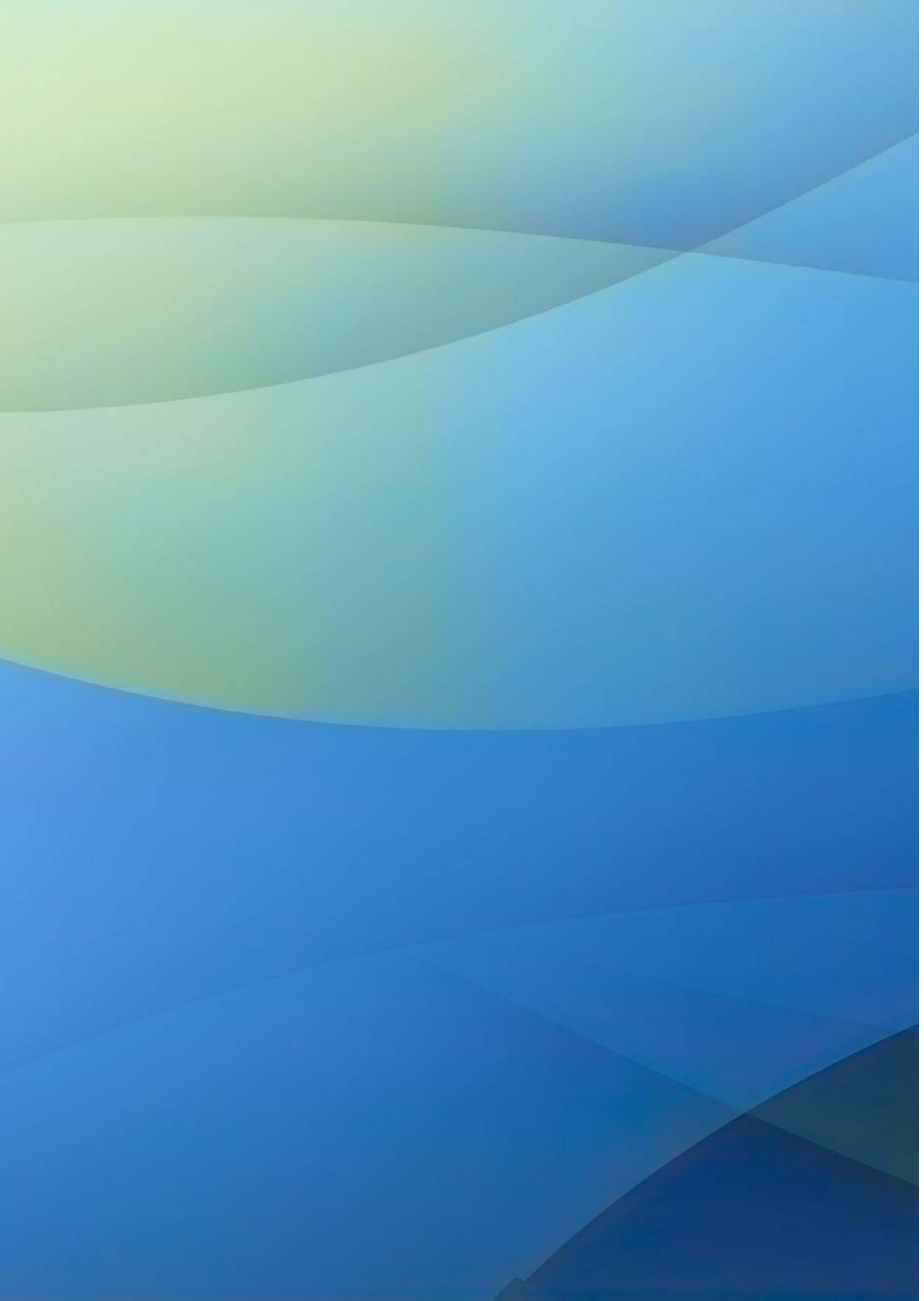


Timetable Management System

PROJECT BY:

- A KASTURI (59)
- A THANUJA (55)
- JUSTIN CLARKE (56)
- G ADITYA SHARMA (53)

PROJECT DESCRIPTION:

- The main theme of this project is to generate the Timetable by taking the faculty list, Department list, Subjects list and Time schedule as input and generates the Timetable by satisfying all the constraints.
- The Admin was given with special privileges to add/modify the faculty details of the Department, Subject details, Period Scheduling. Add/View/Delete a Timetable. The Admin can login to the website and make changes according to his wish.
- Similarly the faculty does have their own credentials to enter into the site and watch the class wise Timetable.

REQUIREMENT ANALYSIS:

System User Management

User management describes the ability for administrators to manage user access to various IT resources like systems, devices, applications, storage systems, network, and more. User management is a core part to any directory service and is a basic security essential for any organization.

Login Management

User management describes the ability for administrators to manage user access to various IT resources like systems, devices, applications. User management is a core part to any directory service and is a basic security essential for any organization. User management enables admins to control user access and on-board and off-board users to and from IT resources.

Subject Management

Subjects are the category names that are associated to specific learning objects to help users find learning. These subject names can be used by users when searching for teaching. When a user filters teaching by subject, only the learning associated with the subject displays.

Room Management

In this category we allot classrooms to each section in a Department.

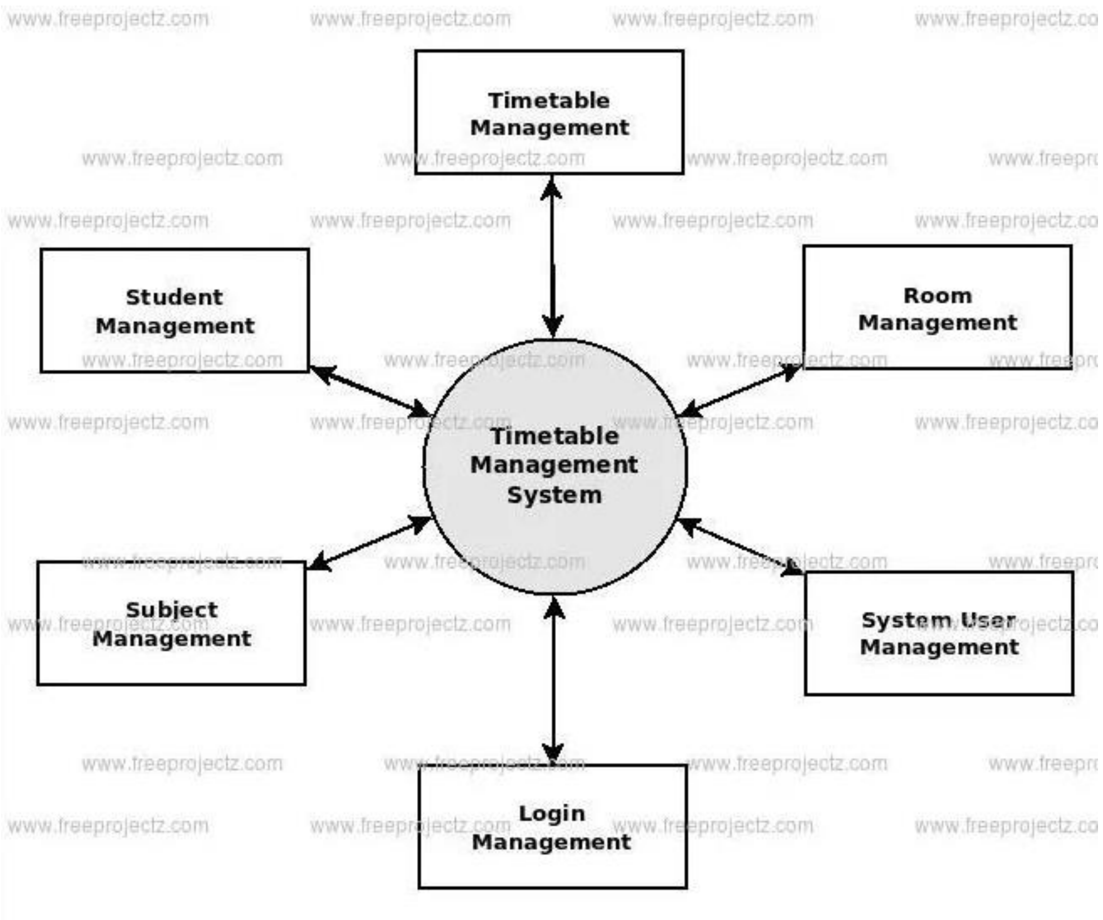
Faculty Management

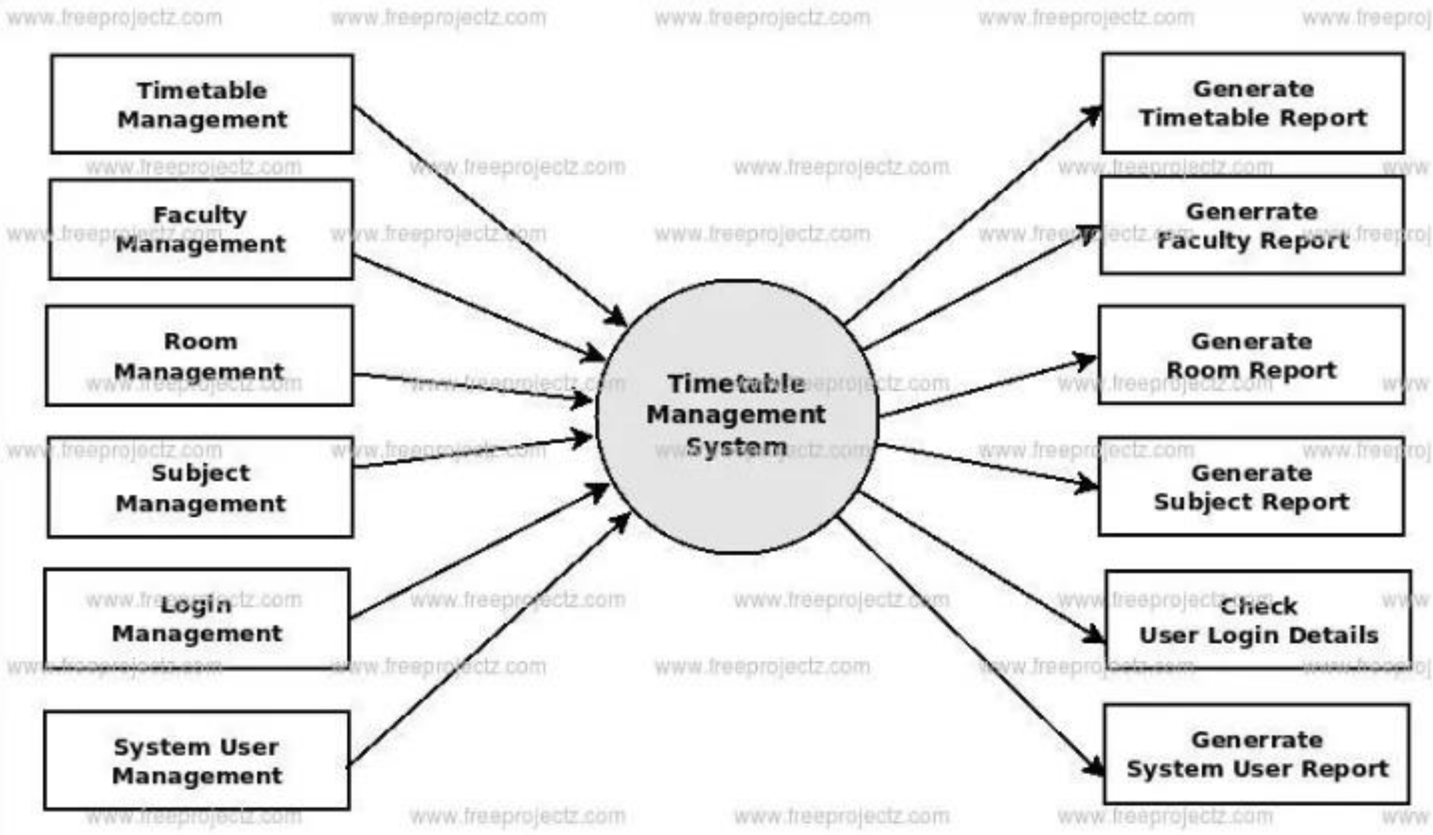
Faculty Management software allows you to maintain comprehensive faculty management information, including qualifications and skill sets, contract documents, define triggers to remind faculty and administrators of contract expiry dates and more, in the Faculty Management system.

Timetable Management

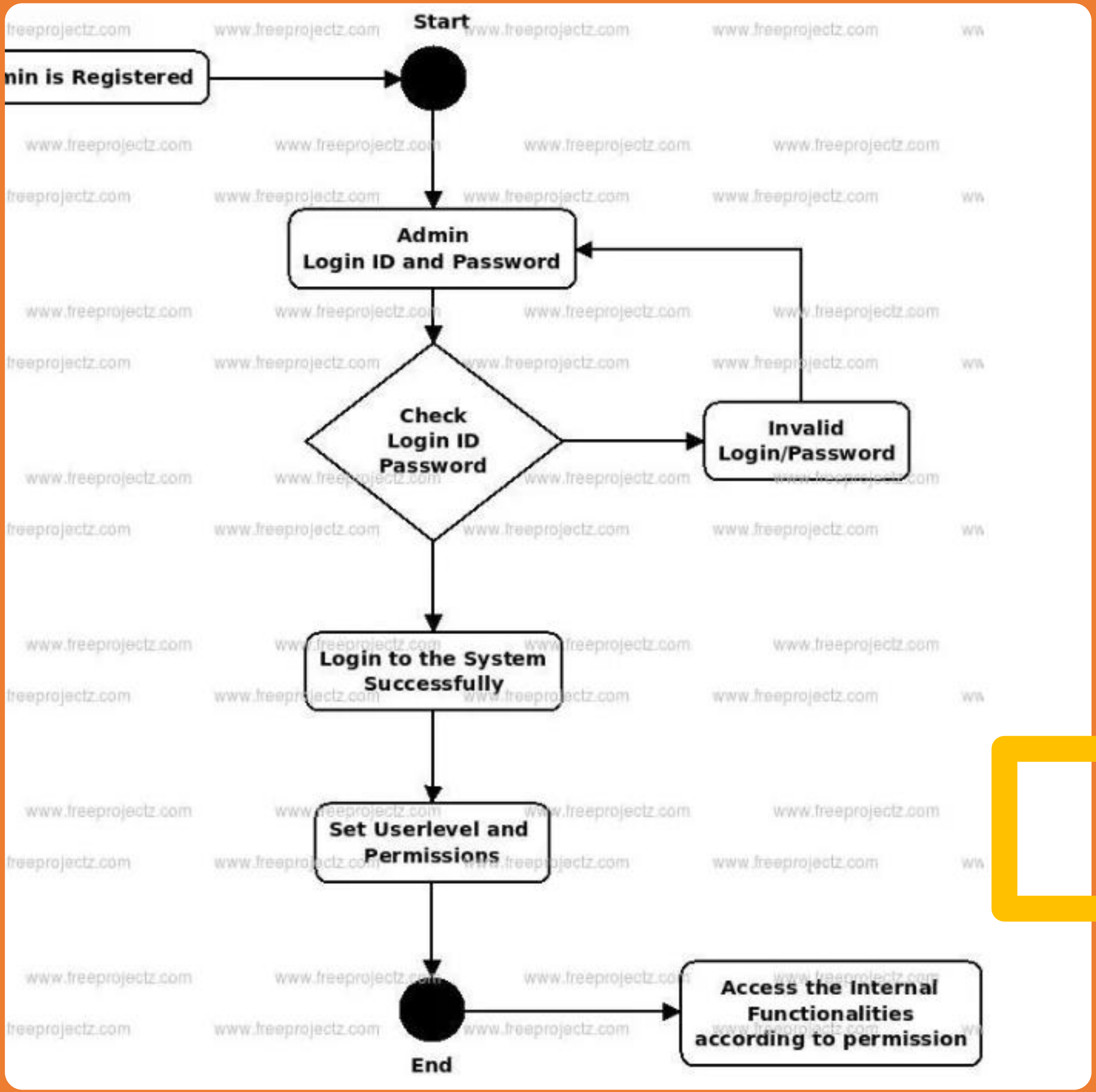
Timetable Management System contains a database, which stores the staff and students' personal details. The College Timetable system is a very useful system for Students, Faculties & Administrator through which students/parents can get the information about them/their schedule.

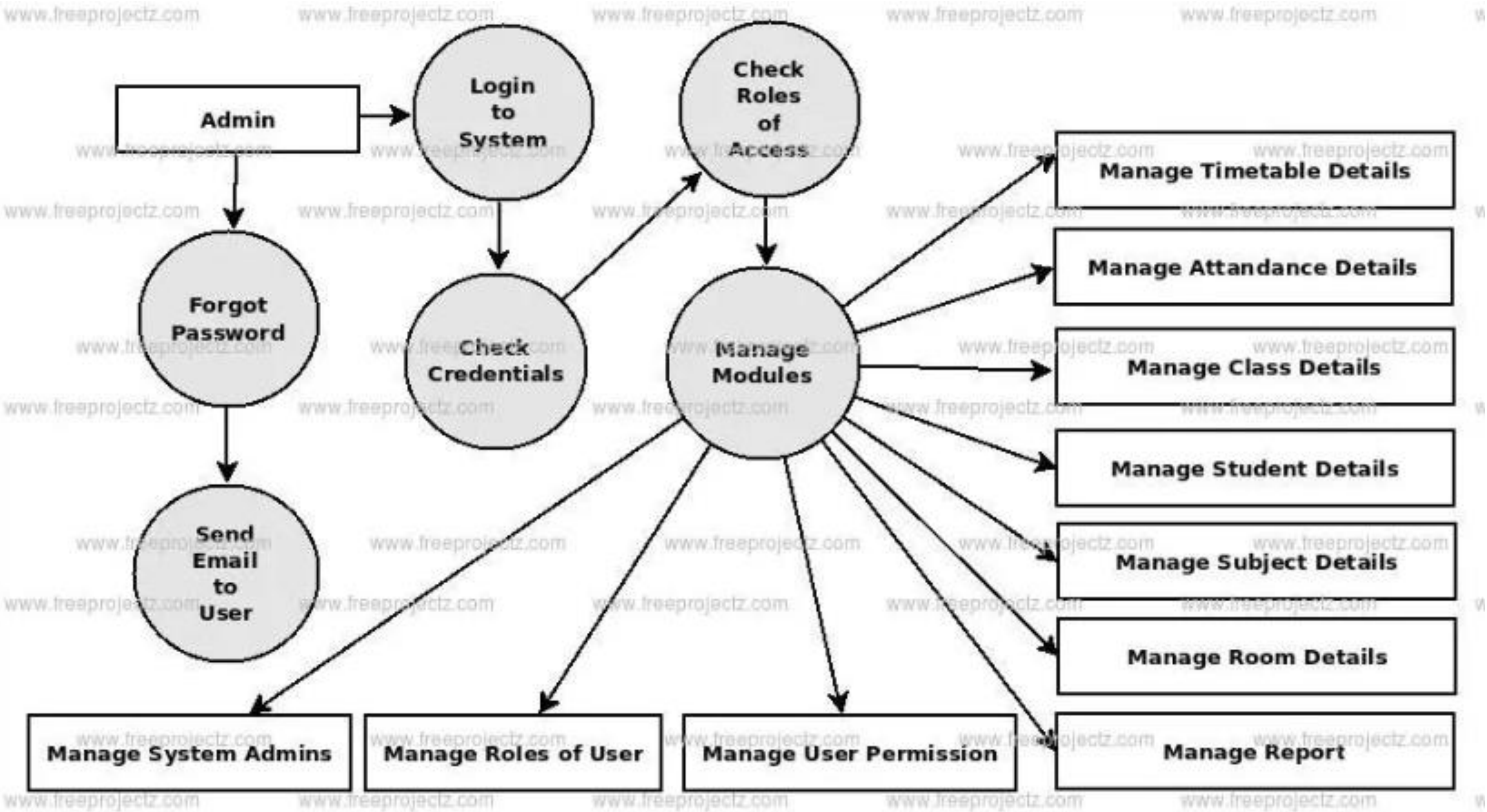
DESIGN ANALYSIS:



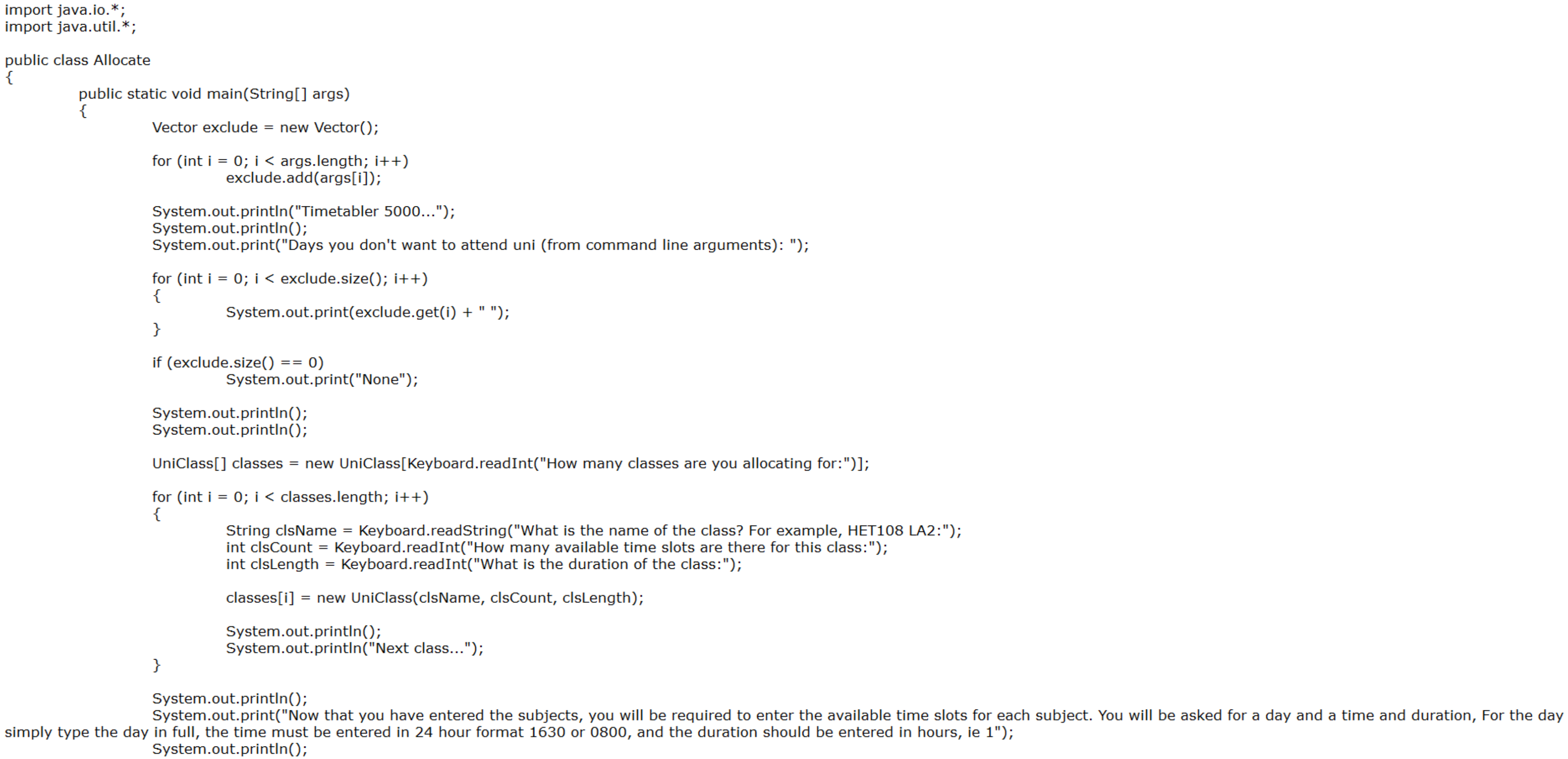

Timetable Management
Faculty Management
Room Management
Subject Management
Login Management
System User Management
Timetable Management System
Generate Timetable Report
Generrate Faculty Report
Generate Room Report
Generate Subject Report
Check User Login Details
Generrate System User Report

Flow chart:


Start
min is Registered
Admin
Login ID and Password
Check
Login ID
Password
Invalid
Login/Password
Login to the System
Successfully
Set Userlevel and
Permissions
End
Access the Internal
Functionalities
according to permission

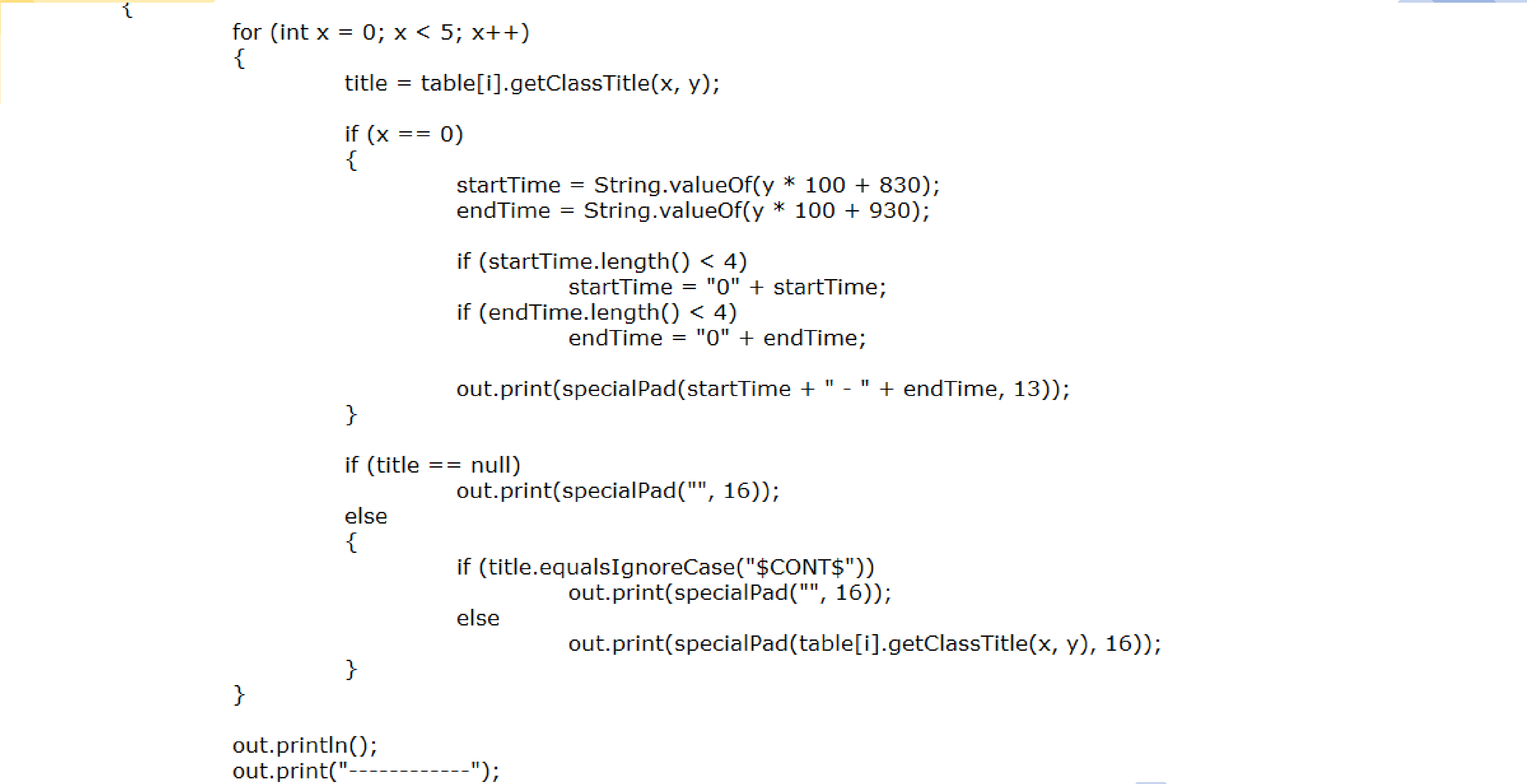

Admin
Login to System
Check Credentials
Check Roles of Access
Forgot Password
Send Email to User
Manage Modules
Manage Timetable Details
Manage Attandance Details
Manage Class Details
Manage Student Details
Manage Subject Details
Manage Room Details
Manage Report
Manage System Admins
Manage Roles of User
Manage User Permission

IMPLEMENTATION OF CODE:

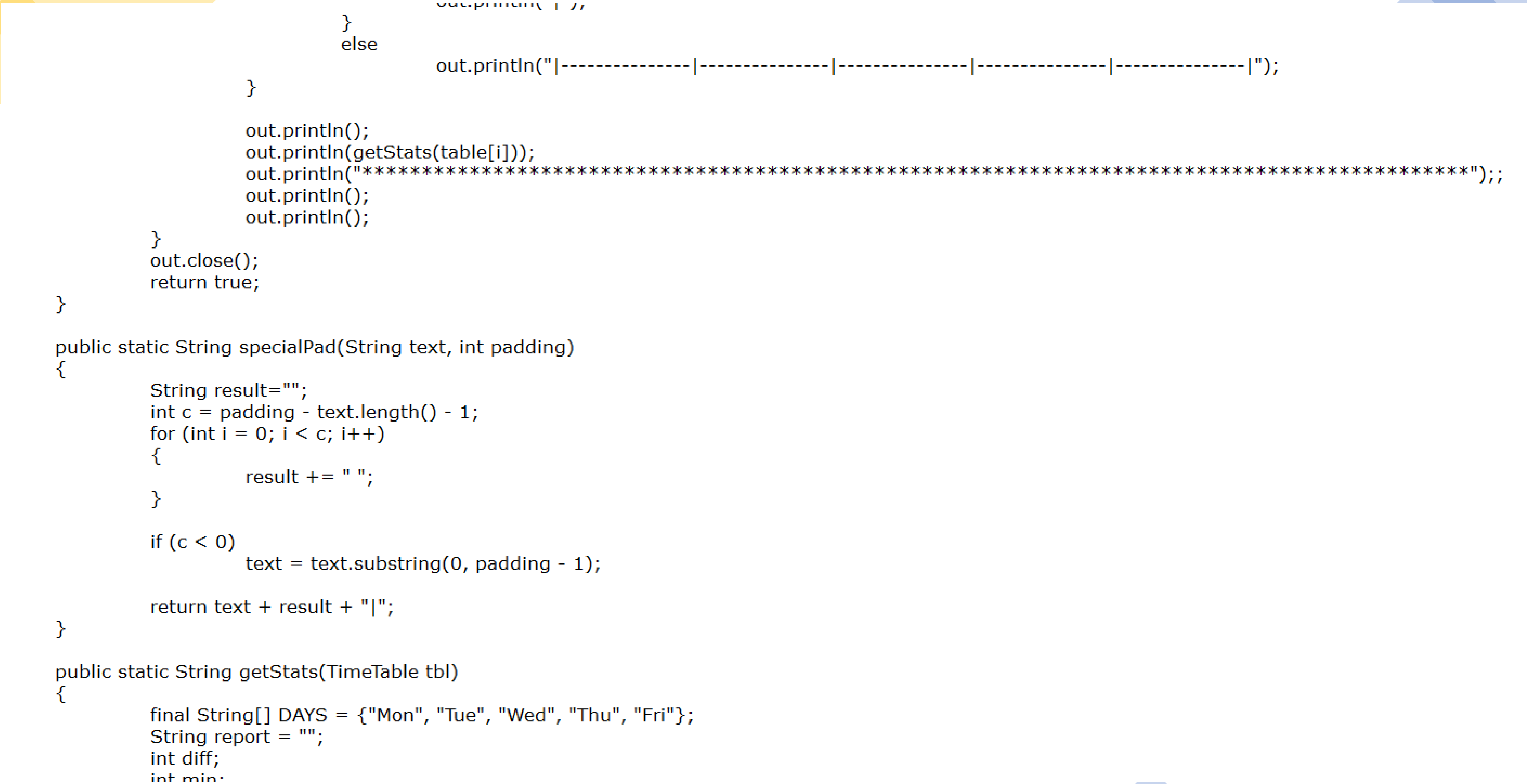


```
import java.io.*;
import java.util.*;

public class Allocate
{
        public static void main(String[] args)
        {
                Vector exclude = new Vector();

                for (int i = 0; i < args.length; i++)
                        exclude.add(args[i]);

                System.out.println("Timetabler 5000...");
                System.out.println();
                System.out.print("Days you don't want to attend uni (from command line arguments): ");

                for (int i = 0; i < exclude.size(); i++)
                {
                        System.out.print(exclude.get(i) + " ");
                }

                if (exclude.size() == 0)
                        System.out.print("None");

                System.out.println();
                System.out.println();

                UniClass[] classes = new UniClass[Keyboard.readInt("How many classes are you allocating for:")];

                for (int i = 0; i < classes.length; i++)
                {
                        String clsName = Keyboard.readString("What is the name of the class? For example, HET108 LA2:");
                        int clsCount = Keyboard.readInt("How many available time slots are there for this class:");
                        int clsLength = Keyboard.readInt("What is the duration of the class:");

                        classes[i] = new UniClass(clsName, clsCount, clsLength);

                        System.out.println();
                        System.out.println("Next class...");
                }

                System.out.println();
                System.out.print("Now that you have entered the subjects, you will be required to enter the available time slots for each subject. You will be asked for a day and a time and duration, For the day
simply type the day in full, the time must be entered in 24 hour format 1630 or 0800, and the duration should be entered in hours, ie 1");
                System.out.println();
        for (int i = 0; i < classes.length; i++)
        {
                String day;
                String time;
                int duration;
                int slot;

                System.out.println("For " + classes[i].getName());

                for (int j = 0; j < classes[i].slotsCount(); j++)
                {
                        System.out.println();
                        System.out.println("Class: " + (j+1));
                        day = Keyboard.readString("Day:");
                        time = Keyboard.readString("Time:");
                        classes[i].addTime(day, time);
                }

                System.out.println();
        }
        int trials = Keyboard.readInt("Allocate will attempt to randomly generate non-clashing timetables. How many would you like to generate:");
        System.out.println();

        try{
                makeTable(classes, exclude, "08:30", "23:30", trials, "timetables_manual.txt");
        }
        catch (IOException exp)
        {
                System.out.println(exp);
        }
}

public static boolean makeTable(UniClass[] classes, Vector exclude, String min_start, String max_finish, int n, String theFile) throws IOException
{
        TimeTable[] table = new TimeTable[n];
        int k = 0;
        int slot_tries;
        int timetable_tries = 0;
        int tmp;
        int min;
        int max;
        boolean allocated = false;
        boolean abort = false;
        int slot;

        tmp = Integer.valueOf(min_start.replaceAll(":", "")); //remove ":" from string
        min = (tmp - 830) / 100;
```

```
tmp = Integer.valueOf(max_finish.replaceAll(":", "")); //remove ":" from string
max = (tmp - 830) / 100;

do
{
    table[k] = new TimeTable();

    for (int i = 0; i < classes.length; i++)
    {
        slot_tries = 0;
        do
        {
            slot = (int)Math.floor(Math.random() * classes[i].slotsCount());
            allocated = table[k].addClass(classes[i], slot, exclude, min, max);
            slot_tries++;
        }
        while ((slot_tries < Math.pow(classes[i].slotsCount(), 3)) && !allocated);

        if (allocated == false)
        {
            timetable_tries++;
            if (timetable_tries > 1000)
                abort = true;

            break;
        }
    }
    if (allocated == true)
    {
        timetable_tries = 0; //timetable generated, move onto next timetable
        k++;
    }
}
while (k < n && !abort);

if (abort)
{
    return false; // timetable could not be generated
}

String title;
String startTime = "";
String endTime = "";
PrintWriter out = new PrintWriter(new FileWriter(theFile));
```

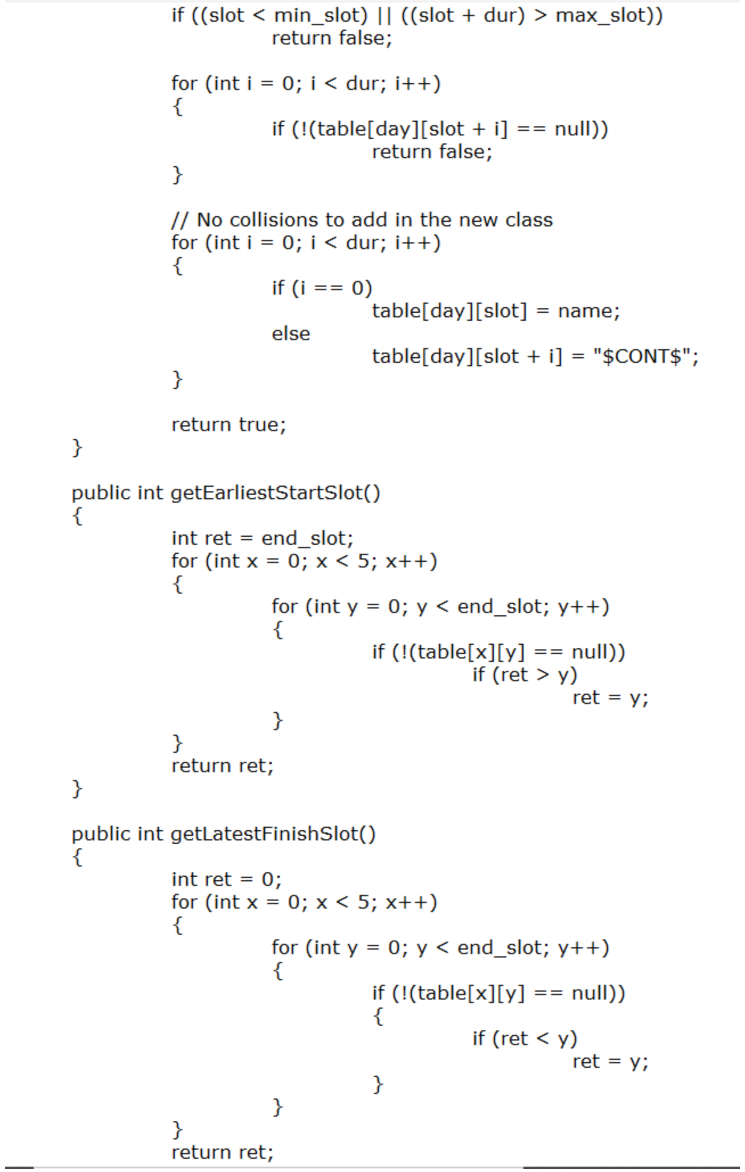


```
{
    for (int x = 0; x < 5; x++)
    {
        title = table[i].getClassTitle(x, y);

        if (x == 0)
        {
            startTime = String.valueOf(y * 100 + 830);
            endTime = String.valueOf(y * 100 + 930);

            if (startTime.length() < 4)
                startTime = "0" + startTime;
            if (endTime.length() < 4)
                endTime = "0" + endTime;

            out.print(specialPad(startTime + " - " + endTime, 13));
        }

        if (title == null)
            out.print(specialPad("", 16));
        else
        {
            if (title.equalsIgnoreCase("$CONT$"))
                out.print(specialPad("", 16));
            else
                out.print(specialPad(table[i].getClassTitle(x, y), 16));
        }
    }

    out.println();
    out.print("------------");
```

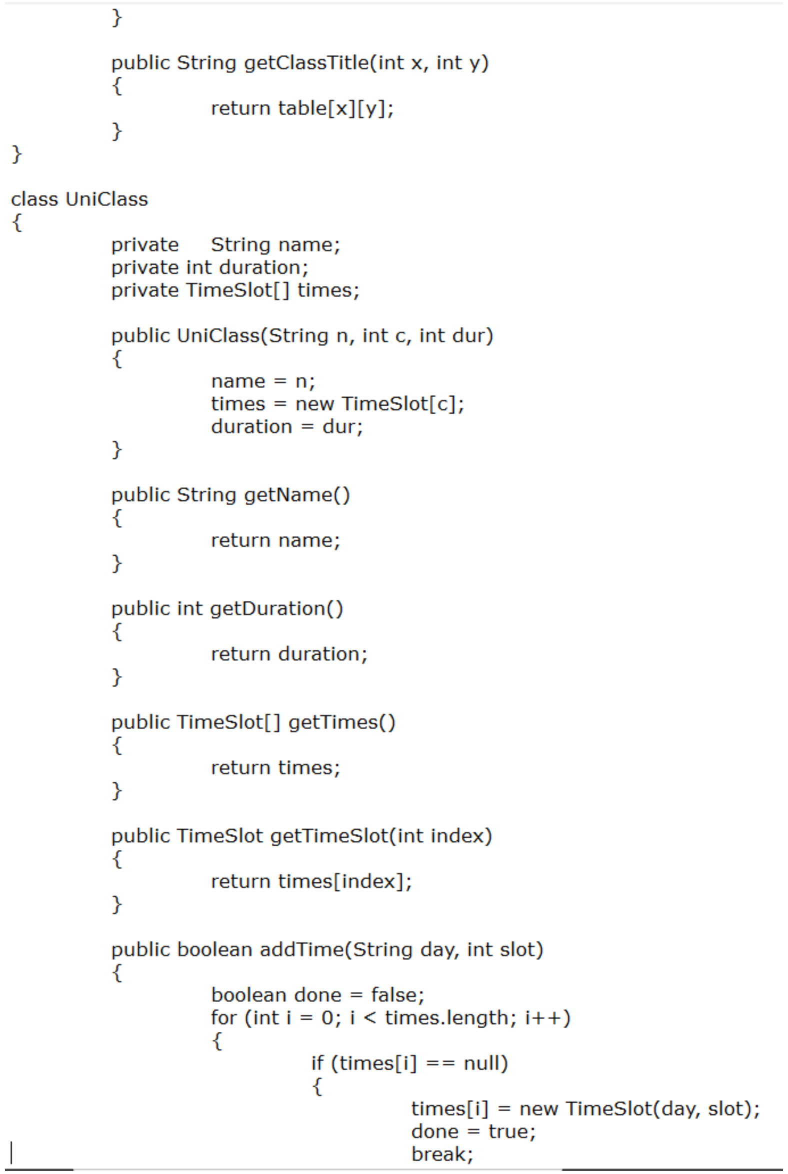


```
                                        out.println("|");
                              }
                              else
                                        out.println("|---------------|---------------|---------------|---------------|---------------|");
                    }

                    out.println();
                    out.println(getStats(table[i]));
                    out.println("***************************************************************************************************************");;
                    out.println();
                    out.println();
          }
          out.close();
          return true;
}

public static String specialPad(String text, int padding)
{
          String result="";
          int c = padding - text.length() - 1;
          for (int i = 0; i < c; i++)
          {
                    result += " ";
          }

          if (c < 0)
                    text = text.substring(0, padding - 1);

          return text + result + "|";
}

public static String getStats(TimeTable tbl)
{
          final String[] DAYS = {"Mon", "Tue", "Wed", "Thu", "Fri"};
          String report = "";
          int diff;
          int min;
```

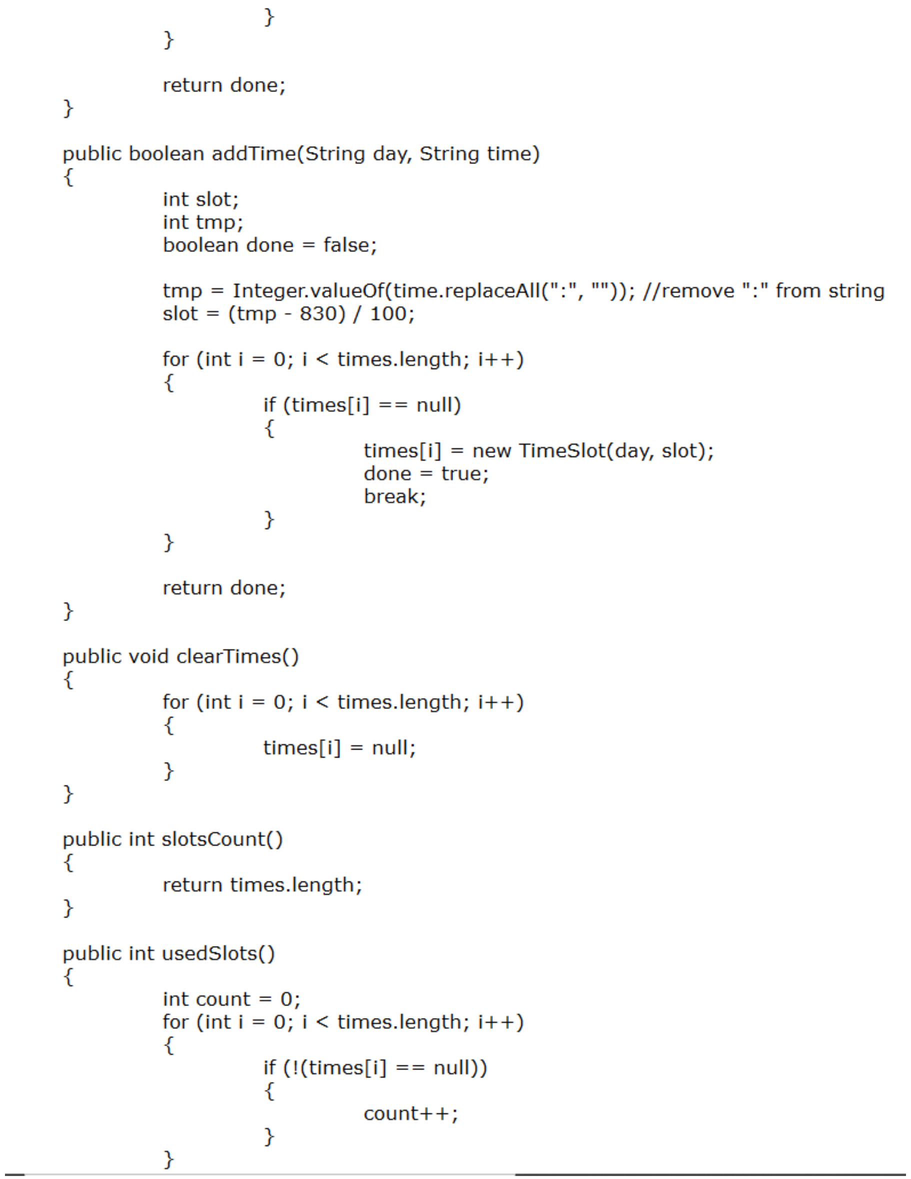
```
        return report;
    }
}

class TimeTable
{
    public final static int end_slot = 15;
    private String[][] table = new String[5][end_slot];

    public TimeTable(){}

    public boolean addClass(UniClass cls, int index, Vector excl, int min_slot, int max_slot)
    {
        String name = cls.getName();
        String dayName =  (cls.getTimeSlot(index).getDayName()).substring(0, 3);
        int day = cls.getTimeSlot(index).getDay();
        int slot = cls.getTimeSlot(index).getSlot();
        int dur = cls.getDuration();

        for (int a = 0; a < excl.size(); a++)
        {
            if (((String)excl.get(a)).equalsIgnoreCase(dayName))
            {
```



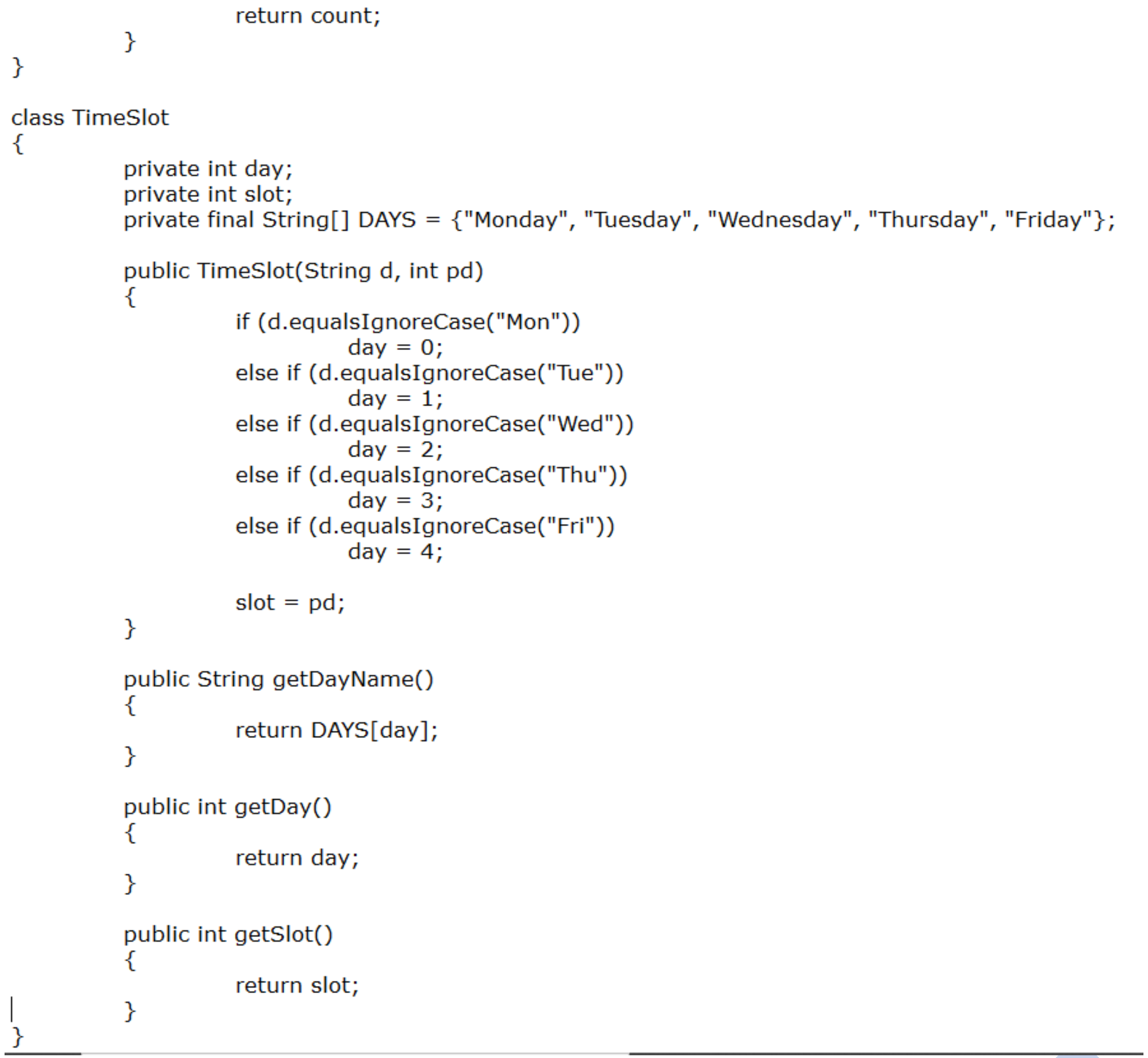
```
        if ((slot < min_slot) || ((slot + dur) > max_slot))
            return false;

        for (int i = 0; i < dur; i++)
        {
            if (!(table[day][slot + i] == null))
                return false;
        }

        // No collisions to add in the new class
        for (int i = 0; i < dur; i++)
        {
            if (i == 0)
                table[day][slot] = name;
            else
                table[day][slot + i] = "$CONT$";
        }

        return true;
    }

    public int getEarliestStartSlot()
    {
        int ret = end_slot;
        for (int x = 0; x < 5; x++)
        {
            for (int y = 0; y < end_slot; y++)
            {
                if (!(table[x][y] == null))
                    if (ret > y)
                        ret = y;
            }
        }
        return ret;
    }

    public int getLatestFinishSlot()
    {
        int ret = 0;
        for (int x = 0; x < 5; x++)
        {
            for (int y = 0; y < end_slot; y++)
            {
                if (!(table[x][y] == null))
                {
                    if (ret < y)
                        ret = y;
                }
            }
        }
        return ret;
```



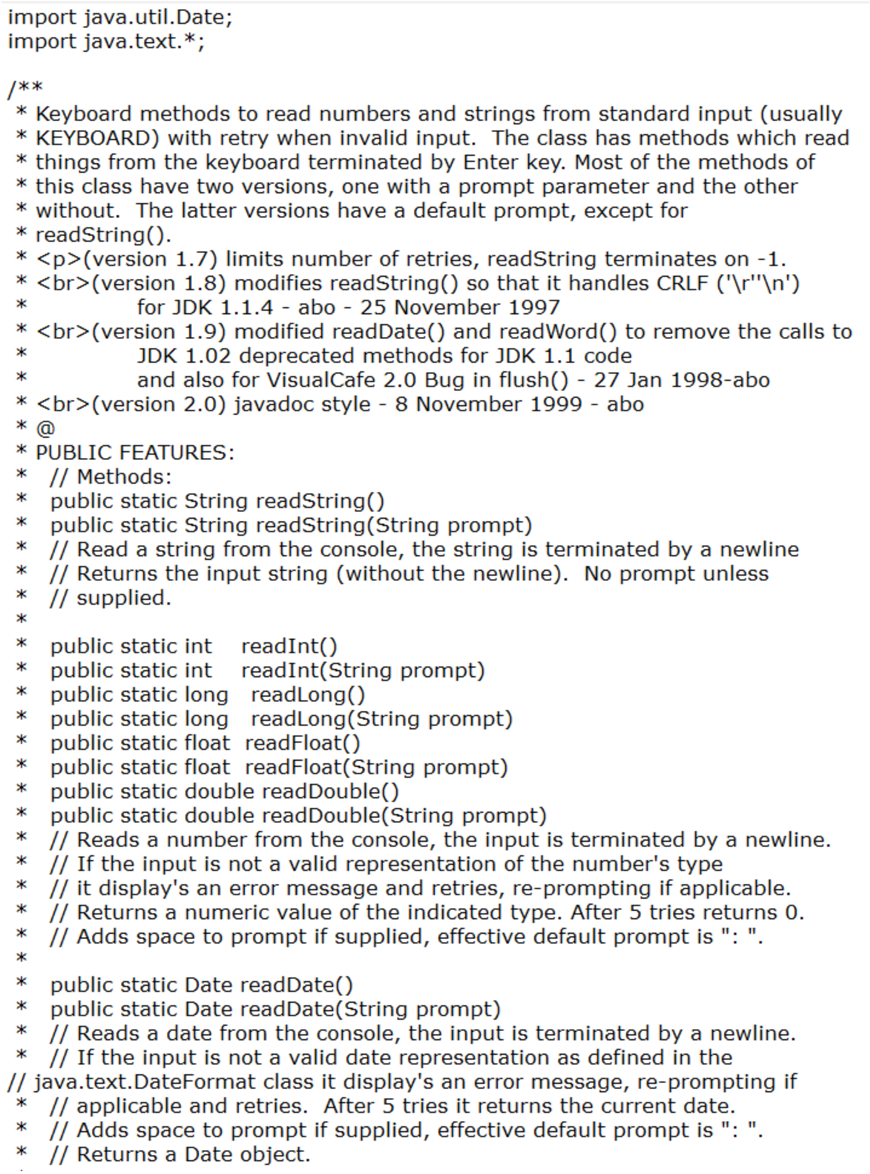
```
        }

        public String getClassTitle(int x, int y)
        {
                return table[x][y];
        }
}

class UniClass
{
        private     String name;
        private int duration;
        private TimeSlot[] times;

        public UniClass(String n, int c, int dur)
        {
                name = n;
                times = new TimeSlot[c];
                duration = dur;
        }

        public String getName()
        {
                return name;
        }

        public int getDuration()
        {
                return duration;
        }

        public TimeSlot[] getTimes()
        {
                return times;
        }

        public TimeSlot getTimeSlot(int index)
        {
                return times[index];
        }

        public boolean addTime(String day, int slot)
        {
                boolean done = false;
                for (int i = 0; i < times.length; i++)
                {
                        if (times[i] == null)
                        {
                                times[i] = new TimeSlot(day, slot);
                                done = true;
                                break;
```



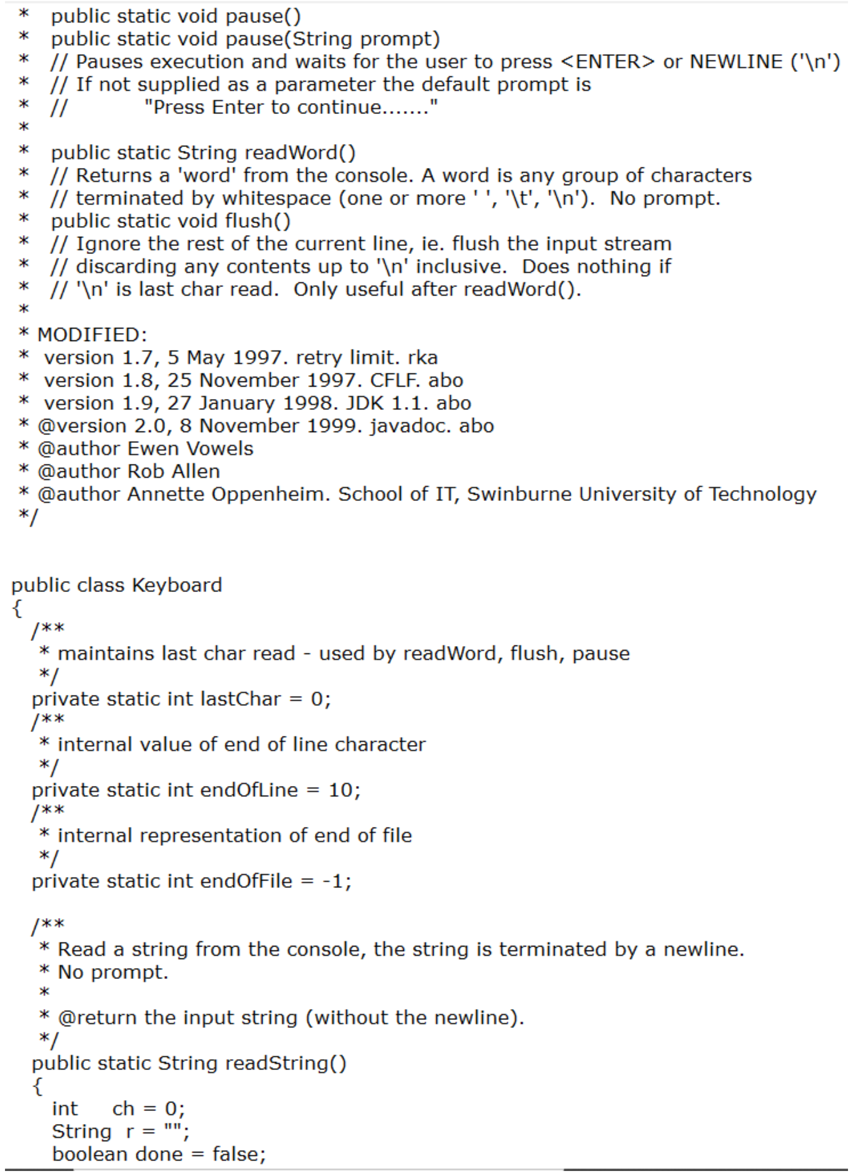
```
        }
    }

    return done;
}

public boolean addTime(String day, String time)
{
    int slot;
    int tmp;
    boolean done = false;

    tmp = Integer.valueOf(time.replaceAll(":", "")); //remove ":" from string
    slot = (tmp - 830) / 100;

    for (int i = 0; i < times.length; i++)
    {
        if (times[i] == null)
        {
            times[i] = new TimeSlot(day, slot);
            done = true;
            break;
        }
    }

    return done;
}

public void clearTimes()
{
    for (int i = 0; i < times.length; i++)
    {
        times[i] = null;
    }
}

public int slotsCount()
{
    return times.length;
}

public int usedSlots()
{
    int count = 0;
    for (int i = 0; i < times.length; i++)
    {
        if (!(times[i] == null))
        {
            count++;
        }
    }
```



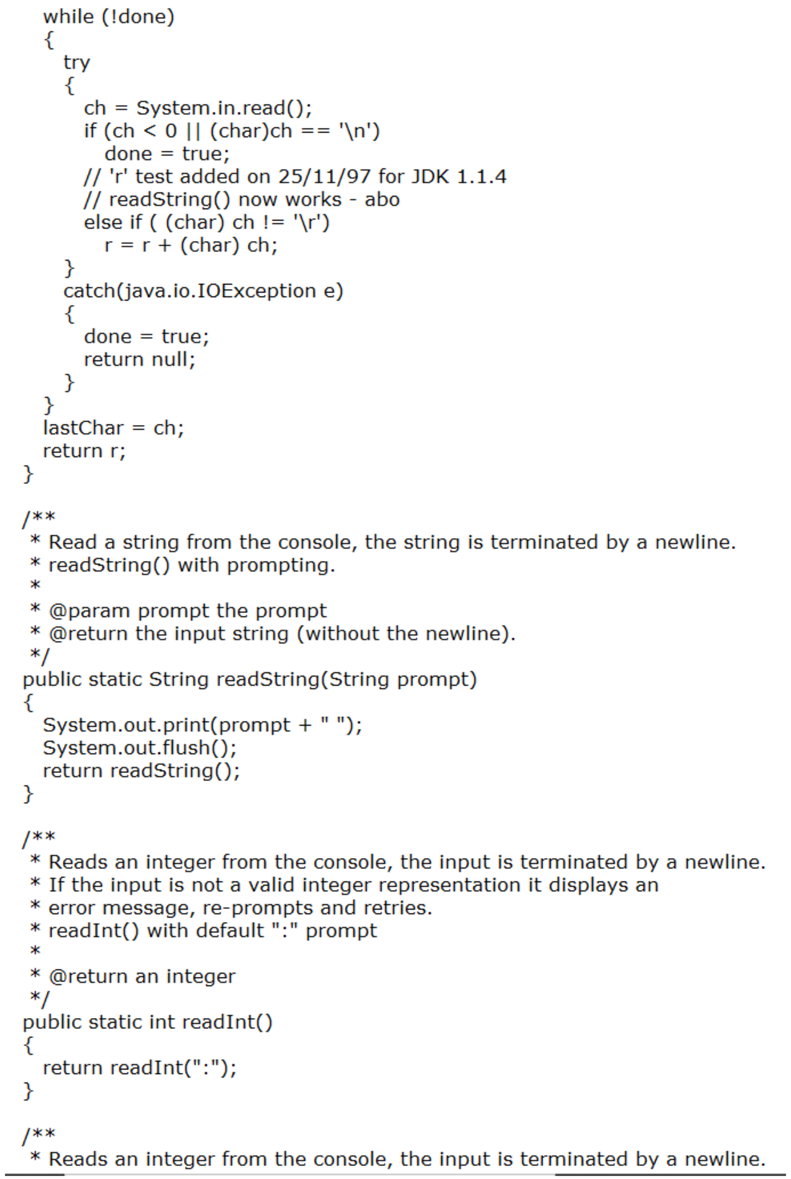
```
        return count;
    }
}

class TimeSlot
{
    private int day;
    private int slot;
    private final String[] DAYS = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday"};

    public TimeSlot(String d, int pd)
    {
        if (d.equalsIgnoreCase("Mon"))
            day = 0;
        else if (d.equalsIgnoreCase("Tue"))
            day = 1;
        else if (d.equalsIgnoreCase("Wed"))
            day = 2;
        else if (d.equalsIgnoreCase("Thu"))
            day = 3;
        else if (d.equalsIgnoreCase("Fri"))
            day = 4;

        slot = pd;
    }

    public String getDayName()
    {
        return DAYS[day];
    }

    public int getDay()
    {
        return day;
    }

    public int getSlot()
    {
        return slot;
    }
}
```



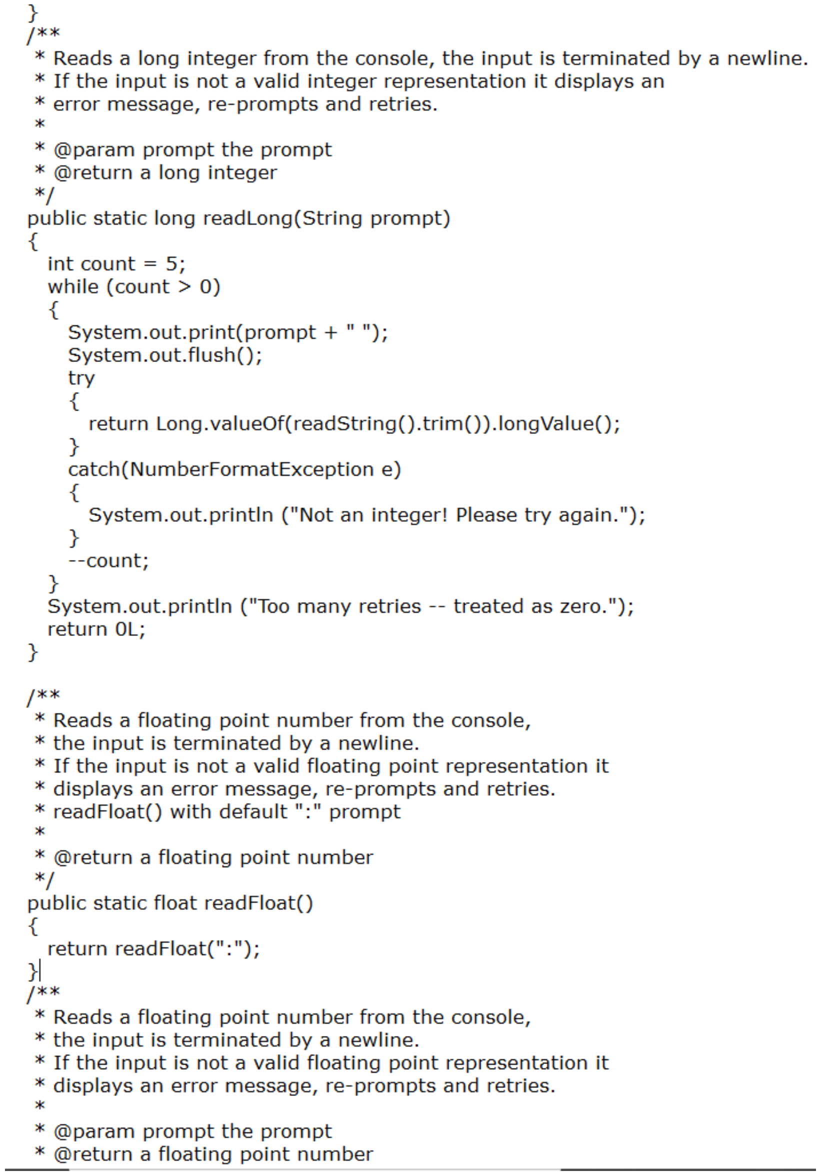
```
import java.util.Date;
import java.text.*;

/**
 * Keyboard methods to read numbers and strings from standard input (usually
 * KEYBOARD) with retry when invalid input.  The class has methods which read
 * things from the keyboard terminated by Enter key. Most of the methods of
 * this class have two versions, one with a prompt parameter and the other
 * without.  The latter versions have a default prompt, except for
 * readString().
 * <p>(version 1.7) limits number of retries, readString terminates on -1.
 * <br>(version 1.8) modifies readString() so that it handles CRLF ('\r''\n')
 *             for JDK 1.1.4 - abo - 25 November 1997
 * <br>(version 1.9) modified readDate() and readWord() to remove the calls to
 *             JDK 1.02 deprecated methods for JDK 1.1 code
 *             and also for VisualCafe 2.0 Bug in flush() - 27 Jan 1998-abo
 * <br>(version 2.0) javadoc style - 8 November 1999 - abo
 * @
 * PUBLIC FEATURES:
 *  // Methods:
 *  public static String readString()
 *  public static String readString(String prompt)
 *  // Read a string from the console, the string is terminated by a newline
 *  // Returns the input string (without the newline).  No prompt unless
 *  // supplied.
 *
 *  public static int    readInt()
 *  public static int    readInt(String prompt)
 *  public static long   readLong()
 *  public static long   readLong(String prompt)
 *  public static float  readFloat()
 *  public static float  readFloat(String prompt)
 *  public static double readDouble()
 *  public static double readDouble(String prompt)
 *  // Reads a number from the console, the input is terminated by a newline.
 *  // If the input is not a valid representation of the number's type
 *  // it display's an error message and retries, re-prompting if applicable.
 *  // Returns a numeric value of the indicated type. After 5 tries returns 0.
 *  // Adds space to prompt if supplied, effective default prompt is ": ".
 *
 *  public static Date readDate()
 *  public static Date readDate(String prompt)
 *  // Reads a date from the console, the input is terminated by a newline.
 *  // If the input is not a valid date representation as defined in the
// java.text.DateFormat class it display's an error message, re-prompting if
 *  // applicable and retries.  After 5 tries it returns the current date.
 *  // Adds space to prompt if supplied, effective default prompt is ": ".
 *  // Returns a Date object.
```



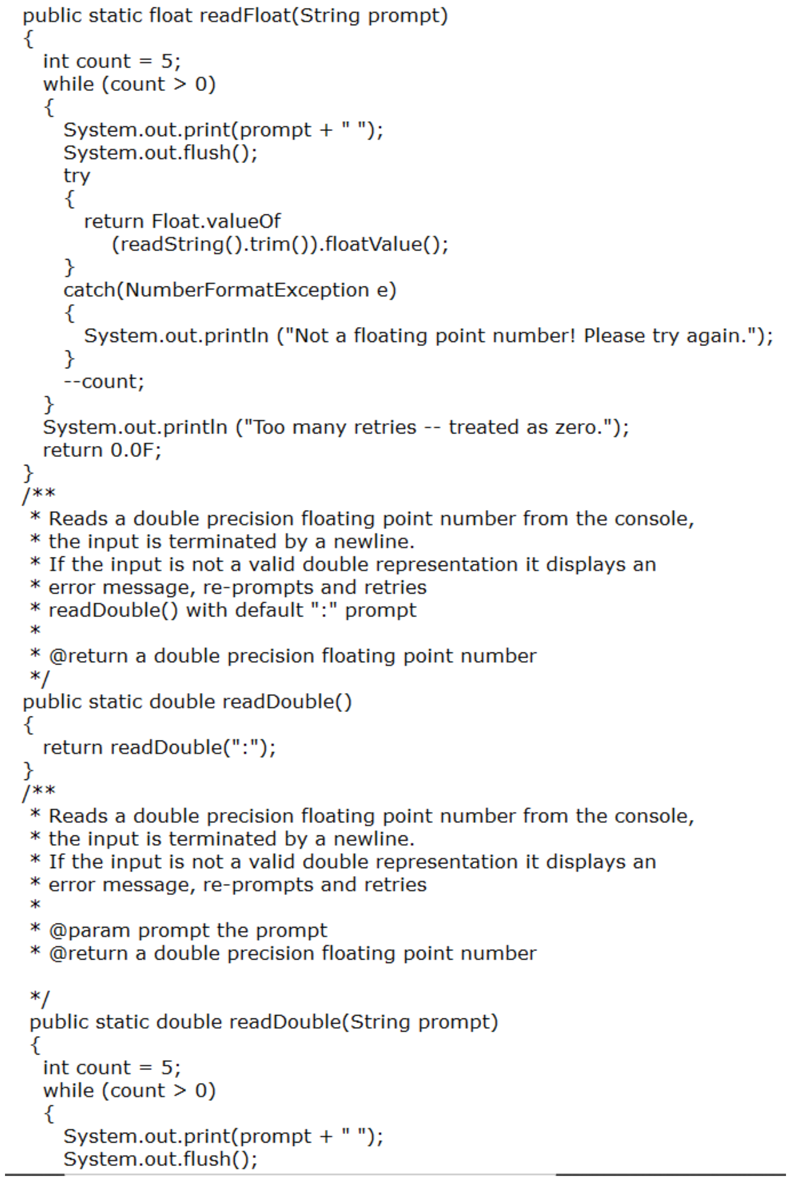
```
 *   public static void pause()
 *   public static void pause(String prompt)
 *   // Pauses execution and waits for the user to press <ENTER> or NEWLINE ('\n')
 *   // If not supplied as a parameter the default prompt is
 *   //          "Press Enter to continue......."
 *
 *   public static String readWord()
 *   // Returns a 'word' from the console. A word is any group of characters
 *   // terminated by whitespace (one or more ' ', '\t', '\n').  No prompt.
 *   public static void flush()
 *   // Ignore the rest of the current line, ie. flush the input stream
 *   // discarding any contents up to '\n' inclusive.  Does nothing if
 *   // '\n' is last char read.  Only useful after readWord().
 *
 * MODIFIED:
 *  version 1.7, 5 May 1997. retry limit. rka
 *  version 1.8, 25 November 1997. CFLF. abo
 *  version 1.9, 27 January 1998. JDK 1.1. abo
 * @version 2.0, 8 November 1999. javadoc. abo
 * @author Ewen Vowels
 * @author Rob Allen
 * @author Annette Oppenheim. School of IT, Swinburne University of Technology
 */


public class Keyboard
{
  /**
   * maintains last char read - used by readWord, flush, pause
   */
  private static int lastChar = 0;
  /**
   * internal value of end of line character
   */
  private static int endOfLine = 10;
  /**
   * internal representation of end of file
   */
  private static int endOfFile = -1;

  /**
   * Read a string from the console, the string is terminated by a newline.
   * No prompt.
   *
   * @return the input string (without the newline).
   */
  public static String readString()
  {
    int     ch = 0;
    String  r = "";
    boolean done = false;
```



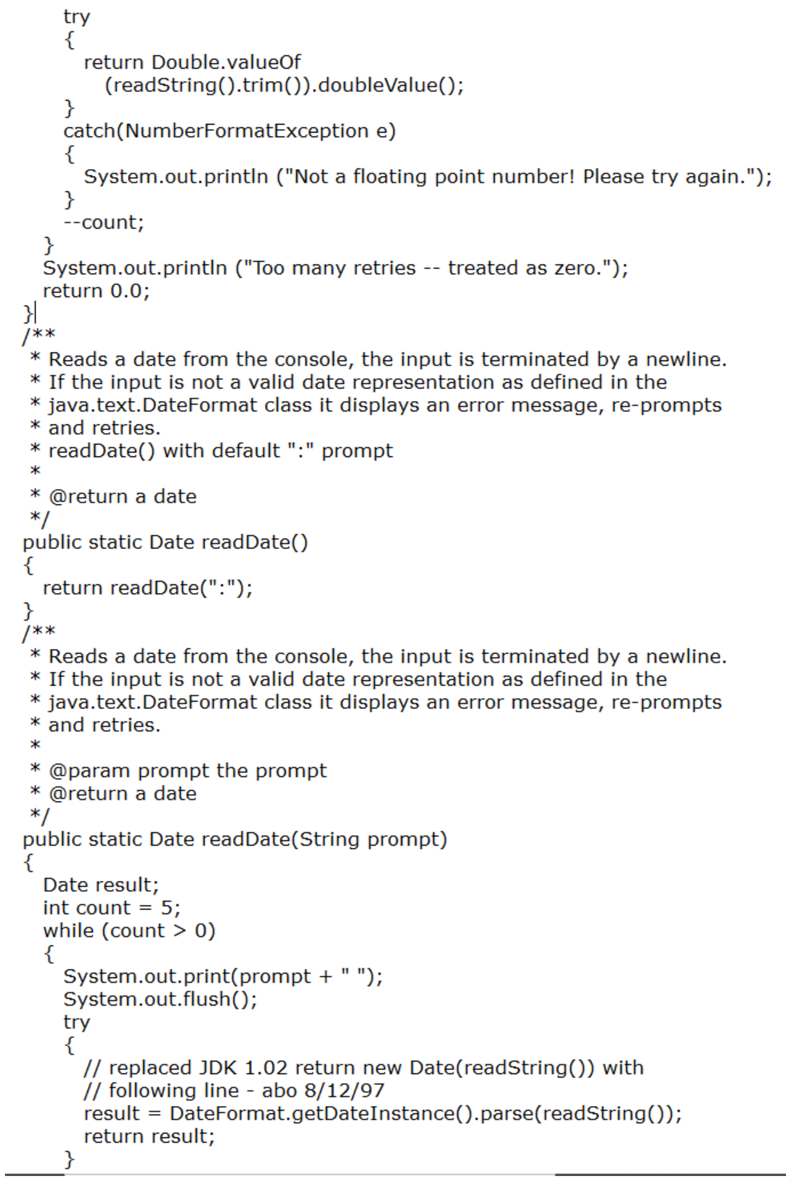
```
  while (!done)
  {
    try
    {
      ch = System.in.read();
      if (ch < 0 || (char)ch == '\n')
        done = true;
      // 'r' test added on 25/11/97 for JDK 1.1.4
      // readString() now works - abo
      else if ( (char) ch != '\r')
        r = r + (char) ch;
    }
    catch(java.io.IOException e)
    {
      done = true;
      return null;
    }
  }
  lastChar = ch;
  return r;
}

/**
 * Read a string from the console, the string is terminated by a newline.
 * readString() with prompting.
 *
 * @param prompt the prompt
 * @return the input string (without the newline).
 */
public static String readString(String prompt)
{
  System.out.print(prompt + " ");
  System.out.flush();
  return readString();
}

/**
 * Reads an integer from the console, the input is terminated by a newline.
 * If the input is not a valid integer representation it displays an
 * error message, re-prompts and retries.
 * readInt() with default ":" prompt
 *
 * @return an integer
 */
public static int readInt()
{
  return readInt(":");
}

/**
 * Reads an integer from the console, the input is terminated by a newline.
 * If the input is not a valid integer representation it displays an
 * error message, re-prompts and retries.
 * readInt() with default ":" prompt
 *
 * @return an integer
 */
public static int readInt()
{
  return readInt(":");
}

/**
 * Reads an integer from the console, the input is terminated by a newline.
 * If the input is not a valid integer representation it displays an
 * error message, re-prompts and retries.
 *
 * @param prompt the prompt
 * @return an integer
 */
 public static int readInt(String prompt)
 {
  int count = 5;
  while (count > 0)
  {
    System.out.print(prompt + " ");
    System.out.flush();
    try
    {
      return Integer.valueOf(readString().trim()).intValue();
    }
    catch(NumberFormatException e)
    {
      System.out.println ("Not an integer! Please try again.");
    }
    --count;
  }
  System.out.println ("Too many retries -- treated as zero.");
  return 0;
}

/**
 * Reads a long integer from the console, the input is terminated by a newline.
 * If the input is not a valid integer representation it displays an
 * error message, re-prompts and retries.
 * readLong() with default ":" prompt
 *
 * @return a long integer
 */
public static long readLong()
{
  return readLong(":");
```



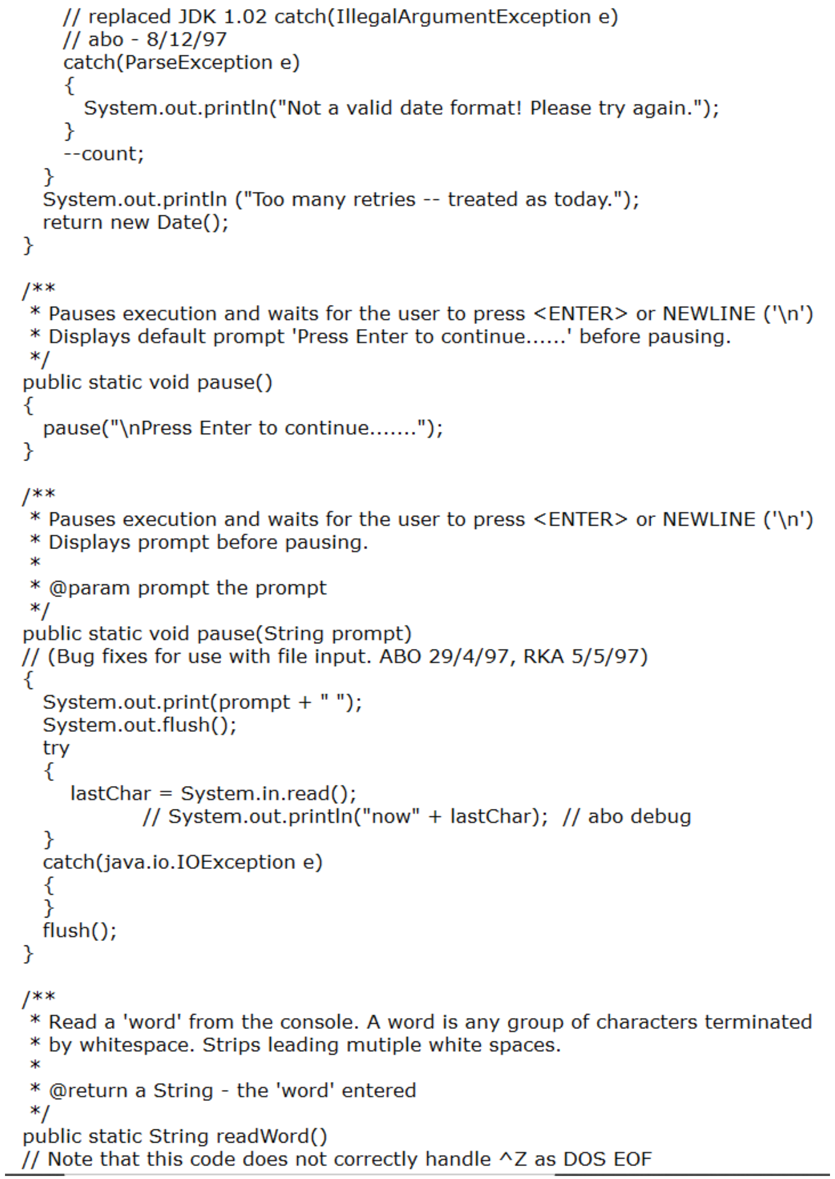
```
}
/**
 * Reads a long integer from the console, the input is terminated by a newline.
 * If the input is not a valid integer representation it displays an
 * error message, re-prompts and retries.
 *
 * @param prompt the prompt
 * @return a long integer
 */
public static long readLong(String prompt)
{
  int count = 5;
  while (count > 0)
  {
    System.out.print(prompt + " ");
    System.out.flush();
    try
    {
      return Long.valueOf(readString().trim()).longValue();
    }
    catch(NumberFormatException e)
    {
      System.out.println ("Not an integer! Please try again.");
    }
    --count;
  }
  System.out.println ("Too many retries -- treated as zero.");
  return 0L;
}

/**
 * Reads a floating point number from the console,
 * the input is terminated by a newline.
 * If the input is not a valid floating point representation it
 * displays an error message, re-prompts and retries.
 * readFloat() with default ":" prompt
 *
 * @return a floating point number
 */
public static float readFloat()
{
  return readFloat(":");
}
/**
 * Reads a floating point number from the console,
 * the input is terminated by a newline.
 * If the input is not a valid floating point representation it
 * displays an error message, re-prompts and retries.
 *
 * @param prompt the prompt
 * @return a floating point number
```



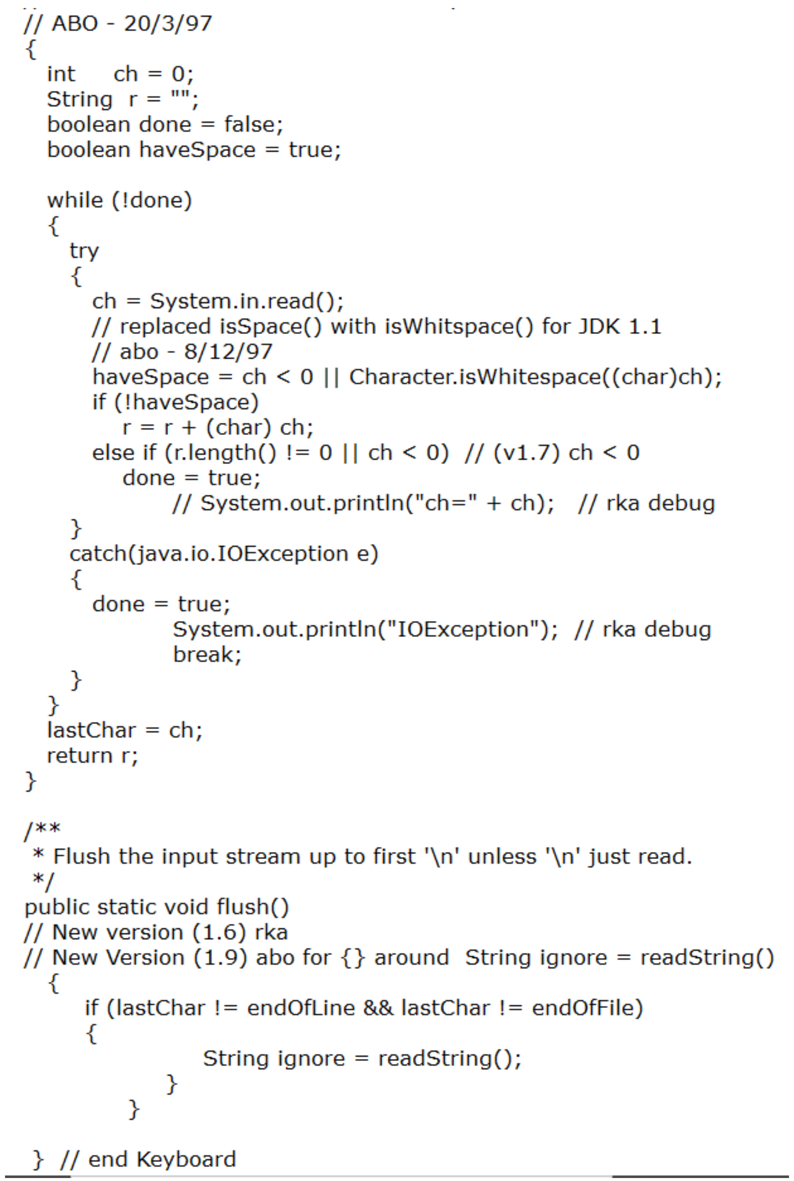
```
public static float readFloat(String prompt)
{
  int count = 5;
  while (count > 0)
  {
    System.out.print(prompt + " ");
    System.out.flush();
    try
    {
      return Float.valueOf
         (readString().trim()).floatValue();
    }
    catch(NumberFormatException e)
    {
      System.out.println ("Not a floating point number! Please try again.");
    }
    --count;
  }
  System.out.println ("Too many retries -- treated as zero.");
  return 0.0F;
}
/**
 * Reads a double precision floating point number from the console,
 * the input is terminated by a newline.
 * If the input is not a valid double representation it displays an
 * error message, re-prompts and retries
 * readDouble() with default ":" prompt
 *
 * @return a double precision floating point number
 */
public static double readDouble()
{
  return readDouble(":");
}
/**
 * Reads a double precision floating point number from the console,
 * the input is terminated by a newline.
 * If the input is not a valid double representation it displays an
 * error message, re-prompts and retries
 *
 * @param prompt the prompt
 * @return a double precision floating point number

 */
 public static double readDouble(String prompt)
 {
  int count = 5;
  while (count > 0)
  {
    System.out.print(prompt + " ");
    System.out.flush();
```



```
    try
    {
      return Double.valueOf
        (readString().trim()).doubleValue();
    }
    catch(NumberFormatException e)
    {
      System.out.println ("Not a floating point number! Please try again.");
    }
    --count;
  }
  System.out.println ("Too many retries -- treated as zero.");
  return 0.0;
}
/**
 * Reads a date from the console, the input is terminated by a newline.
 * If the input is not a valid date representation as defined in the
 * java.text.DateFormat class it displays an error message, re-prompts
 * and retries.
 * readDate() with default ":" prompt
 *
 * @return a date
 */
public static Date readDate()
{
  return readDate(":");
}
/**
 * Reads a date from the console, the input is terminated by a newline.
 * If the input is not a valid date representation as defined in the
 * java.text.DateFormat class it displays an error message, re-prompts
 * and retries.
 *
 * @param prompt the prompt
 * @return a date
 */
public static Date readDate(String prompt)
{
  Date result;
  int count = 5;
  while (count > 0)
  {
    System.out.print(prompt + " ");
    System.out.flush();
    try
    {
      // replaced JDK 1.02 return new Date(readString()) with
      // following line - abo 8/12/97
      result = DateFormat.getDateInstance().parse(readString());
      return result;
    }
```

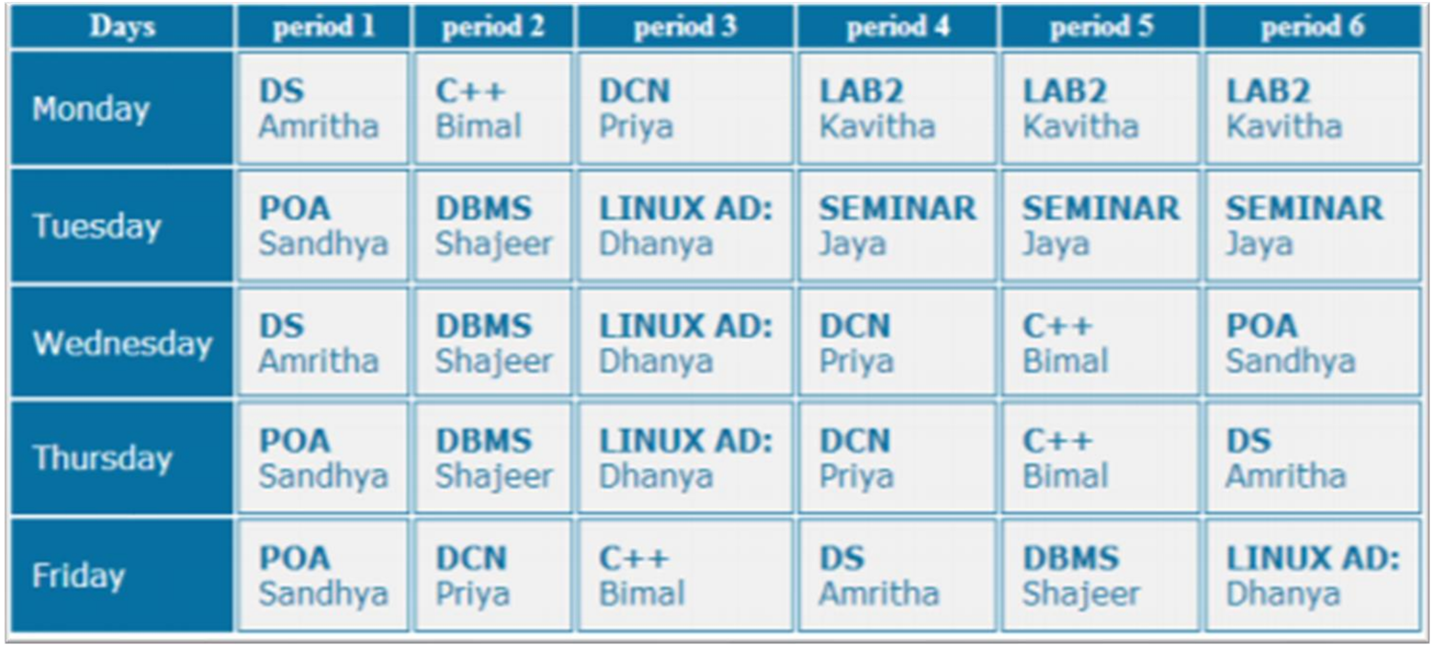


```
    // replaced JDK 1.02 catch(IllegalArgumentException e)
    // abo - 8/12/97
    catch(ParseException e)
    {
      System.out.println("Not a valid date format! Please try again.");
    }
    --count;
  }
  System.out.println ("Too many retries -- treated as today.");
  return new Date();
}

/**
 * Pauses execution and waits for the user to press <ENTER> or NEWLINE ('\n')
 * Displays default prompt 'Press Enter to continue......' before pausing.
 */
public static void pause()
{
  pause("\nPress Enter to continue.......");
}

/**
 * Pauses execution and waits for the user to press <ENTER> or NEWLINE ('\n')
 * Displays prompt before pausing.
 *
 * @param prompt the prompt
 */
public static void pause(String prompt)
// (Bug fixes for use with file input. ABO 29/4/97, RKA 5/5/97)
{
  System.out.print(prompt + " ");
  System.out.flush();
  try
  {
     lastChar = System.in.read();
            // System.out.println("now" + lastChar);  // abo debug
  }
  catch(java.io.IOException e)
  {
  }
  flush();
}

/**
 * Read a 'word' from the console. A word is any group of characters terminated
 * by whitespace. Strips leading mutiple white spaces.
 *
 * @return a String - the 'word' entered
 */
public static String readWord()
// Note that this code does not correctly handle ^Z as DOS EOF
```



```
// ABO - 20/3/97
{
  int     ch = 0;
  String  r = "";
  boolean done = false;
  boolean haveSpace = true;

  while (!done)
  {
    try
    {
      ch = System.in.read();
      // replaced isSpace() with isWhitspace() for JDK 1.1
      // abo - 8/12/97
      haveSpace = ch < 0 || Character.isWhitespace((char)ch);
      if (!haveSpace)
         r = r + (char) ch;
      else if (r.length() != 0 || ch < 0)  // (v1.7) ch < 0
         done = true;
              // System.out.println("ch=" + ch);   // rka debug
    }
    catch(java.io.IOException e)
    {
      done = true;
              System.out.println("IOException");  // rka debug
              break;
    }
  }
  lastChar = ch;
  return r;
}

/**
 * Flush the input stream up to first '\n' unless '\n' just read.
 */
public static void flush()
// New version (1.6) rka
// New Version (1.9) abo for {} around  String ignore = readString()
  {
     if (lastChar != endOfLine && lastChar != endOfFile)
     {
                String ignore = readString();
             }
         }

 }  // end Keyboard
```

VERIFICATION AND VALIDATION



Days	period 1	period 2	period 3	period 4	period 5	period 6
Monday	DS Amritha	C++ Bimal	DCN Priya	LAB2 Kavitha	LAB2 Kavitha	LAB2 Kavitha
Tuesday	POA Sandhya	DBMS Shajeer	LINUX AD: Dhanya	SEMINAR Jaya	SEMINAR Jaya	SEMINAR Jaya
Wednesday	DS Amritha	DBMS Shajeer	LINUX AD: Dhanya	DCN Priya	C++ Bimal	POA Sandhya
Thursday	POA Sandhya	DBMS Shajeer	LINUX AD: Dhanya	DCN Priya	C++ Bimal	DS Amritha
Friday	POA Sandhya	DCN Priya	C++ Bimal	DS Amritha	DBMS Shajeer	LINUX AD: Dhanya

RESULT ANALYSIS

We have a written a code for Timetable Management. Automatic Timetable Generator is an application for generating timetable automatically. It is a great difficult task that to manage many Faculty's and allocating subjects for them at a time manually. So the proposed system will help to overcome this disadvantage. Thus we can generate timetable for any number of courses and multiple semesters. This system will help to create dynamic pages so that for implementing such a system we can make use of the different tools are widely applicable and free to use also.

IDENTIFY LIMITATIONS AND FUTURE WORK

The main limitation of the Timetable Management System is that it is not linking to the current database. This is because if the system connects to the current database the requirement for the hardware and software is higher.

From this Timetable system, we are able to obtain useful information for future work.

Further development includes expanding algorithm for solving timetabling problem of more than one department at same time.

We can also improve modeling and search technique , reducing execution time and enhancing graphical user interface.

REFERENCES / BIBILIOGRAPHY

BOOKS

Software Engineering Fifth edition by R. S. Pressman, McGraw-Hill Publication, 1997

WEBSITES

1. http://en.wikipedia.org/wiki/MySQL
2. http://www.wpi.edu/Serlvets & JSP - Falkner Jones.pdf
3. http://edutechwiki.unige.ch/en/Educational_technology
4. http://www.wpi.edu/Images/Tutorial_JSP
5. http://en.wikipedia.org/wiki/JSP