Week 8

Comment

The purpose of this prac is to get experience with File I/O, and particularly representing complex objects as strings in order to be able to encode and decode them for writing to and reading from files.

1 Airline Schedule

Download the classes `Schedule.java`, `Airport.java`, `Flight.java`, `DayOfWeek.java` and `BadScheduleException.java` from Blackboard. These classes represent an airline's schedule of flights between a collection of airports.

1. Implement the incomplete `toString()` method in `Schedule.java` according to its Javadoc specification.
2. Implement the method `void save(String filename)` in `Schedule.java` to save a schedule to a file called `filename`. Any exceptions generated in this method should be propagated up instead of being handled within the method.
3. Implement the method `static Schedule load(String filename)` in `Schedule.java` to load a schedule from `filename`. As with `save()`, any exceptions should be propagated.
   - Your solution should make use of two private helper methods, `Flight readFlight(...)` and `Airport readAirport(...)` to reduce the length of `load()`.
   - If a numeric value fails to be parsed, or if a route uses an airport code that was not in the list of airports, then a `BadScheduleException` should be thrown.

2 Matrix (*)

Create the following class:

```java
public class Matrix {
    double[][] matrix;
    int rows;
    int columns;

    public Matrix(int rows, int columns) {
        matrix = new double[rows][columns];
        this.rows = rows;
        this.columns = columns;
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append('[');
```

```
        for (int i = 0; i < rows; i++) {
            if (i != 0) {
                builder.append(';');
            }
            builder.append('[');
            for (int j = 0; j < columns; j++) {
                if (j != 0) {
                    builder.append(',');
                }
                builder.append(matrix[i][j]);
            }
            builder.append(']');
        }
        builder.append(']');
        return builder.toString();
    }

    private void boundsCheck(int row, int column) {
        if (row < 0 || row >= rows || column < 0 || column >= columns) {
            throw new ArrayIndexOutOfBoundsException("(" + row + ','
                                                     + column + ")");
        }
    }

    public void set(int row, int column, double value) {
        boundsCheck(row, column);
        matrix[row][column] = value;
    }

    public double get(int row, int column) {
        boundsCheck(row, column);
        return matrix[row][column];
    }

    public void fill() {
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < columns; c++) {
                matrix[r][c] = r * columns + c;
            }
        }
    }
```

```
    public static void main(String args[]) {
        Matrix matrix = new Matrix(3, 2);
        matrix.fill();
        System.out.println(matrix);
```

```
    }
}
```

2.1 Permissions

What modifications would be necessary to prevent the following (answer each part independently):

1. Any modifications to the class state by other classes.
2. Redefinition of `set` and `get` in subclasses.

2.2 I/O

1. Add `void save(String filename)` to save a matrix to a file called `filename`.
2. Add `static Matrix load(String filename)` to load a matrix from `filename`.