Homework T2 Submission Document

You can work individually or in groups of 2.

Names: KD Adkins Joanne Wardell

Test Derivation for *addEmp(e:Employee)*

We will use the category-partition method to derive a set of test cases for this method.

1. Identify what criteria must hold true for each test case to pass

The side-effects of this method are:

- numEmps becomes incremented by 1 if the operation succeeds in adding an employee to the store
- the getNumEmps method will return a different value if the operation succeeds in adding the employee
- the getTotalHours method could return a different value depending on the state of the Employee which might be added
- the getTotalPay method could return a different value depending on the state of the Employee which might be added
- the removeEmployee method could behave differently if the Employee is added successfully

2. Identify inputs/parameters for each feature under test.

The input parameters are:

- primitive data type numEmps: integer
  - o the number of employees that a currently in the store
- reference data type emps: Employee[]
  - o the container in which the emps are collected
- reference data type e: Employee
  - o the employee that is trying to be added
- primitive data type added: Boolean
  - o the indication of success or failure upon add attempt

3. Identify the categories/characteristics for each input/parameter.

Categories of numEmps:

- value

Categories of emps:

- length
- content

Categories of e:

- state

Categories of added:

- value

4. Partition categories into choices

The choices for the numEmps parameter are *positions remaining = PR* and *no positions remaining = P*.

- PR – numEmps ∈ [0,19]
- P – numEmps = 20

The choices for the emps parameter are *full = F* and *not full = NF*.

- NF – emps array has indices [0,19] available for use

- F – emps array has no indices available for use

The choices for the e parameter are *null = N* and *non-null = NN*.

- NN – e is a reference to an Employee object
- N – e is the null data type
- The choices for the added parameter are *success = S* and *fail = FA*.

5. Number of tests before constraints added

Input	Category	Choices	Coded Choices
numEmps	Value	Positions Remaining, No Positions Remaining	PR,P
emps	Length	Full, Not Full	F,NF
e	State	Null, Not Null	N,NN
added	Value	Success, Fail	T,FA

By the counting principle, there are 2*2*2*2 = 16 possible combinations before constraints are added.
PRFNT, PRFNFA, PRFNNT, PRFNNFA,
PRNFNT, PRNFNFA, PRNFNNT, PRNFNNFA,
PFNT, PFNFA, PFNNT, PFNNFA,
PNFNT, PNFNFA, PNFNNT, PNFNNFA

The following combinations are removed because they are invalid:
PRFNT – This means that there are no positions remaining but the employee was added successfully.
PFNT– This means that there are no positions remaining but the employee was added successfully.
PNFNT – This means that there are no positions remaining but the employee was added successfully.
PFNNT– This means that there are no positions remaining but the employee was added successfully.
PRFNT – This means that the array is full but there are positions remaining and the employee was added successfully.
PRFNNT – This means that the array is full but there are positions remaining and the employee was added successfully.
PRNFNT – This means that the array is full but there are positions remaining and the employee was added successfully.

This leaves 9 combinations.

6. TSL Input

The *TSL* input file is shown below.

```
NumEmps:
  PositionsRemaining.
  NoPositionsRemaining. [property max]
Emps:
  Full.      [property full]
  NotFull.
E:
  Null.   [property null]
  NotNull.
Added:
  Success.   [if !full && !max && !null]
  Fail.
```

Justification for the constraints:

Constraint	Justification

Employees can only be added if there is room in the emps array and if the number of current employees is less than 20.	Employees can only be added if the emps array has indices available for use and numEmps is suggests that positions are remaining.
Only references to Employee objects can be added, no null data can be added.	Null objects are not handled, can cause null pointer exceptions, and cause failures.

7. TSL Output – Test Frames

The 9 test frames are show below:

```
Test Case 1              (Key = 1.1.1.2.)
   NumEmps :  PositionsRemaining
   Emps    :  Full
   E       :  Null
   Added   :  Fail


Test Case 2              (Key = 1.1.2.2.)
   NumEmps :  PositionsRemaining
   Emps    :  Full
   E       :  NotNull
   Added   :  Fail


Test Case 3              (Key = 1.2.1.2.)
   NumEmps :  PositionsRemaining
   Emps    :  NotFull
   E       :  Null
   Added   :  Fail


Test Case 4              (Key = 1.2.2.1.)
   NumEmps :  PositionsRemaining
   Emps    :  NotFull
   E       :  NotNull
   Added   :  Success


Test Case 5              (Key = 1.2.2.2.)
   NumEmps :  PositionsRemaining
   Emps    :  NotFull
   E       :  NotNull
   Added   :  Fail


Test Case 6              (Key = 2.1.1.2.)
   NumEmps :  NoPositionsRemaining
   Emps    :  Full
   E       :  Null
   Added   :  Fail


Test Case 7              (Key = 2.1.2.2.)
   NumEmps :  NoPositionsRemaining
   Emps    :  Full
   E       :  NotNull
```

```
    Added    :  Fail


Test Case 8             (Key = 2.2.1.2.)
    NumEmps :  NoPositionsRemaining
    Emps    :  NotFull
    E       :  Null
    Added   :  Fail


Test Case 9             (Key = 2.2.2.2.)
    NumEmps :  NoPositionsRemaining
    Emps    :  NotFull
    E       :  NotNull
    Added   :  Fail
```

8. Test Cases

The test cases we derived are shown below.

PRFNFA, PRFNNFA, PRNFNFA, PRNFNNT, PRNFNNFA, PFNFA, PFNNFA, PNFNFA, PNFNNFA
Notice that the highlighted cases are not valid for logical reasons.
PRFNFA – It is impossible for there to be positions remaining with the emps array being full.
PRFNNFA – It is impossible for there to be positions remaining with the emps array being full.
PRNFNNFA – If there are positions remaining and the emp is not null, then it should be successfully added.
PNFNFA – It is impossible for there to be no positions remaining while the emps array is not full.
PNFNNFA – It is impossible for there to be no positions remaining while the emps array is not full.

We are left with 4 test cases.

Test	Value [PR / P]	Length [F / NF]	State [N / NN]	Value [T / FA]
1	PR	NF	N	FA
2	PR	NF	NN	T
3	P	F	N	FA
4	P	F	NN	FA

Test Derivation for *removeEmployee(pos:int):Employee*

We will use the category-partition method to derive a set of test cases for this method.

1. Identify what criteria must hold true for each test case to pass

The side-effects of this method are:

- OutOfBoundsException. If a user removes an employee at the end of the array, it'll try and shift out of bounds.
- If we run the method and the array is empty.

2. Identify inputs/parameters for each feature under test.

The index being passed through is needed.
Variable to return the moved employee will be needed.
Will need the employees array and variable for number of employees

3. Identify the categories/characteristics for each input/parameter.

The index that gets passed is an integer.
Variable that holds the removed employee should be of type Employee.
Employee array is type employee with values.
Number of employees is a positive integer.

4. Partition categories into choices

Exceed Boundaries: index = Is the index < 0 or > 20
Valid Data: Null, Non-Null, Zero, Non-Zero

5. Number of tests before constraints added

Input	Category	Choices	Coded Choices
index	Value	Negative, Positive, Exceed	N, P, X
emps	Value	Null, NN	N, NN
numEmp	Value	Zero, Non-Zero	Z, NZ

I have 12 combinations (3*2*2).
NNZ, NNNZ, NNNZ, NNNNZ, PNZ, PNNZ, PNNZ, PNNNZ, XNZ, XNNZ, XNNZ, XNNNZ

6. TSL Input

The *TSL* input file is shown below.

Justification for the constraints:

Constraint	Justification
NNNZ	There's no way you can have a null array of employees and numEmp is >= 1

PNNZ	There’s no way you can have a null array of employees and numEmp is >= 1
XNNZ	There’s no way you can have a null array of employees and numEmp is >= 1
PNNZ	numEmps can’t be zero if there’s non-null values in emps
XNNZ	numEmps can’t be zero if there’s non-null values in emps
NNNZ	numEmps can’t be zero if there’s non-null values in emps

7. TSL Output – Test Frames

The _6_ test frames are show below:

8. Test Cases

The test cases we derived are shown below.

[Remove unused columns]

Test	[Value]	[Value]	[Value]					
1	N	N	Z					
2	N	N	NZ					
3	N	NN	Z					
4	N	NN	NZ					
5	P	N	Z					
6	P	N	NZ					
7	P	NN	Z					
8	P	NN	NZ					
9	X	N	Z					
10	X	N	NZ					
11	X	NN	Z					
12	X	NN	NZ					