolve(Population population) { return mutatePopulation(crossoverPopulation(population)); }
    Population crossoverPopulation(Population population) {
        Population crossoverPopulation = new Population(population.getSchedules().size(),data);
        IntStream.range(0, Driver.NUMB_OF_ELITE_SCHEDULES).forEach(x-> crossoverPopulation.getSchedules().set(x,
                                                                  population.getSchedules().get(x)));
        IntStream.range(Driver.NUMB_OF_ELITE_SCHEDULES, population.getSchedules().size()).forEach(x -> {
            if(Driver.CROSSOVER_RATE > Math.random()) {
                Schedule schedule1 = selectTournamentPopulation(population).sortByFitness().getSchedules().get(0);
                Schedule schedule2 = selectTournamentPopulation(population).sortByFitness().getSchedules().get(0);
                crossoverPopulation.getSchedules().set(x, crossoverSchedule(schedule1,schedule2));
            }else crossoverPopulation.getSchedules().set(x, population.getSchedules().get(x));
        });
        return crossoverPopulation;
    }
```

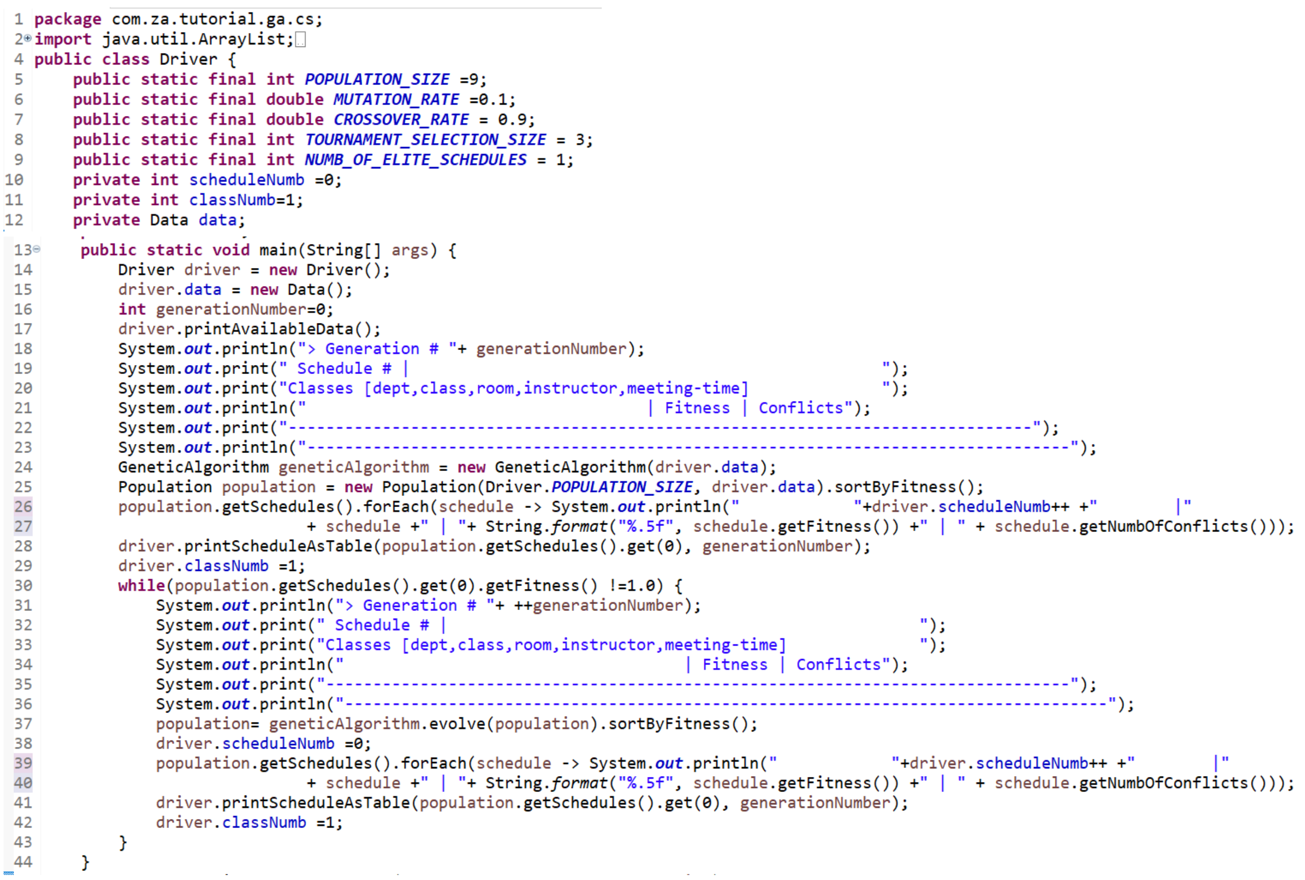
Once the schedules are selected, the crossoverSchedule method performs crossover and returns the crossoverSchedule. The mutatePopulation method selects a schedule to perform mutation on and passes it as a parameter in the mutateSchedule method. The mutateSchedule method performs mutation and returns the mutated schedule.



```java
    Schedule crossoverSchedule(Schedule schedule1, Schedule schedule2) {
        Schedule crossoverSchedule = new Schedule(data).initialize();
        IntStream.range(0, crossoverSchedule.getClasses().size()).forEach(x-> {
            if(Math.random() >0.5) crossoverSchedule.getClasses().set(x, schedule1.getClasses().get(x));
            else crossoverSchedule.getClasses().set(x, schedule2.getClasses().get(x));
        });
        return crossoverSchedule;
    }
    Population mutatePopulation(Population population) {
        Population mutatePopulation = new Population(population.getSchedules().size(), data);
        ArrayList<Schedule> schedules = mutatePopulation.getSchedules();
        IntStream.range(0, Driver.NUMB_OF_ELITE_SCHEDULES).forEach(x -> schedules.set(x, population.getSchedules().get(x)));
        IntStream.range(Driver.NUMB_OF_ELITE_SCHEDULES, population.getSchedules().size()).forEach(x -> {
            schedules.set(x, mutateSchedule(population.getSchedules().get(x)));
        });
        return mutatePopulation;
    }
    Schedule mutateSchedule(Schedule mutateSchedule) {
        Schedule schedule = new Schedule(data).initialize();
        IntStream.range(0, mutateSchedule.getClasses().size()).forEach(x -> {
            if(Driver.MUTATION_RATE >Math.random()) mutateSchedule.getClasses().set(x, schedule.getClasses().get(x));
        });
        return mutateSchedule;
    }
    Population selectTournamentPopulation(Population population) {
        Population tournamentPopulation = new Population(Driver.TOURNAMENT_SELECTION_SIZE, data);
        IntStream.range(0, Driver.TOURNAMENT_SELECTION_SIZE).forEach(x->{
            tournamentPopulation.getSchedules().set(x, population.getSchedules().get((int)(Math.random()*population.getSchedules().size())));
        });
        return tournamentPopulation;
    }
}
```

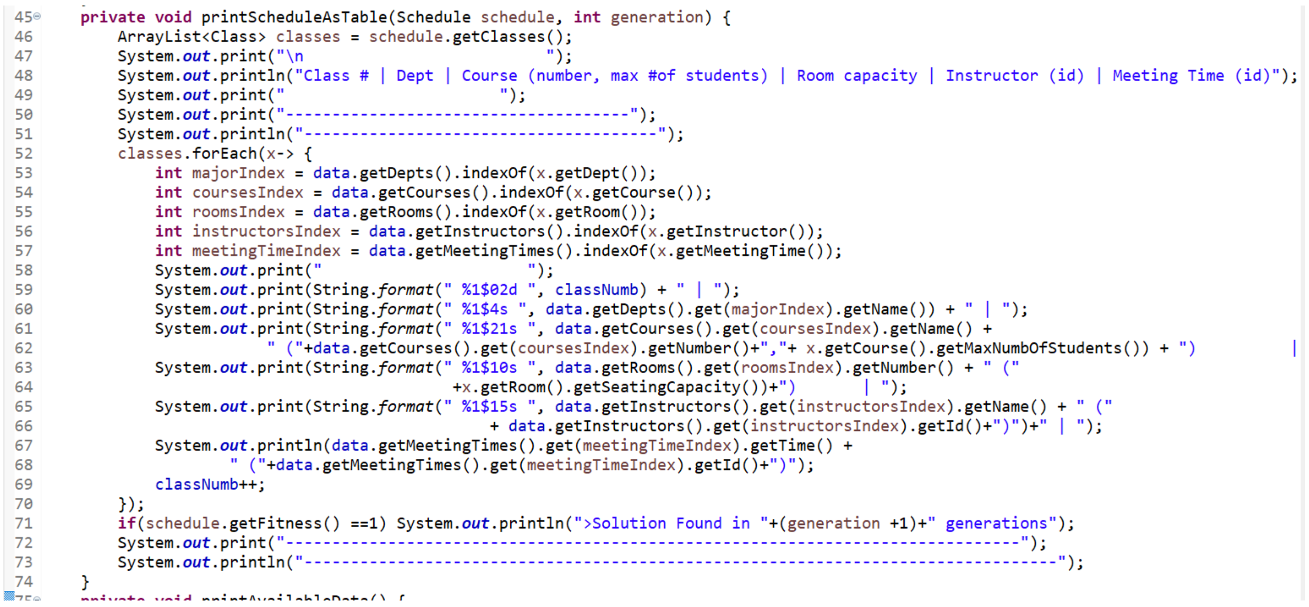
Driver.java

The Driver class is the main class of the algorithm. This class initializes the data required for scheduling and then calls the Genetic Algorithm class. A new population object is created using the Constructor with parameters and the sortByFitness() method is called. It prints out the initial Schedule. The While loop runs until there is a Schedule with fitness of 1.

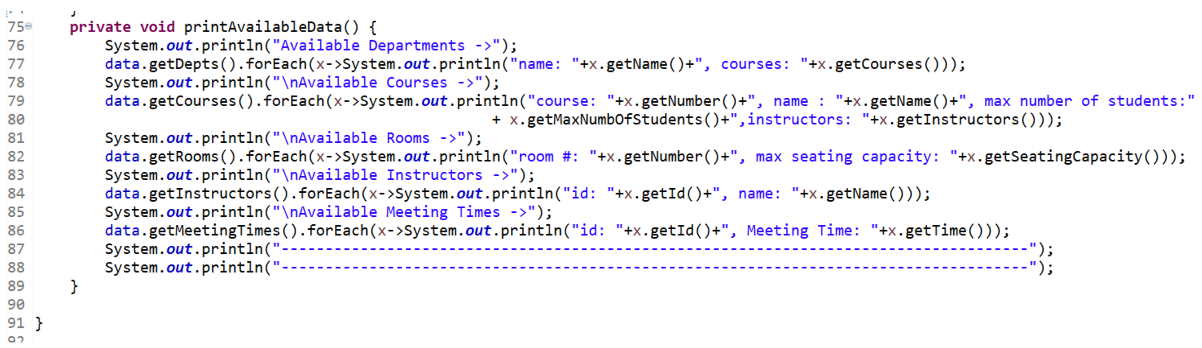


```java
package com.za.tutorial.ga.cs;
import java.util.ArrayList;
public class Driver {
    public static final int POPULATION_SIZE =9;
    public static final double MUTATION_RATE =0.1;
    public static final double CROSSOVER_RATE = 0.9;
    public static final int TOURNAMENT_SELECTION_SIZE = 3;
    public static final int NUMB_OF_ELITE_SCHEDULES = 1;
    private int scheduleNumb =0;
    private int classNumb=1;
    private Data data;
    public static void main(String[] args) {
        Driver driver = new Driver();
        driver.data = new Data();
        int generationNumber=0;
        driver.printAvailableData();
        System.out.println("> Generation # "+ generationNumber);
        System.out.print(" Schedule # |                                                  ");
        System.out.print("Classes [dept,class,room,instructor,meeting-time]              ");
        System.out.println("                                     | Fitness | Conflicts");
        System.out.print("-------------------------------------------------------------------------------");
        System.out.println("---------------------------------------------------------------------------------");
        GeneticAlgorithm geneticAlgorithm = new GeneticAlgorithm(driver.data);
        Population population = new Population(Driver.POPULATION_SIZE, driver.data).sortByFitness();
        population.getSchedules().forEach(schedule -> System.out.println("           "+driver.scheduleNumb++ +"        |"
                                + schedule +" | "+ String.format("%.5f", schedule.getFitness()) +" | " + schedule.getNumbOfConflicts()));
        driver.printScheduleAsTable(population.getSchedules().get(0), generationNumber);
        driver.classNumb =1;
        while(population.getSchedules().get(0).getFitness() !=1.0) {
            System.out.println("> Generation # "+ ++generationNumber);
            System.out.print(" Schedule # |                                                  ");
            System.out.print("Classes [dept,class,room,instructor,meeting-time]              ");
            System.out.println("                                    | Fitness | Conflicts");
            System.out.print("-------------------------------------------------------------------------------");
            System.out.println("---------------------------------------------------------------------------------");
            population= geneticAlgorithm.evolve(population).sortByFitness();
            driver.scheduleNumb =0;
            population.getSchedules().forEach(schedule -> System.out.println("           "+driver.scheduleNumb++ +"        |"
                                + schedule +" | "+ String.format("%.5f", schedule.getFitness()) +" | " + schedule.getNumbOfConflicts()));
            driver.printScheduleAsTable(population.getSchedules().get(0), generationNumber);
            driver.classNumb =1;
        }
    }
```

This method is called to print out the fittest Schedule of a generation as a table:



```java
    private void printScheduleAsTable(Schedule schedule, int generation) {
        ArrayList<Class> classes = schedule.getClasses();
        System.out.print("\n                        ");
        System.out.println("Class # | Dept | Course (number, max #of students) | Room capacity | Instructor (id) | Meeting Time (id)");
        System.out.print("                      ");
        System.out.print("------------------------------------");
        System.out.println("-------------------------------------");
        classes.forEach(x-> {
            int majorIndex = data.getDepts().indexOf(x.getDept());
            int coursesIndex = data.getCourses().indexOf(x.getCourse());
            int roomsIndex = data.getRooms().indexOf(x.getRoom());
            int instructorsIndex = data.getInstructors().indexOf(x.getInstructor());
            int meetingTimeIndex = data.getMeetingTimes().indexOf(x.getMeetingTime());
            System.out.print("                        ");
            System.out.print(String.format(" %1$02d ", classNumb) + " | ");
            System.out.print(String.format(" %1$4s ", data.getDepts().get(majorIndex).getName()) + " | ");
            System.out.print(String.format(" %1$21s ", data.getCourses().get(coursesIndex).getName() +
                    " ("+data.getCourses().get(coursesIndex).getNumber()+","+ x.getCourse().getMaxNumbOfStudents()) + ")          |
            System.out.print(String.format(" %1$10s ", data.getRooms().get(roomsIndex).getNumber() + " ("
                                        +x.getRoom().getSeatingCapacity())+")       | ");
            System.out.print(String.format(" %1$15s ", data.getInstructors().get(instructorsIndex).getName() + " ("
                                            + data.getInstructors().get(instructorsIndex).getId()+")")+" | ");
            System.out.println(data.getMeetingTimes().get(meetingTimeIndex).getTime() +
                " ("+data.getMeetingTimes().get(meetingTimeIndex).getId()+")");
            classNumb++;
        });
        if(schedule.getFitness() ==1) System.out.println(">Solution Found in "+(generation +1)+" generations");
        System.out.print("------------------------------------------------------------------------------");
        System.out.println("--------------------------------------------------------------------------------");
    }
```

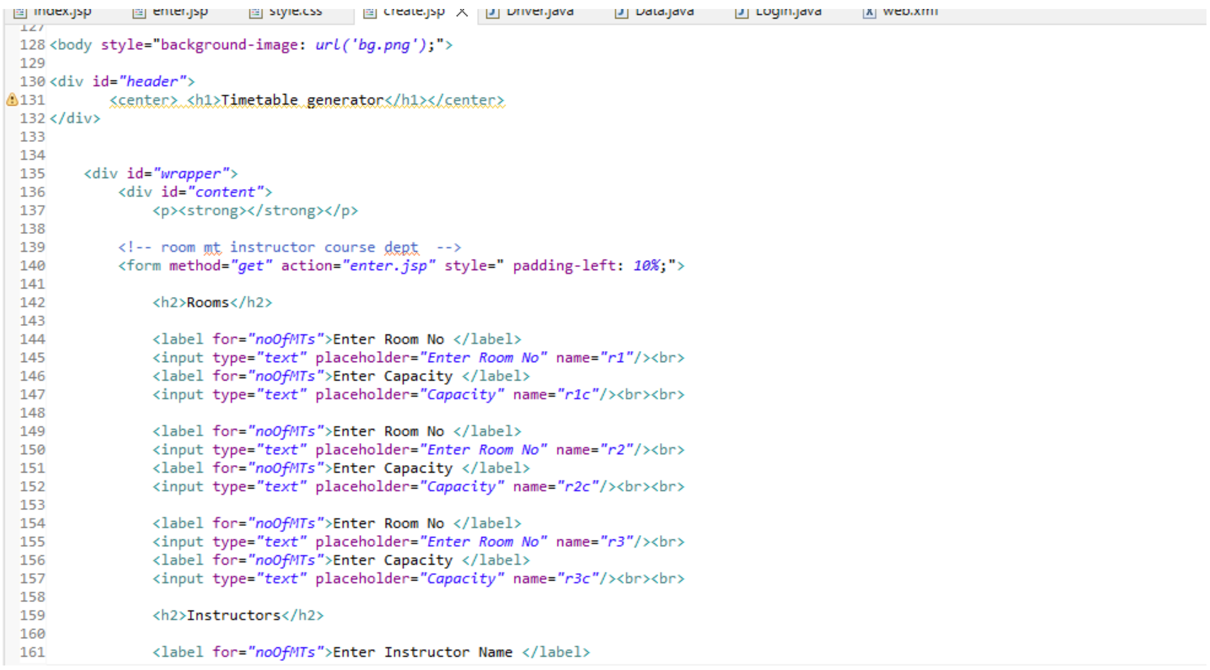


```java
    private void printAvailableData() {
        System.out.println("Available Departments ->");
        data.getDepts().forEach(x->System.out.println("name: "+x.getName()+", courses: "+x.getCourses()));
        System.out.println("\nAvailable Courses ->");
        data.getCourses().forEach(x->System.out.println("course: "+x.getNumber()+", name : "+x.getName()+", max number of students:"
                                            + x.getMaxNumbOfStudents()+",instructors: "+x.getInstructors()));
        System.out.println("\nAvailable Rooms ->");
        data.getRooms().forEach(x->System.out.println("room #: "+x.getNumber()+", max seating capacity: "+x.getSeatingCapacity()));
        System.out.println("\nAvailable Instructors ->");
        data.getInstructors().forEach(x->System.out.println("id: "+x.getId()+", name: "+x.getName()));
        System.out.println("\nAvailable Meeting Times ->");
        data.getMeetingTimes().forEach(x->System.out.println("id: "+x.getId()+", Meeting Time: "+x.getTime()));
        System.out.println("-------------------------------------------------------------------------------------");
        System.out.println("-------------------------------------------------------------------------------------");
    }

}
```

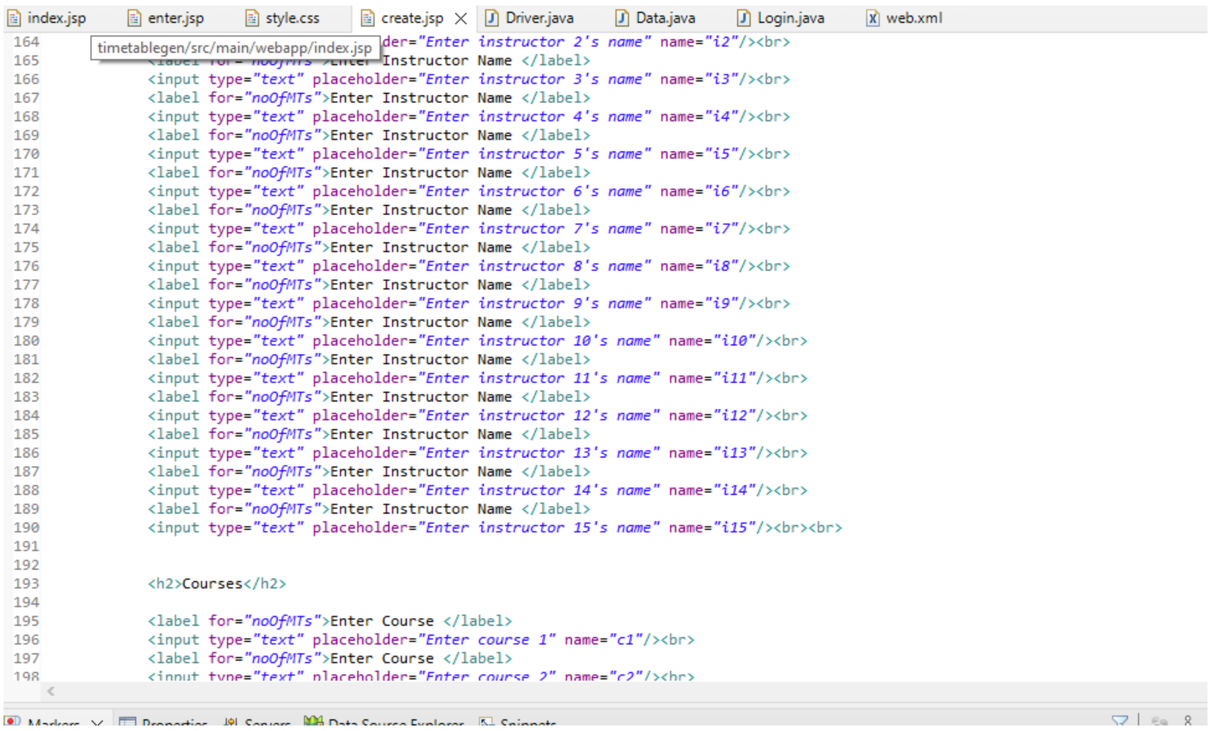
Module 3: The Data Web Module

The classes previously created in Data Module now need to be used to obtain the user’s input via JSP. This involves form creation and getting input from the user and redirecting the user to the processing page.

Create.jsp:



```
<body style="background-image: url('bg.png');">

<div id="header">
      <center> <h1>Timetable generator</h1></center>
</div>


    <div id="wrapper">
        <div id="content">
            <p><strong></strong></p>

        <!-- room mt instructor course dept  -->
        <form method="get" action="enter.jsp" style=" padding-left: 10%;">

            <h2>Rooms</h2>

            <label for="noOfMTs">Enter Room No </label>
            <input type="text" placeholder="Enter Room No" name="r1"/><br>
            <label for="noOfMTs">Enter Capacity </label>
            <input type="text" placeholder="Capacity" name="r1c"/><br><br>

            <label for="noOfMTs">Enter Room No </label>
            <input type="text" placeholder="Enter Room No" name="r2"/><br>
            <label for="noOfMTs">Enter Capacity </label>
            <input type="text" placeholder="Capacity" name="r2c"/><br><br>

            <label for="noOfMTs">Enter Room No </label>
            <input type="text" placeholder="Enter Room No" name="r3"/><br>
            <label for="noOfMTs">Enter Capacity </label>
            <input type="text" placeholder="Capacity" name="r3c"/><br><br>

            <h2>Instructors</h2>

            <label for="noOfMTs">Enter Instructor Name </label>
```

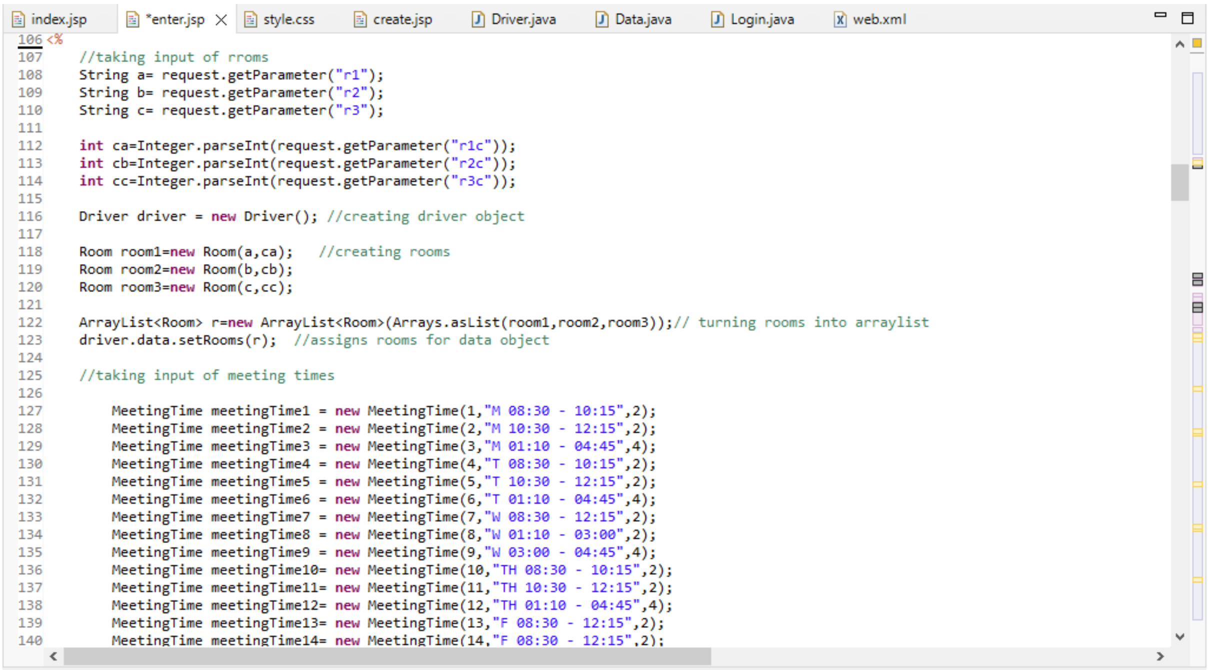


```
                                                    der="Enter instructor 2's name" name="i2"/><br>
                                                    Instructor Name </label>
            <input type="text" placeholder="Enter instructor 3's name" name="i3"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 4's name" name="i4"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 5's name" name="i5"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 6's name" name="i6"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 7's name" name="i7"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 8's name" name="i8"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 9's name" name="i9"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 10's name" name="i10"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 11's name" name="i11"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 12's name" name="i12"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 13's name" name="i13"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 14's name" name="i14"/><br>
            <label for="noOfMTs">Enter Instructor Name </label>
            <input type="text" placeholder="Enter instructor 15's name" name="i15"/><br><br>


            <h2>Courses</h2>

            <label for="noOfMTs">Enter Course </label>
            <input type="text" placeholder="Enter course 1" name="c1"/><br>
            <label for="noOfMTs">Enter Course </label>
            <input type="text" placeholder="Enter course 2" name="c2"/><br>
```

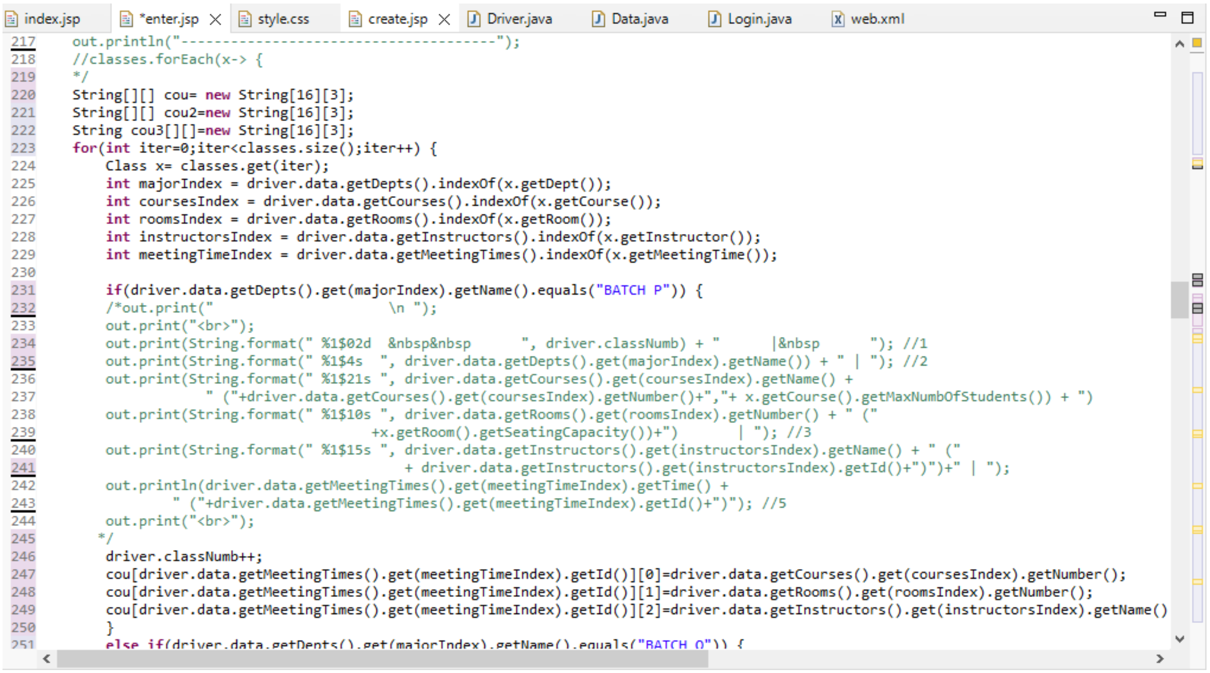
Module 4: The Genetic Algorithm Web Module

The inputs received via creation of forms and getting input from the user is redirected the user to the processing page. The algorithm here requires creation of population using the data already generated and running of schedules till the fitness mark has been reached. For this purpose, we need Population, Data, Driver, Schedule and Genetic Algorithm itself. Once the algorithm runs, the output is displayed as a time table.

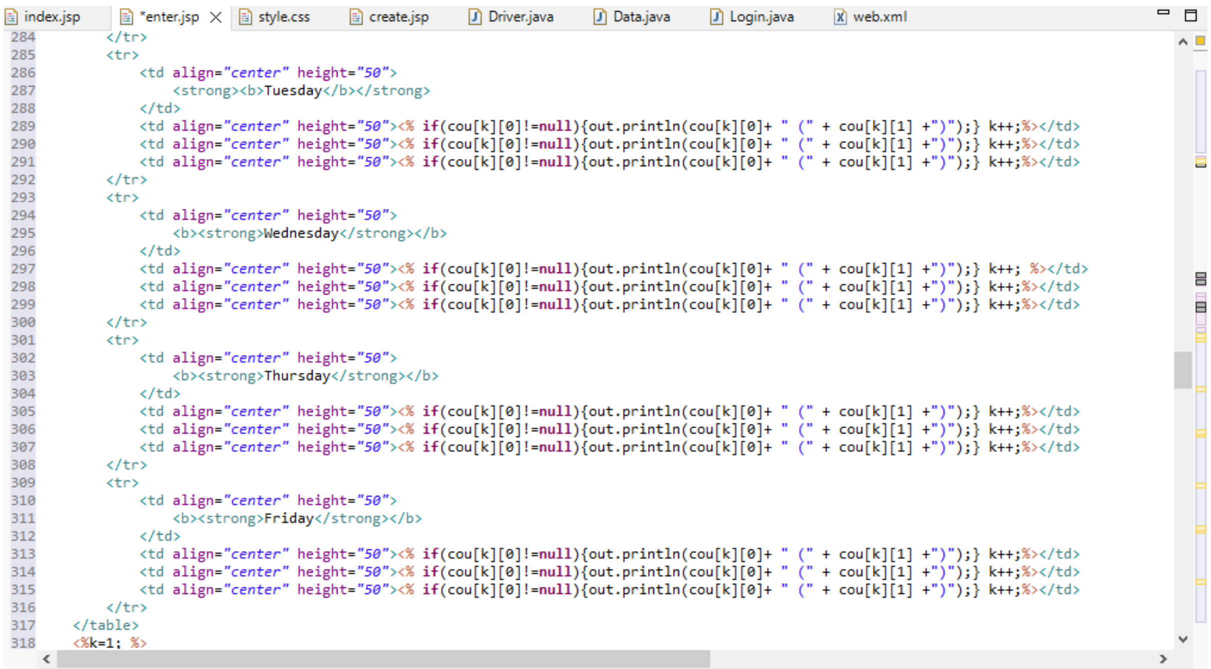
Enter.jsp



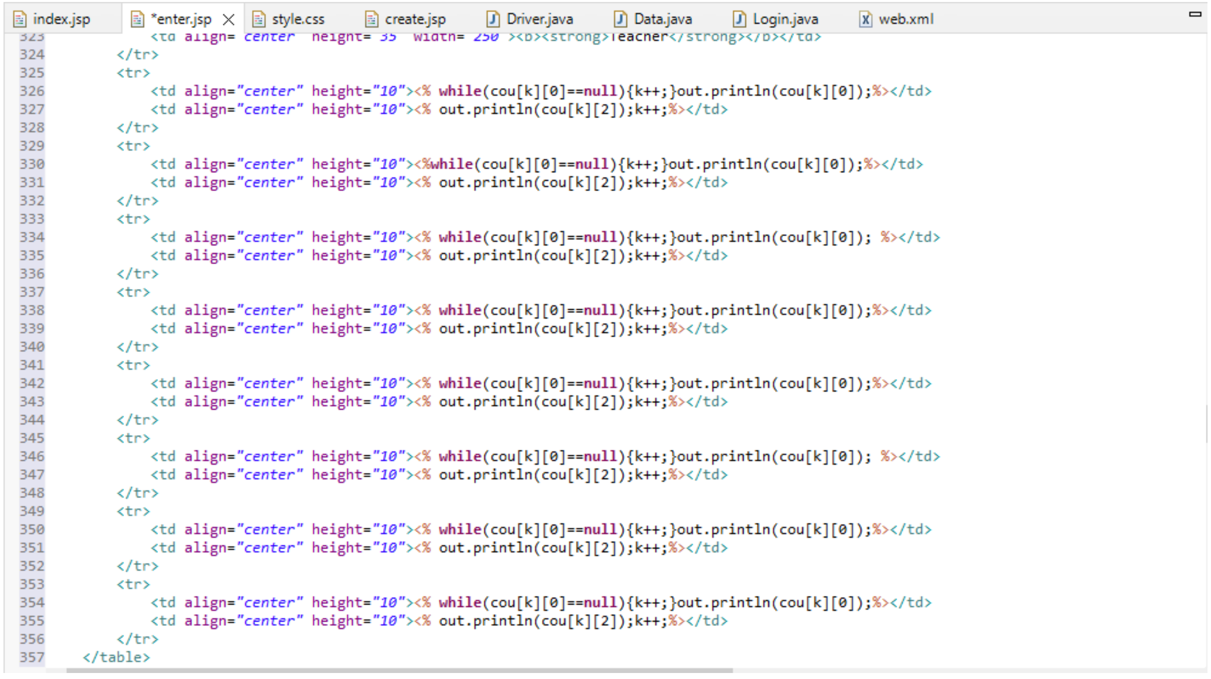
```
<%
    //taking input of rroms
    String a= request.getParameter("r1");
    String b= request.getParameter("r2");
    String c= request.getParameter("r3");

    int ca=Integer.parseInt(request.getParameter("r1c"));
    int cb=Integer.parseInt(request.getParameter("r2c"));
    int cc=Integer.parseInt(request.getParameter("r3c"));

    Driver driver = new Driver(); //creating driver object

    Room room1=new Room(a,ca);   //creating rooms
    Room room2=new Room(b,cb);
    Room room3=new Room(c,cc);

    ArrayList<Room> r=new ArrayList<Room>(Arrays.asList(room1,room2,room3));// turning rooms into arraylist
    driver.data.setRooms(r);  //assigns rooms for data object

    //taking input of meeting times

        MeetingTime meetingTime1 = new MeetingTime(1,"M 08:30 - 10:15",2);
        MeetingTime meetingTime2 = new MeetingTime(2,"M 10:30 - 12:15",2);
        MeetingTime meetingTime3 = new MeetingTime(3,"M 01:10 - 04:45",4);
        MeetingTime meetingTime4 = new MeetingTime(4,"T 08:30 - 10:15",2);
        MeetingTime meetingTime5 = new MeetingTime(5,"T 10:30 - 12:15",2);
        MeetingTime meetingTime6 = new MeetingTime(6,"T 01:10 - 04:45",4);
        MeetingTime meetingTime7 = new MeetingTime(7,"W 08:30 - 12:15",2);
        MeetingTime meetingTime8 = new MeetingTime(8,"W 01:10 - 03:00",2);
        MeetingTime meetingTime9 = new MeetingTime(9,"W 03:00 - 04:45",4);
        MeetingTime meetingTime10= new MeetingTime(10,"TH 08:30 - 10:15",2);
        MeetingTime meetingTime11= new MeetingTime(11,"TH 10:30 - 12:15",2);
        MeetingTime meetingTime12= new MeetingTime(12,"TH 01:10 - 04:45",4);
        MeetingTime meetingTime13= new MeetingTime(13,"F 08:30 - 12:15",2);
        MeetingTime meetingTime14= new MeetingTime(14,"F 08:30 - 12:15",2);
```



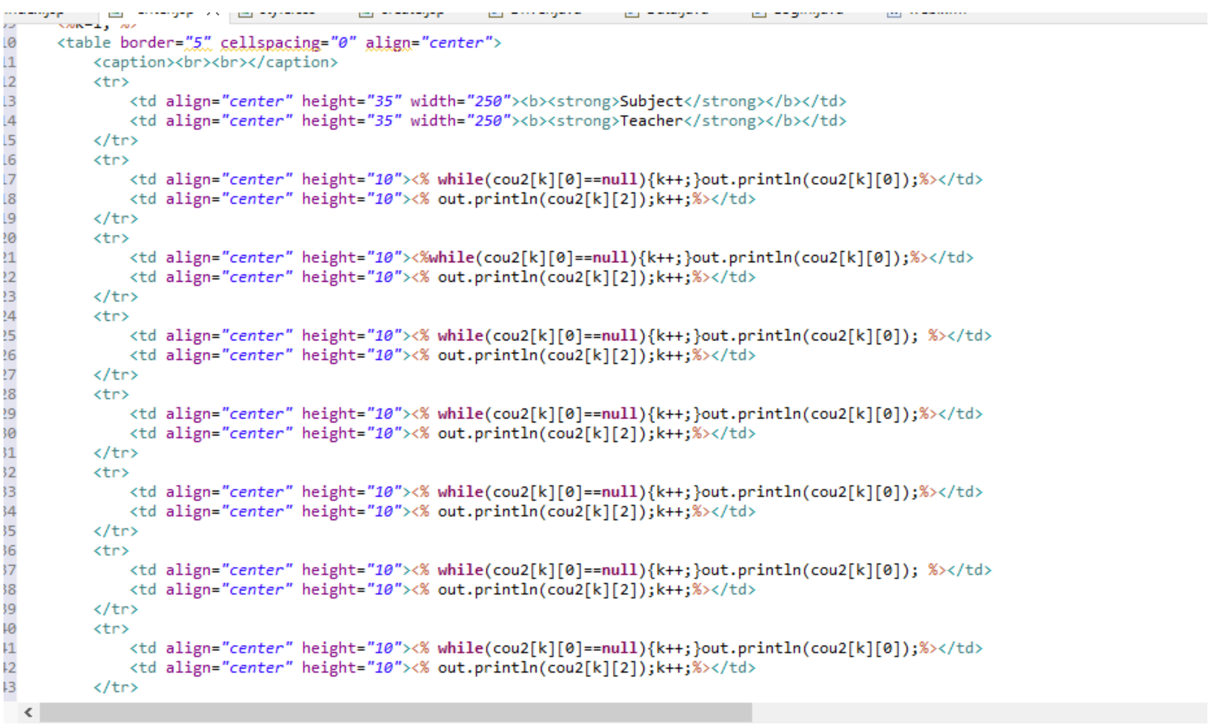
```
    out.println("-------------------------------------");
    //classes.forEach(x-> {
    */
    String[][] cou= new String[16][3];
    String[][] cou2=new String[16][3];
    String cou3[][]=new String[16][3];
    for(int iter=0;iter<classes.size();iter++) {
        Class x= classes.get(iter);
        int majorIndex = driver.data.getDepts().indexOf(x.getDept());
        int coursesIndex = driver.data.getCourses().indexOf(x.getCourse());
        int roomsIndex = driver.data.getRooms().indexOf(x.getRoom());
        int instructorsIndex = driver.data.getInstructors().indexOf(x.getInstructor());
        int meetingTimeIndex = driver.data.getMeetingTimes().indexOf(x.getMeetingTime());

        if(driver.data.getDepts().get(majorIndex).getName().equals("BATCH P")) {
        /*out.print("                     \n ");
        out.print("<br>");
        out.print(String.format(" %1$02d  &nbsp&nbsp      ", driver.classNumb) + "      |&nbsp      "); //1
        out.print(String.format(" %1$4s  ", driver.data.getDepts().get(majorIndex).getName()) + " | "); //2
        out.print(String.format(" %1$21s ", driver.data.getCourses().get(coursesIndex).getName() +
                    " ("+driver.data.getCourses().get(coursesIndex).getNumber()+","+ x.getCourse().getMaxNumbOfStudents()) + ")
        out.print(String.format(" %1$10s ", driver.data.getRooms().get(roomsIndex).getNumber() + " ("
                                +x.getRoom().getSeatingCapacity())+")       | "); //3
        out.print(String.format(" %1$15s ", driver.data.getInstructors().get(instructorsIndex).getName() + " ("
                                    + driver.data.getInstructors().get(instructorsIndex).getId()+")")+" | ");
        out.println(driver.data.getMeetingTimes().get(meetingTimeIndex).getTime() +
                " ("+driver.data.getMeetingTimes().get(meetingTimeIndex).getId()+")"); //5
        out.print("<br>");
       */
        driver.classNumb++;
        cou[driver.data.getMeetingTimes().get(meetingTimeIndex).getId()][0]=driver.data.getCourses().get(coursesIndex).getNumber();
        cou[driver.data.getMeetingTimes().get(meetingTimeIndex).getId()][1]=driver.data.getRooms().get(roomsIndex).getNumber();
        cou[driver.data.getMeetingTimes().get(meetingTimeIndex).getId()][2]=driver.data.getInstructors().get(instructorsIndex).getName()
        }
        else if(driver.data.getDepts().get(majorIndex).getName().equals("BATCH Q")) {
```

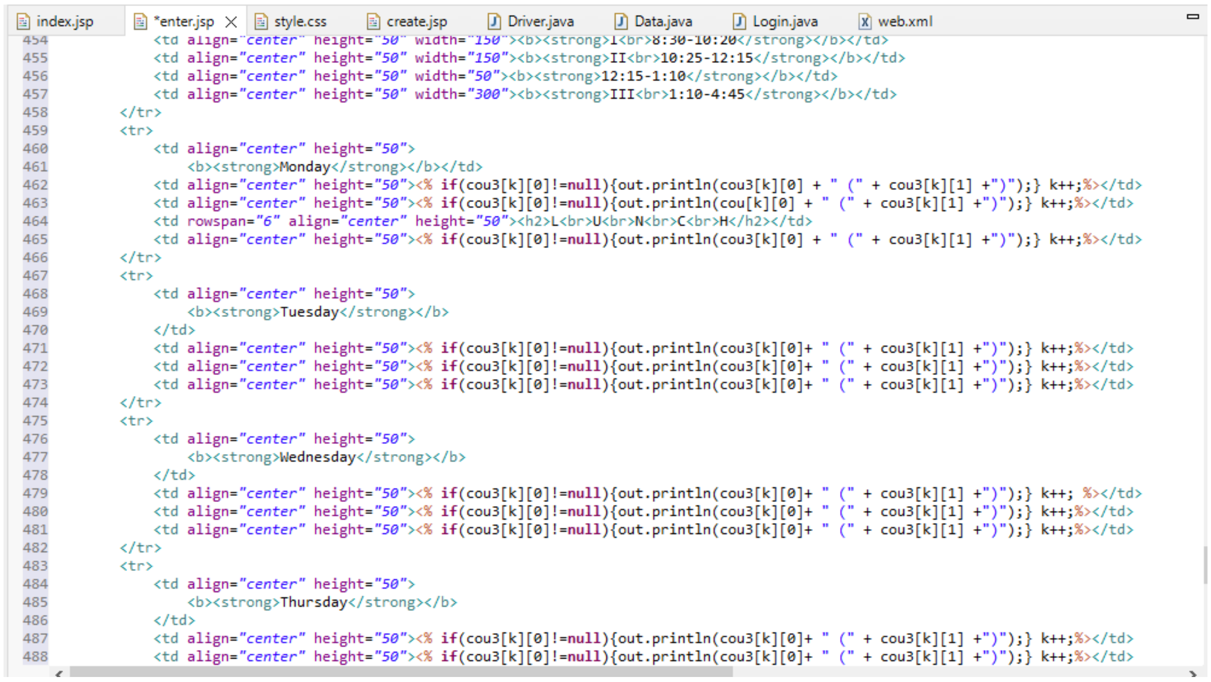


```
        </tr>
        <tr>
            <td align="center" height="50">
                <strong><b>Tuesday</b></strong>
            </td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b><strong>Wednesday</strong></b>
            </td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++; %></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b><strong>Thursday</strong></b>
            </td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b><strong>Friday</strong></b>
            </td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou[k][0]!=null){out.println(cou[k][0]+ " (" + cou[k][1] +")");} k++;%></td>
        </tr>
    </table>
    <%k=1; %>
```

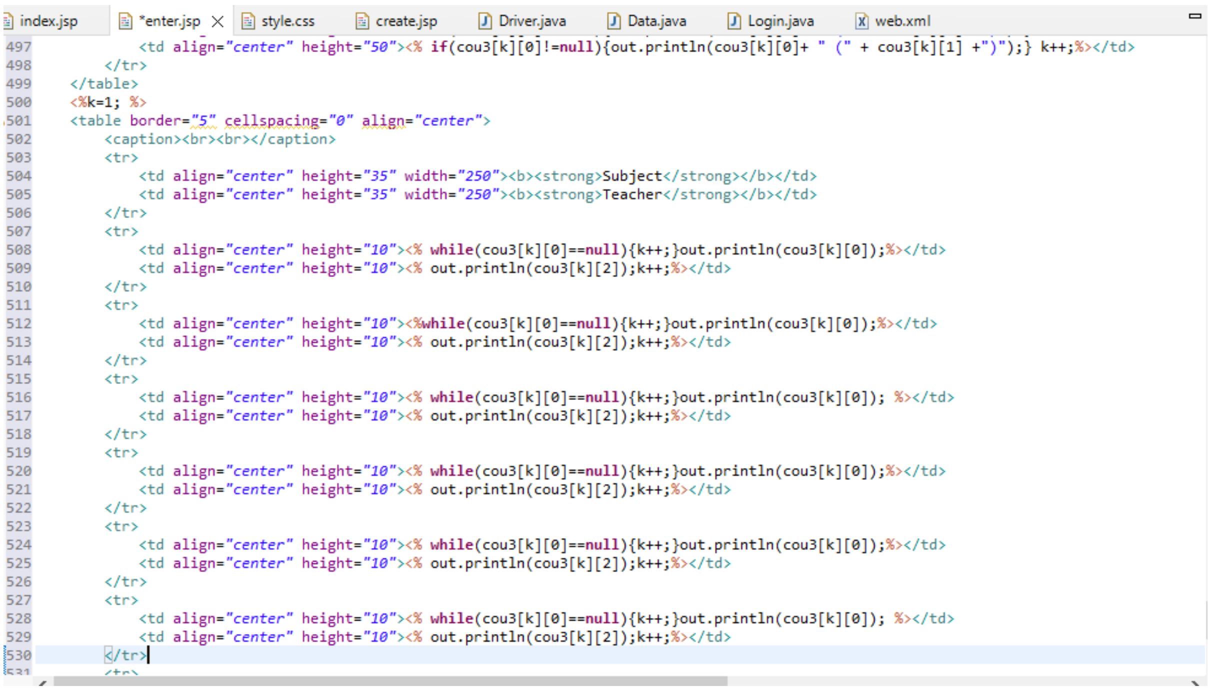


```
            <td align="center" height="35" width="250" ><b><strong>Teacher</strong></b></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou[k][0]==null){k++;}out.println(cou[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><%while(cou[k][0]==null){k++;}out.println(cou[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou[k][0]==null){k++;}out.println(cou[k][0]); %></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou[k][0]==null){k++;}out.println(cou[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou[k][0]==null){k++;}out.println(cou[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou[k][0]==null){k++;}out.println(cou[k][0]); %></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou[k][0]==null){k++;}out.println(cou[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou[k][0]==null){k++;}out.println(cou[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou[k][2]);k++;%></td>
        </tr>
    </table>
```


```
    <%k=1; %>
    <table border="5" cellspacing="0" align="center">
        <caption><br><br></caption>
        <tr>
            <td align="center" height="35" width="250"><b><strong>Subject</strong></b></td>
            <td align="center" height="35" width="250"><b><strong>Teacher</strong></b></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou2[k][0]==null){k++;}out.println(cou2[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou2[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><%while(cou2[k][0]==null){k++;}out.println(cou2[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou2[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou2[k][0]==null){k++;}out.println(cou2[k][0]); %></td>
            <td align="center" height="10"><% out.println(cou2[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou2[k][0]==null){k++;}out.println(cou2[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou2[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou2[k][0]==null){k++;}out.println(cou2[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou2[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou2[k][0]==null){k++;}out.println(cou2[k][0]); %></td>
            <td align="center" height="10"><% out.println(cou2[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou2[k][0]==null){k++;}out.println(cou2[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou2[k][2]);k++;%></td>
        </tr>
```



```
            <td align="center" height="50" width="150"><b><strong>I<br>8:30-10:20</strong></b></td>
            <td align="center" height="50" width="150"><b><strong>II<br>10:25-12:15</strong></b></td>
            <td align="center" height="50" width="50"><b><strong>12:15-1:10</strong></b></td>
            <td align="center" height="50" width="300"><b><strong>III<br>1:10-4:45</strong></b></td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b><strong>Monday</strong></b></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0] + " (" + cou3[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou[k][0] + " (" + cou3[k][1] +")");} k++;%></td>
            <td rowspan="6" align="center" height="50"><h2>L<br>U<br>N<br>C<br>H</h2></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0] + " (" + cou3[k][1] +")");} k++;%></td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b><strong>Tuesday</strong></b>
            </td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b><strong>Wednesday</strong></b>
            </td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++; %></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b><strong>Thursday</strong></b>
            </td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
```

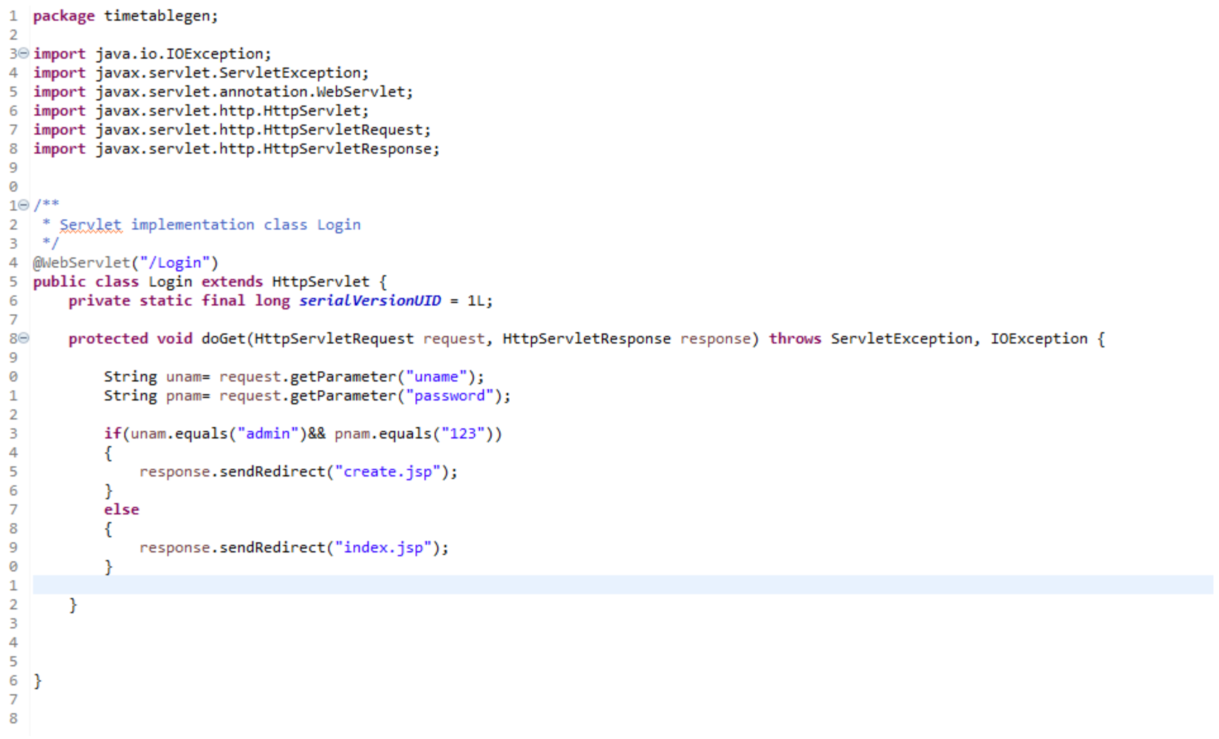


```
            <td align="center" height="50"><% if(cou3[k][0]!=null){out.println(cou3[k][0]+ " (" + cou3[k][1] +")");} k++;%></td>
        </tr>
    </table>
    <%k=1; %>
    <table border="5" cellspacing="0" align="center">
        <caption><br><br></caption>
        <tr>
            <td align="center" height="35" width="250"><b><strong>Subject</strong></b></td>
            <td align="center" height="35" width="250"><b><strong>Teacher</strong></b></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou3[k][0]==null){k++;}out.println(cou3[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou3[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><%while(cou3[k][0]==null){k++;}out.println(cou3[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou3[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou3[k][0]==null){k++;}out.println(cou3[k][0]); %></td>
            <td align="center" height="10"><% out.println(cou3[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou3[k][0]==null){k++;}out.println(cou3[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou3[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou3[k][0]==null){k++;}out.println(cou3[k][0]);%></td>
            <td align="center" height="10"><% out.println(cou3[k][2]);k++;%></td>
        </tr>
        <tr>
            <td align="center" height="10"><% while(cou3[k][0]==null){k++;}out.println(cou3[k][0]); %></td>
            <td align="center" height="10"><% out.println(cou3[k][2]);k++;%></td>
        </tr>
```

Supplementary Module: Login Module

A login component has been implemented via creation of servlet.

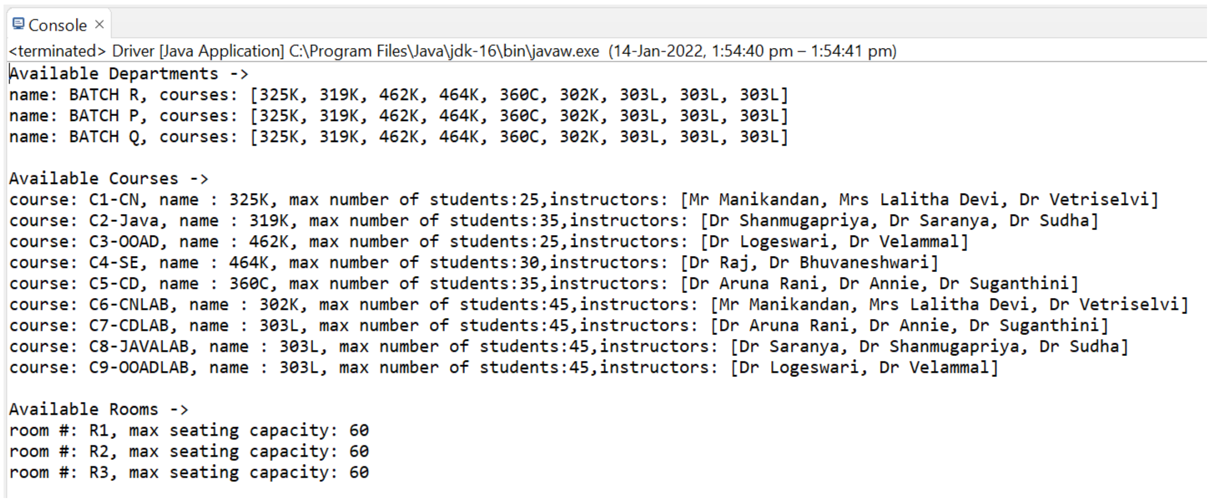
Login.java



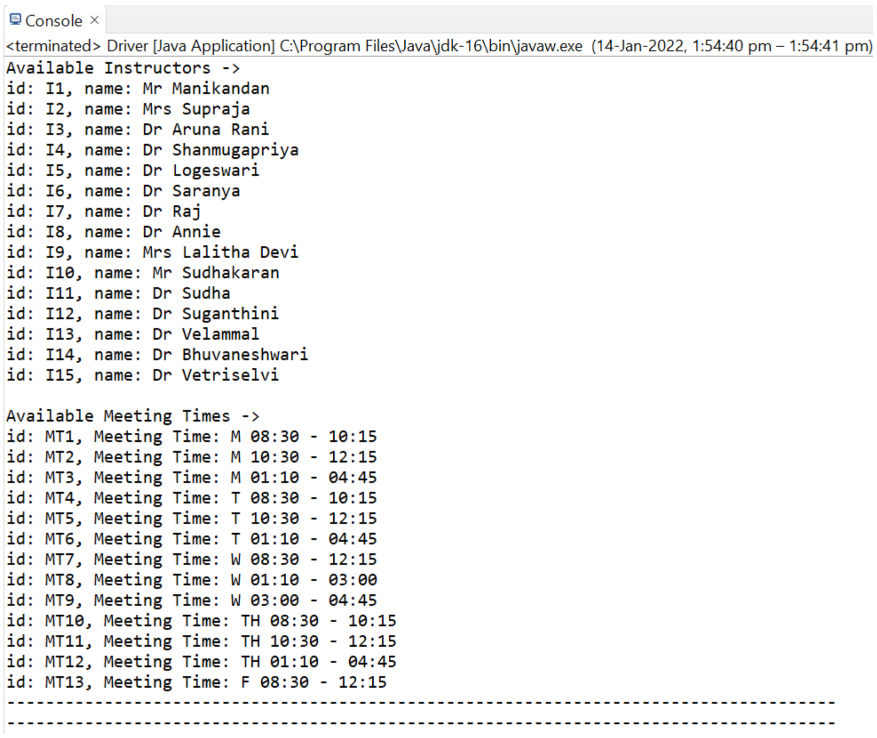
```java
package timetablegen;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * Servlet implementation class Login
 */
@WebServlet("/Login")
public class Login extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String unam= request.getParameter("uname");
        String pnam= request.getParameter("password");

        if(unam.equals("admin")&& pnam.equals("123"))
        {
            response.sendRedirect("create.jsp");
        }
        else
        {
            response.sendRedirect("index.jsp");
        }

    }



}


```

RESULTS

Printing out all the data

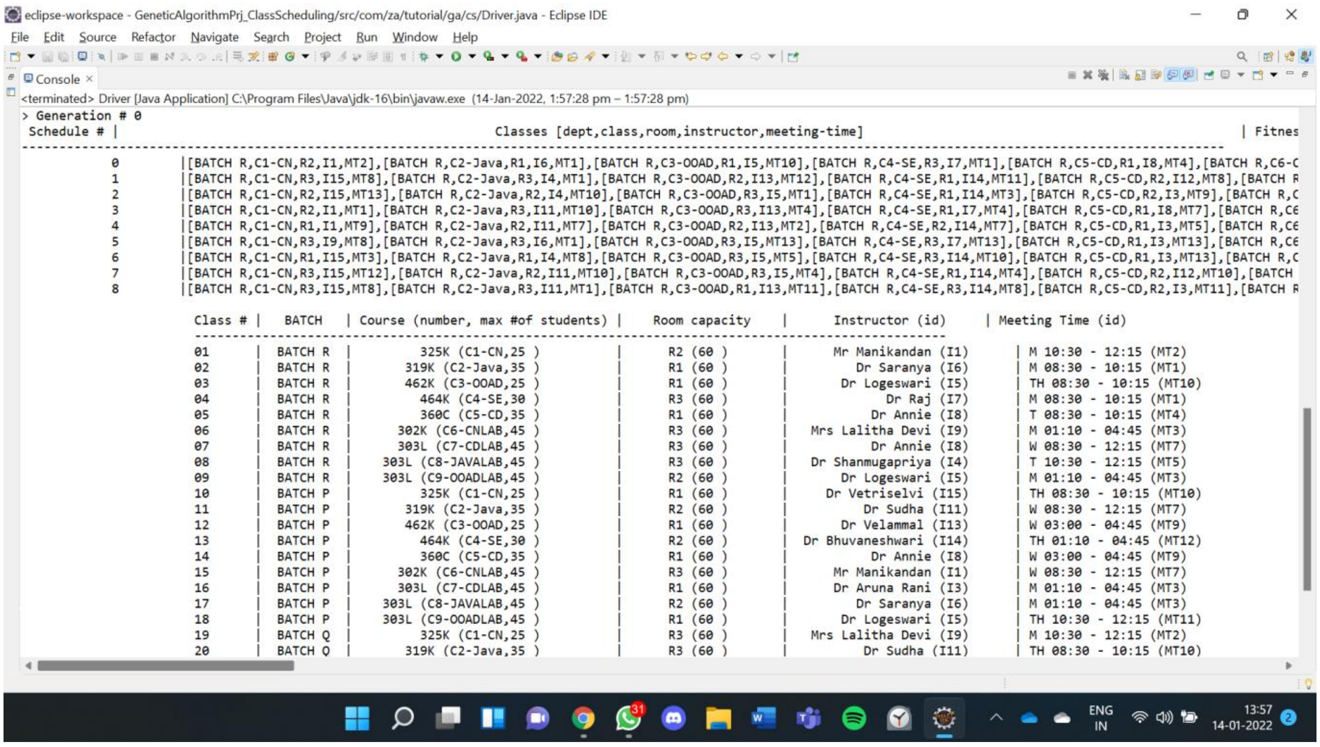


```
<terminated> Driver [Java Application] C:\Program Files\Java\jdk-16\bin\javaw.exe  (14-Jan-2022, 1:54:40 pm – 1:54:41 pm)
Available Departments ->
name: BATCH R, courses: [325K, 319K, 462K, 464K, 360C, 302K, 303L, 303L, 303L]
name: BATCH P, courses: [325K, 319K, 462K, 464K, 360C, 302K, 303L, 303L, 303L]
name: BATCH Q, courses: [325K, 319K, 462K, 464K, 360C, 302K, 303L, 303L, 303L]

Available Courses ->
course: C1-CN, name : 325K, max number of students:25,instructors: [Mr Manikandan, Mrs Lalitha Devi, Dr Vetriselvi]
course: C2-Java, name : 319K, max number of students:35,instructors: [Dr Shanmugapriya, Dr Saranya, Dr Sudha]
course: C3-OOAD, name : 462K, max number of students:25,instructors: [Dr Logeswari, Dr Velammal]
course: C4-SE, name : 464K, max number of students:30,instructors: [Dr Raj, Dr Bhuvaneshwari]
course: C5-CD, name : 360C, max number of students:35,instructors: [Dr Aruna Rani, Dr Annie, Dr Suganthini]
course: C6-CNLAB, name : 302K, max number of students:45,instructors: [Mr Manikandan, Mrs Lalitha Devi, Dr Vetriselvi]
course: C7-CDLAB, name : 303L, max number of students:45,instructors: [Dr Aruna Rani, Dr Annie, Dr Suganthini]
course: C8-JAVALAB, name : 303L, max number of students:45,instructors: [Dr Saranya, Dr Shanmugapriya, Dr Sudha]
course: C9-OOADLAB, name : 303L, max number of students:45,instructors: [Dr Logeswari, Dr Velammal]

Available Rooms ->
room #: R1, max seating capacity: 60
room #: R2, max seating capacity: 60
room #: R3, max seating capacity: 60
```



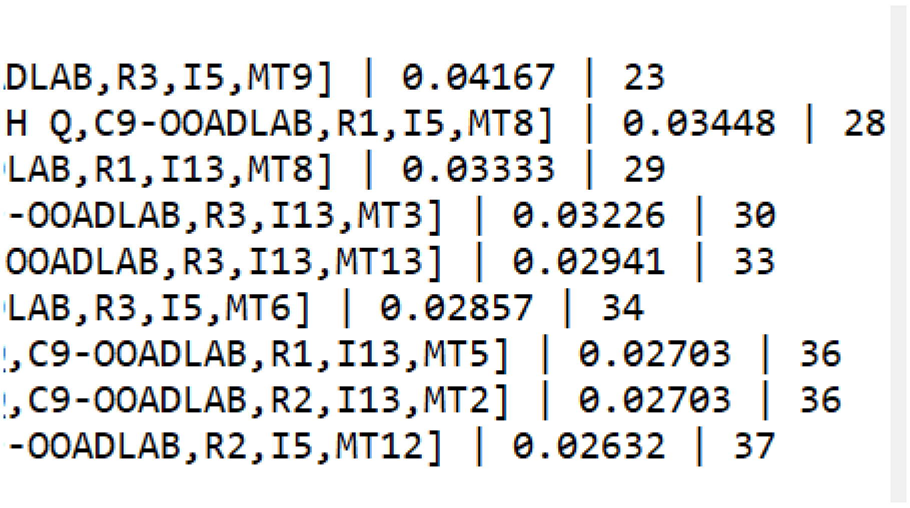
```
<terminated> Driver [Java Application] C:\Program Files\Java\jdk-16\bin\javaw.exe  (14-Jan-2022, 1:54:40 pm – 1:54:41 pm)
Available Instructors ->
id: I1, name: Mr Manikandan
id: I2, name: Mrs Supraja
id: I3, name: Dr Aruna Rani
id: I4, name: Dr Shanmugapriya
id: I5, name: Dr Logeswari
id: I6, name: Dr Saranya
id: I7, name: Dr Raj
id: I8, name: Dr Annie
id: I9, name: Mrs Lalitha Devi
id: I10, name: Mr Sudhakaran
id: I11, name: Dr Sudha
id: I12, name: Dr Suganthini
id: I13, name: Dr Velammal
id: I14, name: Dr Bhuvaneshwari
id: I15, name: Dr Vetriselvi

Available Meeting Times ->
id: MT1, Meeting Time: M 08:30 - 10:15
id: MT2, Meeting Time: M 10:30 - 12:15
id: MT3, Meeting Time: M 01:10 - 04:45
id: MT4, Meeting Time: T 08:30 - 10:15
id: MT5, Meeting Time: T 10:30 - 12:15
id: MT6, Meeting Time: T 01:10 - 04:45
id: MT7, Meeting Time: W 08:30 - 12:15
id: MT8, Meeting Time: W 01:10 - 03:00
id: MT9, Meeting Time: W 03:00 - 04:45
id: MT10, Meeting Time: TH 08:30 - 10:15
id: MT11, Meeting Time: TH 10:30 - 12:15
id: MT12, Meeting Time: TH 01:10 - 04:45
id: MT13, Meeting Time: F 08:30 - 12:15
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
```

Printing out the initial Schedule of Generation #0


eclipse-workspace - GeneticAlgorithmPrj_ClassScheduling/src/com/za/tutorial/ga/cs/Driver.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
ENG
IN
13:57
14-01-2022

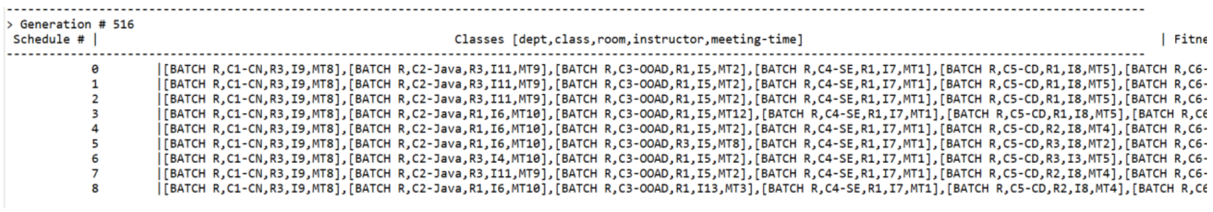
```
<terminated> Driver [Java Application] C:\Program Files\Java\jdk-16\bin\javaw.exe  (14-Jan-2022, 1:57:28 pm – 1:57:28 pm)
> Generation # 0
 Schedule # |                                               Classes [dept,class,room,instructor,meeting-time]                                  | Fitnes
-----------------------------------------------------------------------------------------------------------------------------------------------
           0        |[BATCH R,C1-CN,R2,I1,MT2],[BATCH R,C2-Java,R1,I6,MT1],[BATCH R,C3-OOAD,R1,I5,MT10],[BATCH R,C4-SE,R3,I7,MT1],[BATCH R,C5-CD,R1,I8,MT4],[BATCH R,C6-C
           1        |[BATCH R,C1-CN,R3,I15,MT8],[BATCH R,C2-Java,R3,I4,MT1],[BATCH R,C3-OOAD,R2,I13,MT12],[BATCH R,C4-SE,R1,I14,MT11],[BATCH R,C5-CD,R2,I12,MT8],[BATCH R
           2        |[BATCH R,C1-CN,R2,I15,MT13],[BATCH R,C2-Java,R2,I4,MT10],[BATCH R,C3-OOAD,R3,I5,MT1],[BATCH R,C4-SE,R1,I14,MT3],[BATCH R,C5-CD,R2,I3,MT9],[BATCH R,C
           3        |[BATCH R,C1-CN,R2,I1,MT1],[BATCH R,C2-Java,R3,I11,MT10],[BATCH R,C3-OOAD,R3,I13,MT4],[BATCH R,C4-SE,R1,I7,MT4],[BATCH R,C5-CD,R1,I8,MT7],[BATCH R,C6
           4        |[BATCH R,C1-CN,R1,I1,MT9],[BATCH R,C2-Java,R2,I11,MT7],[BATCH R,C3-OOAD,R2,I13,MT2],[BATCH R,C4-SE,R2,I14,MT7],[BATCH R,C5-CD,R1,I3,MT5],[BATCH R,C6
           5        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R3,I6,MT1],[BATCH R,C3-OOAD,R3,I5,MT13],[BATCH R,C4-SE,R3,I7,MT13],[BATCH R,C5-CD,R1,I3,MT13],[BATCH R,C6
           6        |[BATCH R,C1-CN,R1,I15,MT3],[BATCH R,C2-Java,R1,I4,MT8],[BATCH R,C3-OOAD,R3,I5,MT5],[BATCH R,C4-SE,R3,I14,MT10],[BATCH R,C5-CD,R1,I3,MT13],[BATCH R,C
           7        |[BATCH R,C1-CN,R3,I15,MT12],[BATCH R,C2-Java,R2,I11,MT10],[BATCH R,C3-OOAD,R3,I5,MT4],[BATCH R,C4-SE,R1,I14,MT4],[BATCH R,C5-CD,R2,I12,MT10],[BATCH
           8        |[BATCH R,C1-CN,R3,I15,MT8],[BATCH R,C2-Java,R3,I11,MT1],[BATCH R,C3-OOAD,R1,I13,MT11],[BATCH R,C4-SE,R3,I14,MT8],[BATCH R,C5-CD,R2,I3,MT11],[BATCH R

                    Class # |   BATCH    | Course (number, max #of students) |    Room capacity     |        Instructor (id)      | Meeting Time (id)
                    -----------------------------------------------------------------------------------------------------------
                    01      |  BATCH R  |         325K (C1-CN,25 )          |      R2 (60 )        |      Mr Manikandan (I1)      | M 10:30 - 12:15 (MT2)
                    02      |  BATCH R  |       319K (C2-Java,35 )          |      R1 (60 )        |         Dr Saranya (I6)      | M 08:30 - 10:15 (MT1)
                    03      |  BATCH R  |       462K (C3-OOAD,25 )          |      R1 (60 )        |       Dr Logeswari (I5)      | TH 08:30 - 10:15 (MT10)
                    04      |  BATCH R  |         464K (C4-SE,30 )          |      R3 (60 )        |             Dr Raj (I7)      | M 08:30 - 10:15 (MT1)
                    05      |  BATCH R  |         360C (C5-CD,35 )          |      R1 (60 )        |           Dr Annie (I8)      | T 08:30 - 10:15 (MT4)
                    06      |  BATCH R  |      302K (C6-CNLAB,45 )          |      R3 (60 )        |   Mrs Lalitha Devi (I9)      | M 01:10 - 04:45 (MT3)
                    07      |  BATCH R  |      303L (C7-CDLAB,45 )          |      R3 (60 )        |           Dr Annie (I8)      | W 08:30 - 12:15 (MT7)
                    08      |  BATCH R  |    303L (C8-JAVALAB,45 )          |      R3 (60 )        |   Dr Shanmugapriya (I4)      | T 10:30 - 12:15 (MT5)
                    09      |  BATCH R  |    303L (C9-OOADLAB,45 )          |      R2 (60 )        |       Dr Logeswari (I5)      | M 01:10 - 04:45 (MT3)
                    10      |  BATCH P  |         325K (C1-CN,25 )          |      R1 (60 )        |     Dr Vetriselvi (I15)      | TH 08:30 - 10:15 (MT10)
                    11      |  BATCH P  |       319K (C2-Java,35 )          |      R2 (60 )        |         Dr Sudha (I11)      | W 08:30 - 12:15 (MT7)
                    12      |  BATCH P  |       462K (C3-OOAD,25 )          |      R1 (60 )        |      Dr Velammal (I13)      | W 03:00 - 04:45 (MT9)
                    13      |  BATCH P  |         464K (C4-SE,30 )          |      R2 (60 )        | Dr Bhuvaneshwari (I14)      | TH 01:10 - 04:45 (MT12)
                    14      |  BATCH P  |         360C (C5-CD,35 )          |      R1 (60 )        |           Dr Annie (I8)      | W 03:00 - 04:45 (MT9)
                    15      |  BATCH P  |      302K (C6-CNLAB,45 )          |      R3 (60 )        |      Mr Manikandan (I1)      | W 08:30 - 12:15 (MT7)
                    16      |  BATCH P  |      303L (C7-CDLAB,45 )          |      R1 (60 )        |      Dr Aruna Rani (I3)      | M 01:10 - 04:45 (MT3)
                    17      |  BATCH P  |    303L (C8-JAVALAB,45 )          |      R2 (60 )        |         Dr Saranya (I6)      | M 01:10 - 04:45 (MT3)
                    18      |  BATCH P  |    303L (C9-OOADLAB,45 )          |      R1 (60 )        |       Dr Logeswari (I5)      | TH 10:30 - 12:15 (MT11)
                    19      |  BATCH Q  |         325K (C1-CN,25 )          |      R3 (60 )        |   Mrs Lalitha Devi (I9)      | M 10:30 - 12:15 (MT2)
                    20      |  BATCH Q  |       319K (C2-Java,35 )          |      R3 (60 )        |         Dr Sudha (I11)      | TH 08:30 - 10:15 (MT10)
```

Fitnessrate and the number of conflicts of the Generation

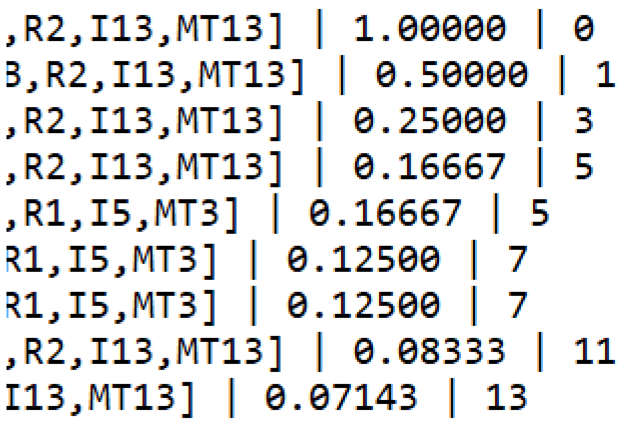


```
DLAB,R3,I5,MT9] | 0.04167 | 23
H Q,C9-OOADLAB,R1,I5,MT8] | 0.03448 | 28
LAB,R1,I13,MT8] | 0.03333 | 29
-OOADLAB,R3,I13,MT3] | 0.03226 | 30
OOADLAB,R3,I13,MT13] | 0.02941 | 33
LAB,R3,I5,MT6] | 0.02857 | 34
,C9-OOADLAB,R1,I13,MT5] | 0.02703 | 36
,C9-OOADLAB,R2,I13,MT2] | 0.02703 | 36
-OOADLAB,R2,I5,MT12] | 0.02632 | 37
```

After running the program through the while loop, in Gen #516, a schedule with Fitness 1 has been found.

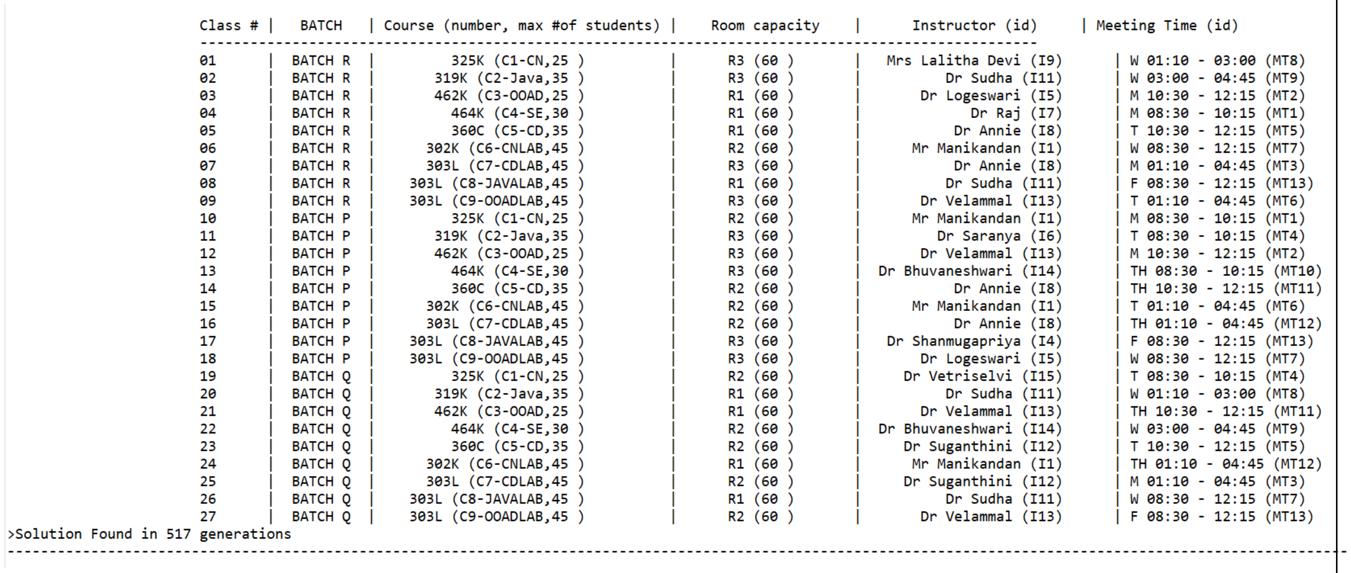


```
-------------------------------------------------------------------------------------------------------------------------------
> Generation # 516
 Schedule # |                                          Classes [dept,class,room,instructor,meeting-time]                            | Fitne
-------------------------------------------------------------------------------------------------------------------------------
           0        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R3,I11,MT9],[BATCH R,C3-OOAD,R1,I5,MT2],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R1,I8,MT5],[BATCH R,C6-
           1        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R3,I11,MT9],[BATCH R,C3-OOAD,R1,I5,MT2],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R1,I8,MT5],[BATCH R,C6-
           2        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R3,I11,MT9],[BATCH R,C3-OOAD,R1,I5,MT2],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R1,I8,MT5],[BATCH R,C6-
           3        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R1,I6,MT10],[BATCH R,C3-OOAD,R1,I5,MT12],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R1,I8,MT5],[BATCH R,C6
           4        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R1,I6,MT10],[BATCH R,C3-OOAD,R1,I5,MT2],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R2,I8,MT4],[BATCH R,C6-
           5        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R1,I6,MT10],[BATCH R,C3-OOAD,R3,I5,MT8],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R3,I8,MT2],[BATCH R,C6-
           6        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R3,I4,MT10],[BATCH R,C3-OOAD,R1,I5,MT2],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R3,I3,MT5],[BATCH R,C6-
           7        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R3,I11,MT9],[BATCH R,C3-OOAD,R1,I5,MT2],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R2,I8,MT4],[BATCH R,C6-
           8        |[BATCH R,C1-CN,R3,I9,MT8],[BATCH R,C2-Java,R1,I6,MT10],[BATCH R,C3-OOAD,R1,I13,MT3],[BATCH R,C4-SE,R1,I7,MT1],[BATCH R,C5-CD,R2,I8,MT4],[BATCH R,C6
```



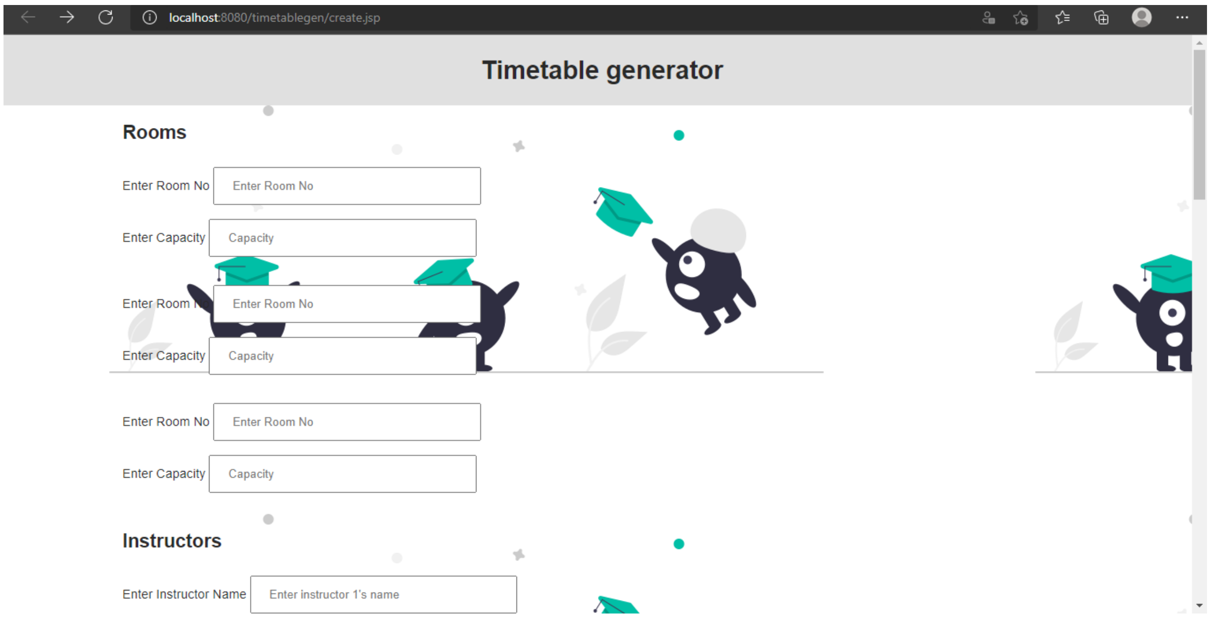
```
,R2,I13,MT13] | 1.00000 | 0
3,R2,I13,MT13] | 0.50000 | 1
,R2,I13,MT13] | 0.25000 | 3
,R2,I13,MT13] | 0.16667 | 5
,R1,I5,MT3] | 0.16667 | 5
R1,I5,MT3] | 0.12500 | 7
R1,I5,MT3] | 0.12500 | 7
,R2,I13,MT13] | 0.08333 | 11
I13,MT13] | 0.07143 | 13
```

Final TimeTable:



```
                        Class # |   BATCH   | Course (number, max #of students) |    Room capacity     |      Instructor (id)      | Meeting Time (id)
                        ----------------------------------------------------------------------------------------------------------
                        01      |  BATCH R  |        325K (C1-CN,25 )           |      R3 (60 )        |   Mrs Lalitha Devi (I9)     | W 01:10 - 03:00 (MT8)
                        02      |  BATCH R  |      319K (C2-Java,35 )           |      R3 (60 )        |         Dr Sudha (I11)      | W 03:00 - 04:45 (MT9)
                        03      |  BATCH R  |      462K (C3-OOAD,25 )           |      R1 (60 )        |      Dr Logeswari (I5)      | M 10:30 - 12:15 (MT2)
                        04      |  BATCH R  |        464K (C4-SE,30 )           |      R1 (60 )        |            Dr Raj (I7)      | M 08:30 - 10:15 (MT1)
                        05      |  BATCH R  |        360C (C5-CD,35 )           |      R1 (60 )        |          Dr Annie (I8)      | T 10:30 - 12:15 (MT5)
                        06      |  BATCH R  |     302K (C6-CNLAB,45 )           |      R2 (60 )        |     Mr Manikandan (I1)      | W 08:30 - 12:15 (MT7)
                        07      |  BATCH R  |     303L (C7-CDLAB,45 )           |      R3 (60 )        |          Dr Annie (I8)      | M 01:10 - 04:45 (MT3)
                        08      |  BATCH R  |   303L (C8-JAVALAB,45 )           |      R1 (60 )        |         Dr Sudha (I11)      | F 08:30 - 12:15 (MT13)
                        09      |  BATCH R  |   303L (C9-OOADLAB,45 )           |      R3 (60 )        |      Dr Velammal (I13)      | T 01:10 - 04:45 (MT6)
                        10      |  BATCH P  |        325K (C1-CN,25 )           |      R2 (60 )        |     Mr Manikandan (I1)      | M 08:30 - 10:15 (MT1)
                        11      |  BATCH P  |      319K (C2-Java,35 )           |      R3 (60 )        |        Dr Saranya (I6)      | T 08:30 - 10:15 (MT4)
                        12      |  BATCH P  |      462K (C3-OOAD,25 )           |      R3 (60 )        |      Dr Velammal (I13)      | M 10:30 - 12:15 (MT2)
                        13      |  BATCH P  |        464K (C4-SE,30 )           |      R3 (60 )        |  Dr Bhuvaneshwari (I14)     | TH 08:30 - 10:15 (MT10)
                        14      |  BATCH P  |        360C (C5-CD,35 )           |      R2 (60 )        |          Dr Annie (I8)      | TH 10:30 - 12:15 (MT11)
                        15      |  BATCH P  |     302K (C6-CNLAB,45 )           |      R2 (60 )        |     Mr Manikandan (I1)      | T 01:10 - 04:45 (MT6)
                        16      |  BATCH P  |     303L (C7-CDLAB,45 )           |      R2 (60 )        |          Dr Annie (I8)      | TH 01:10 - 04:45 (MT12)
                        17      |  BATCH P  |   303L (C8-JAVALAB,45 )           |      R3 (60 )        |   Dr Shanmugapriya (I4)     | F 08:30 - 12:15 (MT13)
                        18      |  BATCH P  |   303L (C9-OOADLAB,45 )           |      R3 (60 )        |      Dr Logeswari (I5)      | W 08:30 - 12:15 (MT7)
                        19      |  BATCH Q  |        325K (C1-CN,25 )           |      R2 (60 )        |    Dr Vetriselvi (I15)      | T 08:30 - 10:15 (MT4)
                        20      |  BATCH Q  |      319K (C2-Java,35 )           |      R1 (60 )        |         Dr Sudha (I11)      | W 01:10 - 03:00 (MT8)
                        21      |  BATCH Q  |      462K (C3-OOAD,25 )           |      R1 (60 )        |      Dr Velammal (I13)      | TH 10:30 - 12:15 (MT11)
                        22      |  BATCH Q  |        464K (C4-SE,30 )           |      R2 (60 )        |  Dr Bhuvaneshwari (I14)     | W 03:00 - 04:45 (MT9)
                        23      |  BATCH Q  |        360C (C5-CD,35 )           |      R2 (60 )        |    Dr Suganthini (I12)      | T 10:30 - 12:15 (MT5)
                        24      |  BATCH Q  |     302K (C6-CNLAB,45 )           |      R1 (60 )        |     Mr Manikandan (I1)      | TH 01:10 - 04:45 (MT12)
                        25      |  BATCH Q  |     303L (C7-CDLAB,45 )           |      R2 (60 )        |    Dr Suganthini (I12)      | M 01:10 - 04:45 (MT3)
                        26      |  BATCH Q  |   303L (C8-JAVALAB,45 )           |      R1 (60 )        |         Dr Sudha (I11)      | W 08:30 - 12:15 (MT7)
                        27      |  BATCH Q  |   303L (C9-OOADLAB,45 )           |      R2 (60 )        |      Dr Velammal (I13)      | F 08:30 - 12:15 (MT13)
>Solution Found in 517 generations
------------------------------------------------------------------------------------------------------------------------------------------
```

Web Module Views:



localhost:8080/timetablegen/create.jsp

Timetable generator

Rooms

Enter Room No [Enter Room No]

Enter Capacity [Capacity]

Enter Room No [Enter Room No]

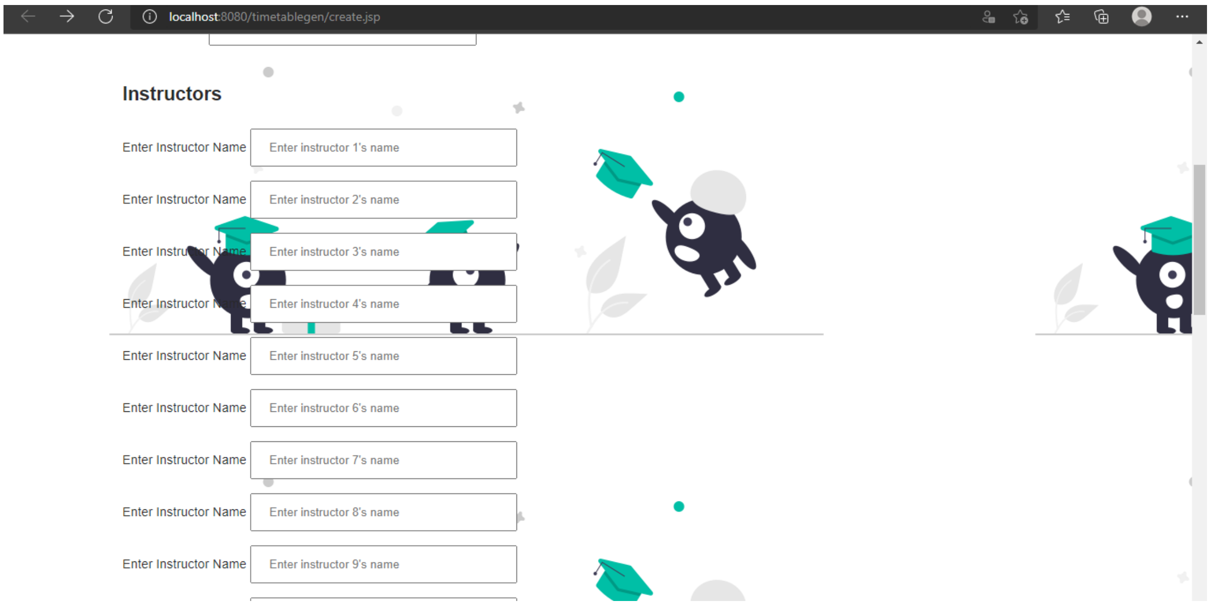
Enter Capacity [Capacity]

Enter Room No [Enter Room No]

Enter Capacity [Capacity]

Instructors

Enter Instructor Name [Enter instructor 1's name]



localhost:8080/timetablegen/create.jsp

Instructors

Enter Instructor Name [Enter instructor 1's name]

Enter Instructor Name [Enter instructor 2's name]

Enter Instructor Name [Enter instructor 3's name]

Enter Instructor Name [Enter instructor 4's name]

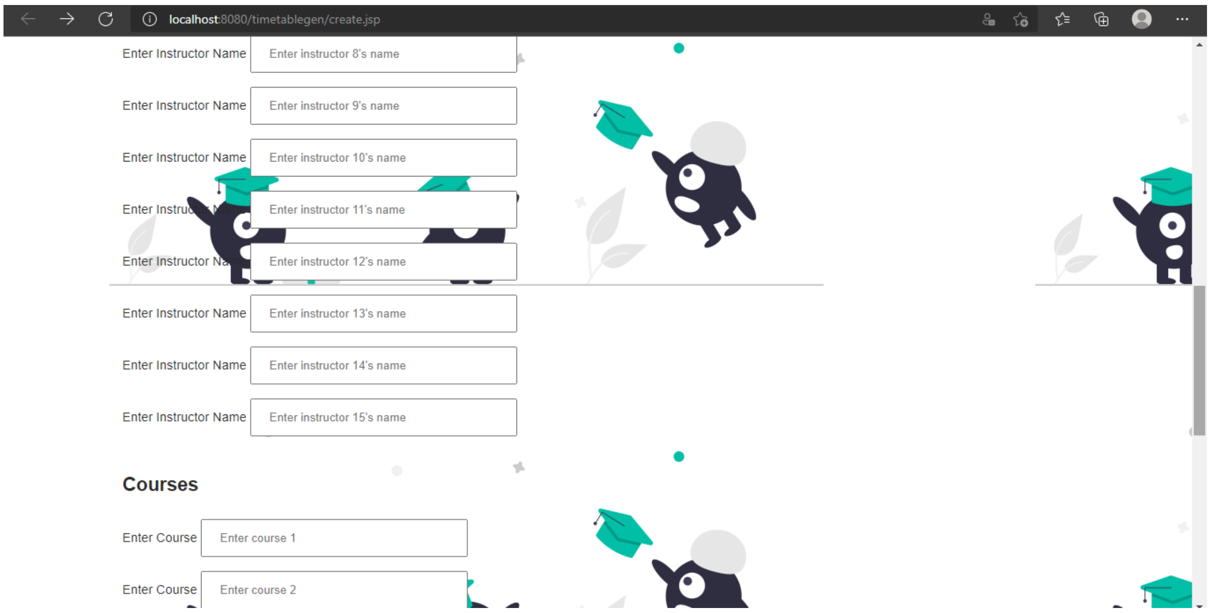
Enter Instructor Name [Enter instructor 5's name]

Enter Instructor Name [Enter instructor 6's name]

Enter Instructor Name [Enter instructor 7's name]

Enter Instructor Name [Enter instructor 8's name]

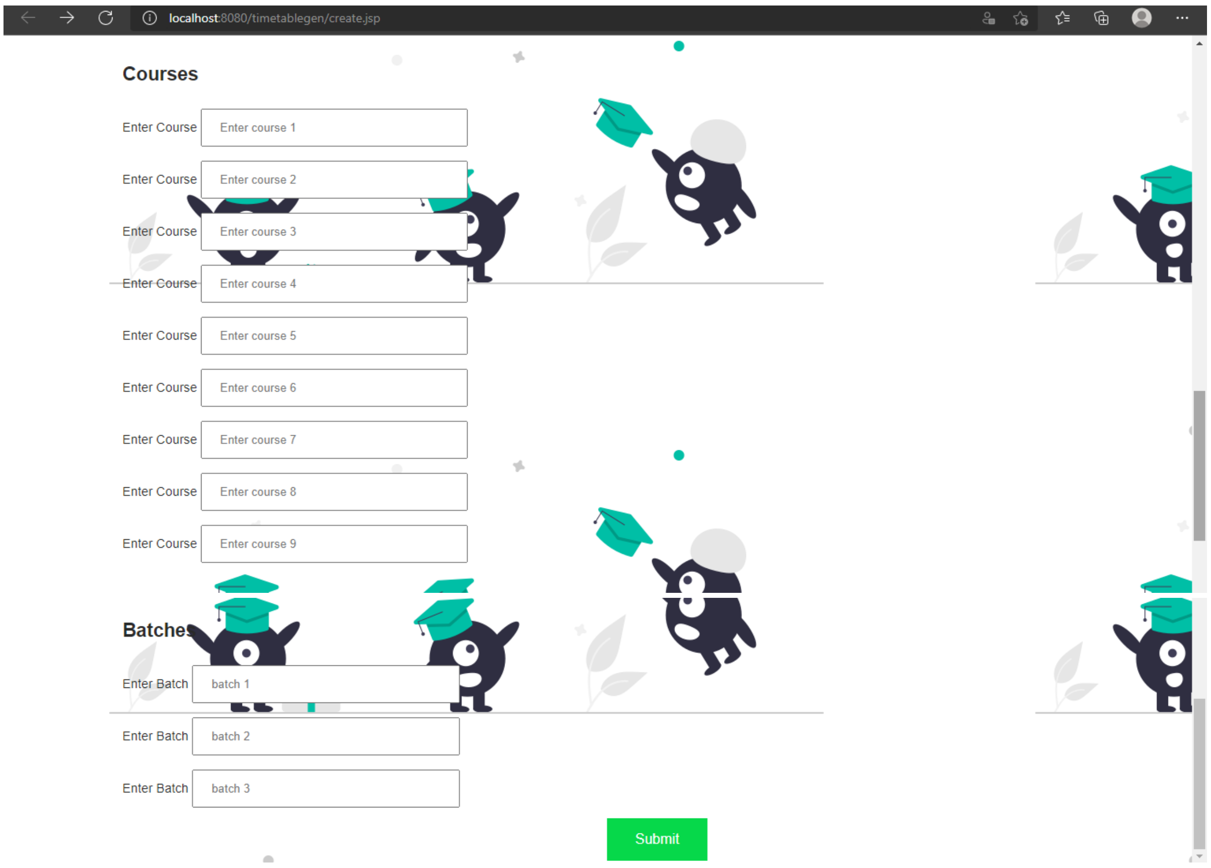
Enter Instructor Name [Enter instructor 9's name]


localhost:8080/timetablegen/create.jsp

Enter Instructor Name | Enter instructor 8's name
Enter Instructor Name | Enter instructor 9's name
Enter Instructor Name | Enter instructor 10's name
Enter Instructor Name | Enter instructor 11's name
Enter Instructor Name | Enter instructor 12's name
Enter Instructor Name | Enter instructor 13's name
Enter Instructor Name | Enter instructor 14's name
Enter Instructor Name | Enter instructor 15's name

Courses

Enter Course | Enter course 1
Enter Course | Enter course 2


localhost:8080/timetablegen/create.jsp

Courses

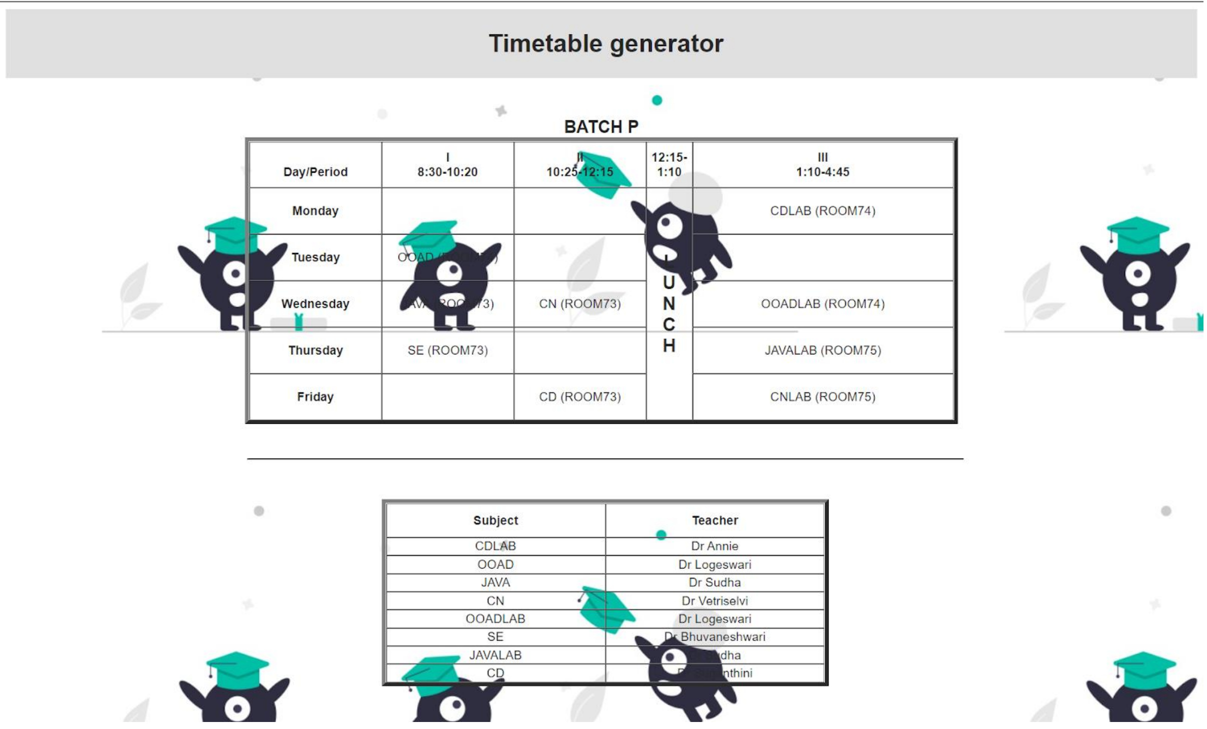
Enter Course | Enter course 1
Enter Course | Enter course 2
Enter Course | Enter course 3
Enter Course | Enter course 4
Enter Course | Enter course 5
Enter Course | Enter course 6
Enter Course | Enter course 7
Enter Course | Enter course 8
Enter Course | Enter course 9

Batches

Enter Batch | batch 1
Enter Batch | batch 2
Enter Batch | batch 3

Submit

Time Tables Generated:

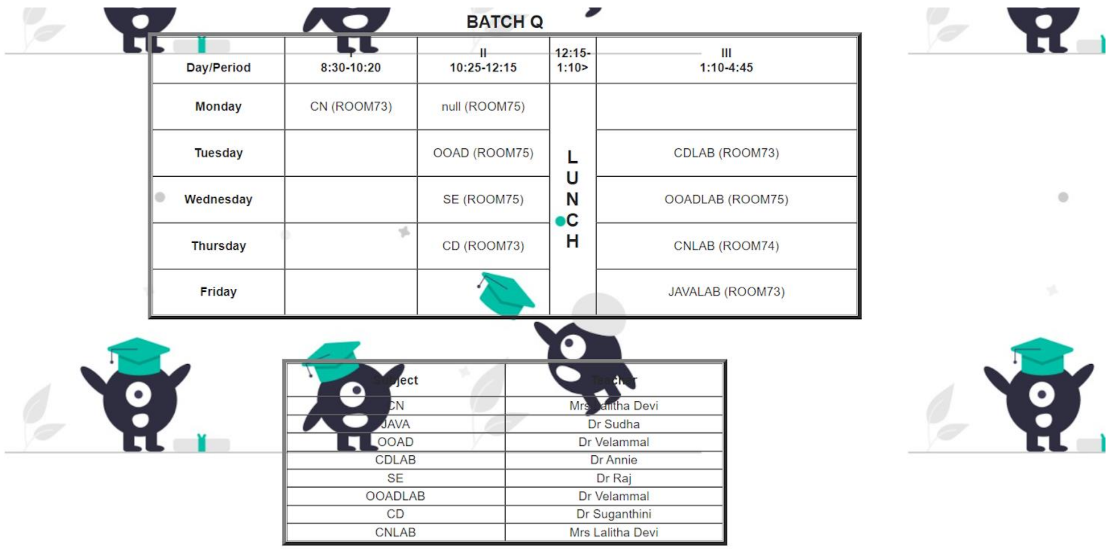


Timetable generator

BATCH P

Day/Period	I 8:30-10:20	II 10:25-12:15	12:15-1:10	III 1:10-4:45
Monday			LUNCH	CDLAB (ROOM74)
Tuesday	OOAD (ROOM73)			
Wednesday	JAVA (ROOM73)	CN (ROOM73)		OOADLAB (ROOM74)
Thursday	SE (ROOM73)			JAVALAB (ROOM75)
Friday		CD (ROOM73)		CNLAB (ROOM75)

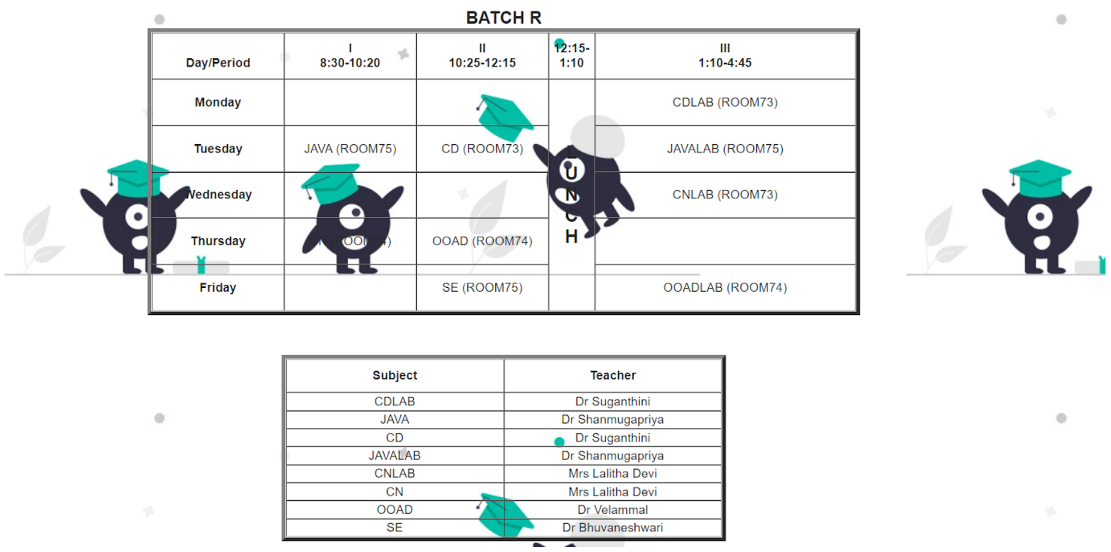
Subject	Teacher
CDLAB	Dr Annie
OOAD	Dr Logeswari
JAVA	Dr Sudha
CN	Dr Vetriselvi
OOADLAB	Dr Logeswari
SE	Dr Bhuvaneshwari
JAVALAB	Dr Sudha
CD	Dr Suganthini



BATCH Q

Day/Period	I 8:30-10:20	II 10:25-12:15	12:15-1:10>	III 1:10-4:45
Monday	CN (ROOM73)	null (ROOM75)	LUNCH	
Tuesday		OOAD (ROOM75)		CDLAB (ROOM73)
Wednesday		SE (ROOM75)		OOADLAB (ROOM75)
Thursday		CD (ROOM73)		CNLAB (ROOM74)
Friday				JAVALAB (ROOM73)

Subject	Teacher
CN	Mrs Lalitha Devi
JAVA	Dr Sudha
OOAD	Dr Velammal
CDLAB	Dr Annie
SE	Dr Raj
OOADLAB	Dr Velammal
CD	Dr Suganthini
CNLAB	Mrs Lalitha Devi

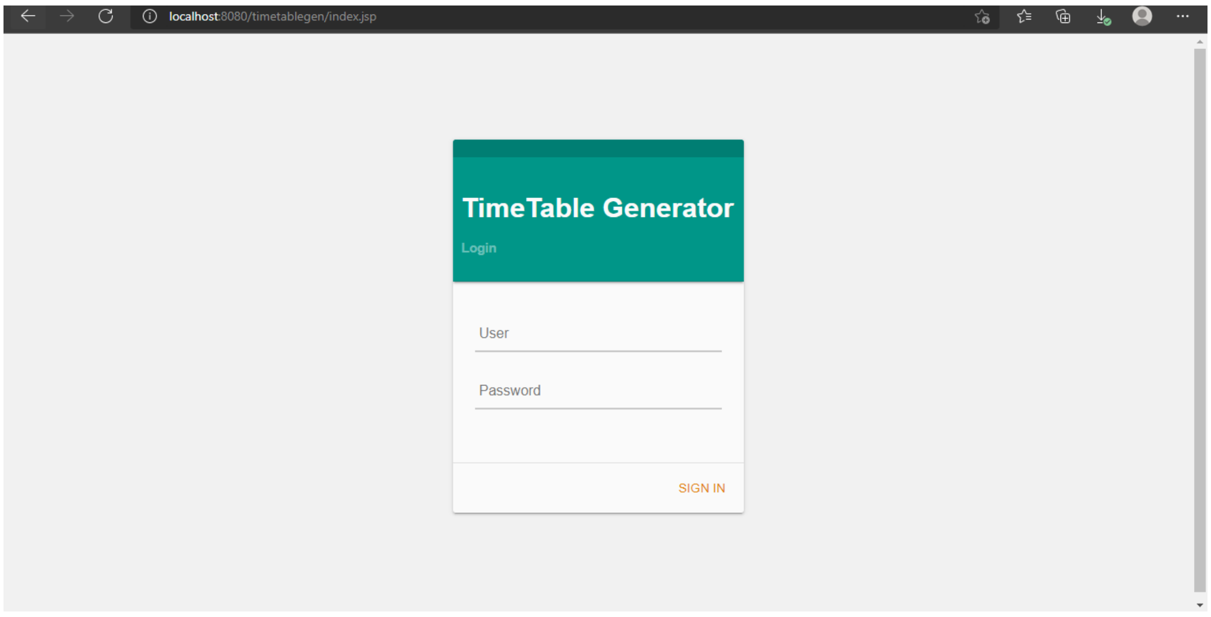


BATCH R

Day/Period	I 8:30-10:20	II 10:25-12:15	12:15- 1:10	III 1:10-4:45
Monday			LUNCH	CDLAB (ROOM73)
Tuesday	JAVA (ROOM75)	CD (ROOM73)		JAVALAB (ROOM75)
Wednesday				CNLAB (ROOM73)
Thursday	CN (ROOM74)	OOAD (ROOM74)		
Friday		SE (ROOM75)		OOADLAB (ROOM74)

Subject	Teacher
CDLAB	Dr Suganthini
JAVA	Dr Shanmugapriya
CD	Dr Suganthini
JAVALAB	Dr Shanmugapriya
CNLAB	Mrs Lalitha Devi
CN	Mrs Lalitha Devi
OOAD	Dr Velammal
SE	Dr Bhuvaneshwari

Login Module:



Conclusion

Thus, using the genetic algorithm and java framework we are able to fully automate the timetable generation process. We have been able to satisfy the constraints required:

• There have not be any single instance of a faculty taking two classes simultaneously
• A class group does not have more than one lectures at the same time
• The minimum number of hours that is required by a course per week has be fulfilled

Further work

Although this program helps generate the timetable for multiple batches, departments, and years, there is a lot more that can be done to make the project even better in terms of soft-constraints like

- Professors giving preferences to certain classes
- Professors having fixed number of hours to teach
- Assigning priorities for different members of the teach faculty

References

- https://in.mathworks.com/help/gads/how-the-genetic-algorithm-works.html
- https://www.codeproject.com/Articles/23111/Making-a-Class-Schedule-Using-a-Genetic-Algorithm#:~:text=The%20genetic%20algorithm%20is%20fairly,on%20the%20pair%20of%20parents.
- https://egrove.olemiss.edu/cgi/viewcontent.cgi?article=1442&context=etd
- https://d1wqtxts1xzle7.cloudfront.net/33105221/download-with-cover-page-v2.pdf?Expires=1642150584&Signature=VvWlI1hwJwl~BuZ-eyOc6i7sWJPZ0kADJUV-mZCJxZP6WZWFQVFO1D~zQgLNjs0apnXUCAF4dAeFzFYFbHJf5itdsrQj6hG3FHq9yZJgCFlgFnWJW554Dz3wwNDUSBAe20~VmwT8uYdd5j07bvmkZGihA~KmHzCM~FnMdwOPWvOkBiqKVqtsSqnM4plsijJTFRqHKayIQOd-P9iQyQXUO~YGSVAl4U17xAgSkfN3SZYe3M6wFMQtPX9s1S3e94BHooiXG5Kht8lExlF0UAd8Sk1E7p9Zq6WnJTa4gvIuAOuXBxWDmldpKoXJhY8MhpjfA8HVDcrc4lihzT483pRChQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA