Unit Testing
Team#NoLife
RideShare Application

Contents:

1. Declaration of variable used throughout unit tests
2. Each team member's unit tests

List of rides to use: rList

Ride 1:

Email:	ccpeck@ucsc.edu
Origin:	1520 Mission St, Santa Cruz, CA 95060
Dest:	1156 High St, Santa Cruz, CA 95064
Depart:	13:00
Arrive:	13:45
Date:	12/5/2015
Seats:	3
Driver/Rider:	Driver
Days of week:	

Ride 2:

Email:	bkeslin@ucsc.edu
Origin:	1156 High St, Santa Cruz, CA 95064
Dest:	920 Pacific Ave, Santa Cruz, CA 95060
Depart:	8:00
Arrive:	8:30
Date:	
Seats:	2
Driver/Rider:	Rider
Days of week:	Monday,Wednesday, Friday

Ride 3:

Email:	bkeslin@ucsc.edu
Origin:	7483 Phinney Way, San Jose, CA 95139
Dest:	359 Western Dr, Santa Cruz, CA 95060
Depart:	9:00
Arrive:	10:00
Date:	
Seats:	1
Driver/Rider:	Driver
Days of week:	Monday, Tuesday, Wednesday, Thursday, Friday

Christopher Peck:

Note: x refers to the variable inputtime

Note: I have assumed that the list of rides rList is valid. It has been checked beforehand.

Ridelist.java: line 341: **public List<Ride> filterByDepart(List<Ride> l, String inputtime)**

```
public List<Ride> filterByDepart(List<Ride> l, String inputtime) {
        List<Ride> rl = new ArrayList<Ride>();
        for (int i = 0; i < l.size(); i++) {
                if (this.convertTime(l.get(i).depart) >= this.convertTime(inputtime)) {
                        rl.add(l.get(i));
                }
        }
        return rl;
  }
```

Equivalence class 1.1: {x<0:00}
Equivalence class 1.2: {0:00<=x<=23:59}
Equivalence class 1.3: {23:59 < x}
Equivalence class 1.4: {x has am or pm in it}
Equivalence class 1.5: {x is not in proper time format}
Equivalence class 1.6:{x is null}

Input	Expected Output
(rList, “0:-01”)	Out of bounds exception
(rList, “0:00”)	{Ride1, Ride 2,Ride3}
(rList, “0:01”)	{Ride1, Ride 2,Ride3}
(rList, “8:30”)	{Ride 1, Ride 3}
(rList, “23:58”)	{}
(rList,”23:59”)	{}
(rList,”24:00”)	Out of bounds exception
(rList, “11:30am”)	Number Format Exception
(rlist, “one fourteen”)	Number Format Exception
(rList, null)	Null pointer exception

Timothy Ma

Note: I have assumed that the list of rides rList is valid. It has been checked beforehand.

Ridelist.java: line 273:

public List<Ride> filterByDrive(List<Ride> l, boolean b)

```
public List<Ride> filterByDrive(List<Ride> l, boolean b) {
        List<Ride> rl = new ArrayList<Ride>();
        for (int i = 0; i < l.size(); i++) {
                if (l.get(i).drive == b) {
                        rl.add(l.get(i));
                }
        }
        return rl;
  }
```

Equivalence Class 1.1: b == true.
Equivalence Class 1.2 b != true
Test Case: (rList is static so other variables can be stated)

Input	Output
(rList, b == true)	{Ride 1, Ride 3}
(rList, b == false)	{Ride 2}

Alice Yu

Ridelist.java: line 353:

```
public List<Ride> filterByArrive(List<Ride> l, String inputtime) {
        List<Ride> rl = new ArrayList<Ride>();
        for (int i = 0; i < l.size(); i++) {
                if (this.convertTime(l.get(i).arrive) <= this.convertTime(inputtime)) {
                        rl.add(l.get(i));
                }
        }
        return rl;
  }
```

Note: *x* refers to the inputted time (inputtime)
Note: I have assumed that we have NOT checked to see whether or not List<Ride> l is a valid list
Equivalence Class 1: List<Ride> l is not a valid list of rides that exists in the database
Equivalence Class 2.1: *x* is null
Equivalence Class 2.2: *x* < 0:00
Equivalence Class 2.3: 0:00 ≤ *x* ≤ 23:59
Equivalence Class 2.4: *x* > 23:59

Equivalence Class 2.5: *x* contains either the string “AM” or the string “PM”
Equivalence Class 2.6: *x* is a string of letters rather than numbers

Test Case Number	**Exact Input**	**Expected Output**
1	(invalidList, null)	Undefined List Error; Null Pointer Exception
2	(invalidList, “-0:01”)	Undefined List Error; Out of bounds exception
3	(invalidList, “0:00”)	Undefined List Error
4	(invalidList, “0:01”)	Undefined List Error; Out of bounds exception
5	(invalidList, “23:58”)	Undefined List Error
6	(invalidList, “23:59”)	Undefined List Error
7	(invalidList, “12:00”)	Undefined List Error
8	(invalidList, “24:00”)	Undefined List Error; Out of bounds exception
9	(invalidList, “6:30AM”)	Undefined List Error; Number Format Exception
10	(invalidList, “six thirty”)	Undefined List Error; Number Format Exception
11	(rList, null)	Null Pointer Exception
12	(rList, “-0:01”)	Out of bounds exception
13	(rList, “0:00”)	{}
14	(rList, “0:01”)	{}
15	(rList, “23:58”)	{Ride1, Ride2, Ride3}
16	(rList, “23:59”)	{Ride1, Ride2, Ride3}
17	(rList, “12:00”)	{Ride2, Ride3}

18	(rList, “24:00”)	Out of bounds exception
19	(rList, “6:30AM”)	Number Format Exception
20	(rList, “six thirty”)	Number Format Exception

Alix Feinsod

Note: I have assumed that the list of rides rList is valid. It has been checked beforehand.

```
Ridelist.java: line 182: public List<Ride> originRadius(List<Ride> l, GeoPt origin, String radius){
        int b = Integer.parseInt(radius);
        List<Ride> rl = new ArrayList<Ride>();
                for (int i = 0; i < l.size(); i++) {
                        if (this.findDist(l.get(i).start, origin) < b) {
                        rl.add(l.get(i));
                        }
                }
                return rl;
  }
```

Notes:
Let myOrigin = 1156 High Street, Santa Cruz, CA
Distance between myOrigin and Ride 1: 5.1499 km
Distance between myOrigin and Ride 3: 63.08628 km

Equivalence class 1.1: (rlist, myOrigin, radius <=0)
Equivalence class 1.2: (rlist, myOrigin, 0 < radius <= 5.1499)
Equivalence class 1.3: (rlist, myOrigin, 5.1499 < radius <= 63.08628)
Equivalence class 1.4: (rlist, myOrigin, radius > 63.08628)

Test Cases:

Input	Expected Output
(rlist, myOrigin, 0)	List <Ride> (Ride 2)
(rlist, myOrigin,2)	List <Ride> (Ride 2)
(rList, myOrigin, 6)	List <Ride> (Ride 1, Ride 2)
(rList, myOrigin, 65)	rList

<u>Brighton Keslin</u>
<u>Note: I have assumed that the list of rides rList is valid. It has been checked beforehand.</u>
Ridelist.java: line 362: public List<Ride> filterBySeatsMore(List<Ride> l, String inputseats)

```
  public List<Ride> filterBySeatsMore(List<Ride> l, String inputseats){
        List <Ride> rl = new ArrayList<Ride>();
        int seats = Integer.parseInt(inputseats);
        for (int i = 0; i < l.size(); i++) {
                if (l.get(i).seats >= seats) {
                        rl.add(l.get(i));
                }
        }
        return rl;
 }
```

Equivalence Class 1.1 : rList, String representing int
Equivalence Class 1.2 : rList, String not representing int
Equivalence Class 1.3 : rList, null string

Test Cases:

Input	Expected Output
(rList, 0)	rList
(rList, -1)	rList
(rList, 1)	List <Ride> (Ride1, Ride2)
(rList, Bacon)	NumberFormatException
(rList, null)	NumberFormatException

Because rList is a static list of rides which has been pre checked on entry into the database we can determine that it will always be correct which allows us to not check the case of rList being invalid.

<u>Carol:</u>
<u>Note: I have assumed that the list of rides rList is valid. It has been checked beforehand.</u>
Ridelist.java: line 285: public List<Ride> filterByEmail(List<Ride> l, String e)

```
public List<Ride> filterByEmail(List<Ride> l, String e) {
        List<Ride> rl = new ArrayList<Ride>();
        for (int i = 0; i < l.size(); i++) {
                if (l.get(i).email.equals(e)) {
                        rl.add(l.get(i));
                }
        }
        return rl;
}
```
Equivalence Class 1.1 : rList, String containing ‘@ucsc.edu’
Equivalence Class 1.2 : rList, String not containing ‘@ucsc.edu’
Equivalence Class 1.3: rList, Integers
Equivalence Class 1.4 : rList, null string

Input	Expected Output
(rList, bkeslin@ucsc.edu)	List <Ride> (Ride 2 , Ride 3)
(rList, carrots)	EmailFormatException
(rList, 1246)	EmailFormatException
(rList, null)	EmailFormatException