TESTING: Test Case Exercises

Equivalence classes

1. *Make equivalences classes for the input variable for this method:*
   `public boolean isEven(int n)`

   `1, 3, 5, 7, -1, -9` are all even numbers thus belong to one equivalence class.

2. *Make equivalences classes for an input variable that represents a mortgage applicant's salary. The valid*
   range is $1000 pr. month to $75,000 pr. month

   The equivalence classes will be the following:
   - **invalid:** values < 1.000 and 75.000 > values; for example: `-500, 250, 999, 75.001, 90.000`
   - **valid:** 1.000 < values < 75.000; for example: `1000, 2300, 17.500, 75.000`

3. *Make equivalences classes for the input variables for this method:*
   `public static int getNumDaysinMonth(int month, int year)`

   The month parameter will be valid if it is in the range of 1 … 12 edges including. Thus two equivalence classes will exist:
   - **valid**: 1 ≤ month ≤ 12; for example: `1, 4, 9, 12`
   - **invalid**: month < 1 and month > 12; for example: `-12, 0, 13, 55`

   The year parameter's valid input range will depend on the requirements of the application. If not specified exactly, we can accept years far ahead in the future, like year 12.000.000 as well as negative numbers to indicate years B.C. Thus we can say that there will be one equivalence class: all **integer** numbers.

Boundary Analysis

1. Since the boundaries are at the edge cases of a range, when it comes to even and odd numbers it doesn't make sense to search for boundary values.
2. Boundaries are 1.000 and 75.000. Thus we want to test the following: `999, 1000, 75.000, 75.001`
3. We can do boundary analysis for the month variable, checking these values: `0, 1, 12, 13`

Decision Tables

1. *No charges are reimbursed (DK: refunderet) to a patient until the deductible (DK: selvrisiko) has been*
   met. After the deductible has been met, reimburse 50% for Doctor's Office visits or 80% for Hospital
   visits.

```
CONDITIONS
------------------------------------------------------------------
*Hospital visit:          truefalse   truefalse
Deductable is met:    truetruefalse   false
------------------------------------------------------------------
ACTIONS
------------------------------------------------------------------
Reimburse 0%                              Y       Y
Reimburse 50%                    Y
Reimburse 80%             Y

*When the hospital visit is false, the doctor has been visited.
```

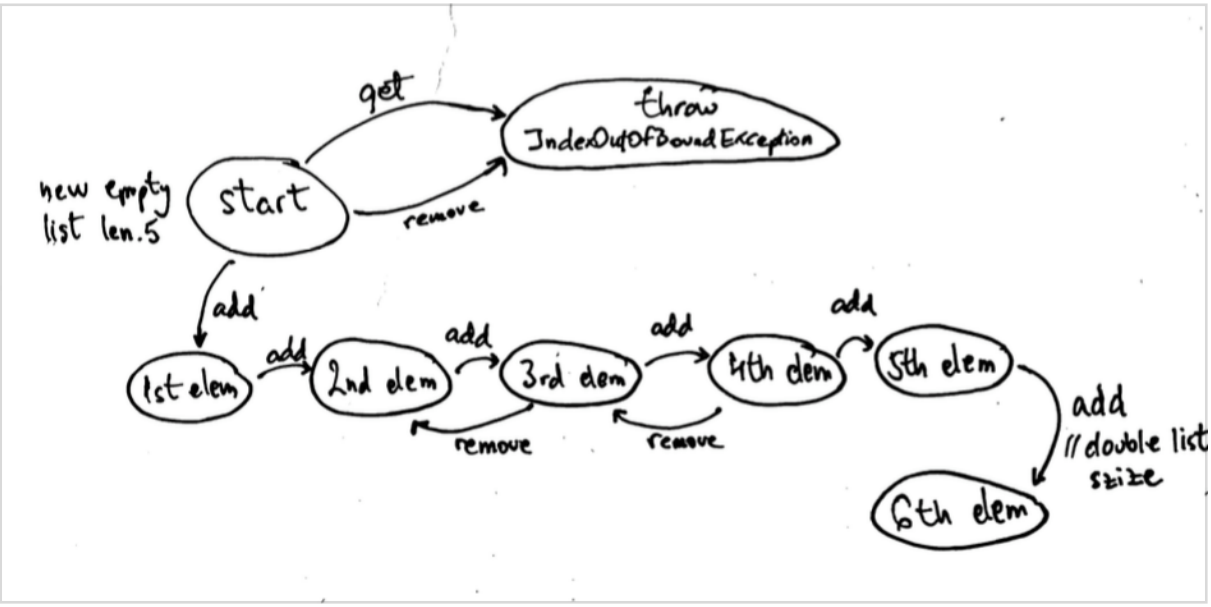
2. *Make a decision table for leap years.*
   Leap year: Most years that are evenly divisible by 4 are leap years.
   An exception to this rule is that years that are evenly divisible by 100 are not leap years, unless they are also
   evenly divisible by 400, in which case they are leap years.

```
CONDITIONS
-------------------------------------------------------------------------------
X % 4 == 0       true   true   true   true   false  false  false  false
X % 100 == 0     true   true   false  false  true   true   false  false
X % 400 == 0     true   false  true   false  true   false  true   false
-------------------------------------------------------------------------------
ACTIONS
-------------------------------------------------------------------------------
LEAP YEAR:       true   false  true   true   false  false  false  false
```

State transition

Test Cases Exercise · GitHub

1. State diagram



2.

TEST CASES	EXPECTED RESULT
=========	==========
Create new list	list.length = 0
Remove element from empty list	IndexOutOfBoundsException
Add new element	list.length = 1
Remove element	list.length = 0
Add 6 elements and get last one	get(last) = added element
Add element at index and get it	same as added

3. See GitHub link
4. See GitHub link
5. Drawing a State Diagram appears more useful for the visually gifted. A table requires us to define connections line by line, but for humans a diagram connecting the states with lines could be more visually comprehensive.
6. In my tests I have went through the different states so that the non-equivalent states are all tested. I have covered the initial state, error state, overflow state etc.
   The transitions were mostly all tested as well. Thus I think we have a good test coverage with at least 80% of the cases being covered in our 7 test cases.

Suggestions

I think it would be useful to hand out the code on GitHub, maybe as a gist. That way we just clone it and are ready to code, no need to set up a new project, copy and stitch the code from the pdf.