```java
import java.io.Serializable;
import java.time.*;
import java.time.format.*;
import java.util.*;

public class Venue implements Serializable
{
    private String location;
    private int capacity;
    private Purpose purpose;
    private ArrayList<Lesson> lessons;
    static DateTimeFormatter timeFormat = DateTimeFormatter.ofPattern("h:mm a");

    public enum Purpose
    {
        LECTURE("Lecture"), TUTELAB("Tute/Lab");

        String displayName;

        Purpose(String displayName)
        {
            this.displayName = displayName;
        }

        public String toString()
        {
            return displayName;
        }
    }

    public Venue(String location, int capacity, Purpose purpose)
    {
        this.location = location;
        this.capacity = capacity;
        this.purpose = purpose;
        lessons = new ArrayList<Lesson>();
    }

    public void checkClash(DayOfWeek day, LocalTime start, LocalTime end)
            throws ClashException
    {

        for(int i = 0; i < lessons.size(); i++)
        {
            if(day == lessons.get(i).getDay())
            {
                if(overlap(start, end, lessons.get(i).getStart(),
                        lessons.get(i).getEnd()))
                {
                    throw new ClashException("Venue Clash");
                }
            }
        }
    }

    private boolean overlap(LocalTime start1, LocalTime end1, LocalTime start2,
            LocalTime end2)
```

```
58      {
59          return start1.isBefore(end2) && start2.isBefore(end1);
60      }
61
62      public void checkLessonBounds(DayOfWeek day, LocalTime start, LocalTime end)
63              throws LessonTimeOutOfBoundsException
64      {
65          if(day == DayOfWeek.SATURDAY || day == DayOfWeek.SUNDAY)
66          {
67              throw new LessonTimeOutOfBoundsException("Venues are not available"
68                      + " on weekends");
69          }
70          if(start.isBefore(LocalTime.of(8, 00))
71                  || start.isAfter(LocalTime.of(21, 00))
72                  || end.isBefore(LocalTime.of(8, 00))
73                  || end.isAfter(LocalTime.of(21, 00)))
74          {
75              throw new LessonTimeOutOfBoundsException("Venues are not available"
76                      + " before 8am or after 9am");
77          }
78      }
79
80      public void printTimetable()
81      {
82          // Get maximum length for each field
83          int courseIDMaxLen = "Course".length();
84          int activityMaxLen = "Activity".length(); // "Tutorial" or "Lecture"
85          int dayMaxLen = "Day".length();
86          int startTimeMaxLen = "Start Time".length(); // Always longer/equal to
87          int endTimeMaxLen = "End Time".length();     // any time
88          int staffMaxLen = "Staff".length();
89          for(Lesson lesson : lessons)
90          {
91              Course course = lesson.getCourseOffering().getCourse();
92
93              // Check if the length of this 'Course' field is largest so far
94              if(course.getID().length() > courseIDMaxLen)
95              {
96                  courseIDMaxLen = course.getID().length();
97              }
98
99              // Check if the length of this 'Day' field is largest so far
100             if(lesson.getDay().getDisplayName(TextStyle.FULL,
101                     Locale.getDefault()).length() > dayMaxLen)
102             {
103                 dayMaxLen = lesson.getDay().getDisplayName(TextStyle.FULL,
104                         Locale.getDefault()).length();
105             }
106
107             // Check if the length of this 'Staff' field is largest so far
108             if(lesson.getStaff().getName().length() > staffMaxLen)
109             {
110                 staffMaxLen = lesson.getStaff().getName().length();
111             }
112         }
113         int totalLength = courseIDMaxLen + activityMaxLen + dayMaxLen
114                 + startTimeMaxLen + endTimeMaxLen + staffMaxLen + 17;
```

```
115
116         // Print top of table
117         System.out.println(String.format("+%" + totalLength + "s+", "")
118                 .replace(' ', '-'));
119         // Print table headers
120         System.out.println(String.format("| %-" + courseIDMaxLen + "s | %-"
121                 + activityMaxLen + "s | %-"
122                 + dayMaxLen + "s | %-" + startTimeMaxLen + "s | %-"
123                 + endTimeMaxLen + "s | %-" + staffMaxLen + "s |", "Course",
124                 "Activity", "Day", "Start Time", "End Time", "Staff"));
125         // Print header separator
126         System.out.println(String.format("| %-" + courseIDMaxLen + "s | %-"
127                 + activityMaxLen + "s | %-" + dayMaxLen + "s | %-"
128                 + startTimeMaxLen + "s | %-" + endTimeMaxLen + "s | %-"
129                 + staffMaxLen + "s |", "", "", "", "", "", "", "")
130                 .replace(' ', '-'));
131
132         // Print each table row
133         for(Lesson lesson : lessons)
134         {
135             Course course = lesson.getCourseOffering().getCourse();
136
137             // Print lecture information
138             String s = "| ";
139             s += String.format("%-" + courseIDMaxLen + "s",
140                     course.getID());
141             if(lesson instanceof Lecture)
142             {
143                 s += String.format(" | %-" + activityMaxLen
144                         + "s | ", "Lecture");
145             }
146             else if(lesson instanceof Tutorial)
147             {
148                 s += String.format(" | %-" + activityMaxLen
149                         + "s | ", "Tutorial");
150             }
151             s += String.format("%-" + dayMaxLen + "s", lesson.getDay()
152                     .getDisplayName(TextStyle.FULL, Locale.getDefault()));
153             s += " | ";
154             s += String.format("%-" + startTimeMaxLen + "s",
155                     String.format("%8s",
156                             timeFormat.format(lesson.getStart())));
157             s += " | ";
158             s += String.format("%-" + endTimeMaxLen + "s",
159                     String.format("%8s", timeFormat.format(lesson.getEnd())));
160             s += " | ";
161             s += String.format("%-" + staffMaxLen + "s | ",
162                     lesson.getStaff().getName());
163             System.out.println(s);
164         }
165
166         // Print bottom of table
167         System.out.println(String.format("+%" + totalLength + "s+", "")
168                 .replace(' ', '-'));
169     }
170
171     public String getLocation()
```

```
172      {
173          return location;
174      }
175
176      public int getCapacity()
177      {
178          return capacity;
179      }
180
181      public Purpose getPurpose()
182      {
183          return purpose;
184      }
185
186      public ArrayList<Lesson> getLessons()
187      {
188          return lessons;
189      }
190
191      public void addLesson(Lesson lesson)
192      {
193          lessons.add(lesson);
194      }
195
196      public String toString()
197      {
198          return "Location: " + location + "\nCapacity: " + capacity
199                  + "\nPurpose: " + purpose;
200      }
201  }
202
```