Testing Document

for

Assignment 8

EECS 293

Prepared by: **Michael Thompson**

October 24, 2019

Table of Contents

Notation

- Code Coverage - CC
- Branch Coverage - B<#>
- Boundary Coverage - b<#>
- Compound Boundary - c<#>

Photo Time

PhotoTime

Conditions

Goal	Notes	Condition
Code Coverage	Call constructor	None

Separate Tests

Test Condition	Condition Satisfied	Assertion
“A”, 1, 2, 1	CC	No errors thrown

validatePriority

Conditions

Goal	Notes	Condition
Code Coverage	All conditions true	_priority >= 0
Branch Coverage	assert _priority >= 0 false	_priority < 0
Boundary	assert _priority >= 0	_priority > 0
Boundary	assert _priority >= 0	_priority = 0
Boundary	assert _priority >= 0	_priority < 0

Separate Tests

Test Condition	Condition Satisfied	Assertion
_priority > 0	CC, b1	No error thrown
_priority == 0	CC, b2	No error thrown
_priority < 0	B1, b3	AssertionError thrown

validateStartEndTime

Conditions

Goal	Notes	Condition
Code Coverage	All conditions true	_startTime.isBefore(_endTime)
Branch coverage	_startTime.isBefore (_endTime) false	_startTime.isAfter(_endTime) _startTime.equals(_endTime)
Boundary	_startTime.isBefore(_endTime)	_startTime.isBefore(_endTime)
Boundary	_startTime.isAfter(_endTime)	_startTime.isAfter(_endTime)
Boundary	_startTime.equals(_endTime)	_startTime.equals(_endTime)

Separate Tests

Test Condition	Condition Satisfied	Assertion
_startTime is 1, _endTime is 2	CC, b1	No error thrown
_startTime is 2, _endTime is 1	B1, b2	AssertionError thrown
_startTime is 1, _endTime is 1	b3	AssertionError thrown

of

Condition

Goal	Notes	Condition
Code Coverage	Call builder	None
Branch Coverage	Error checking	_landMark == null
Branch Coverage	Error checking	_startTime == null
Branch Coverage	Error checking	_endTime == null
Branch Coverage	Error checking	_priority == null

Separate Tests

Test Condition	Condition Satisfied	Assertion
All valid inputs	CC	No error thrown
_landMark = null	B1	NullPointerException thrown
_startTime = null	B2	NullPointerException thrown
_endTime = null	B3	NullPointerException thrown
_priority = null	B4	NullPointerException thrown

of

Conditions

Goal	Notes	Condition
Code Coverage	Call builder	None
Branch Coverage	Error checking	_landMark == null
Branch Coverage	Error checking	_startTime == null
Branch Coverage	Error checking	_endTime == null
Branch Coverage	Error checking	_priority == null

Separate Tests

Test Condition	Condition Satisfied	Assertion
All valid inputs	CC	No error thrown
_landMark = null	B1	NullPointerException thrown
_startTime = null	B2	NullPointerException thrown
_endTime = null	B3	NullPointerException thrown
_priority = null	B4	NullPointerException thrown

value

Conditions

Goal	Notes	Condition
Code Coverage	Call getter	return value = constructor value

Separate Tests

Test Condition	Condition Satisfied	Assertion
All valid inputs	CC	getValue = constructor value

linkedObject

Conditions

Goal	Notes	Condition
Code Coverage	Call getter	Return value = constructor value

Separate Tests

Test Condition	Condition Satisfied	Assertion
All valid inputs	CC	Return value = constructor value

doesOverlap

Conditions

Goal	Notes	Condition
Code Coverage	All conditions True	!this.getM_endTime().isBefore(temp.getM_startTime()) and !this.getM_startTime().isAfter(temp.getM_endTime())
Boundary	_otherWeightedJobSchedulable is not type PhotoTime	!_otherWeightedJobSchedulable.getClass().equals(PhotoTime.class)
Boundary	this.getM_endTime().isBefore(temp.getM_startTime())	this.getM_endTime().isBefore(temp.getM_startTime())
Boundary	this.getM_endTime().isBefore(temp.getM_startTime())	this.getM_endTime().isAfter(temp.getM_startTime())
Boundary	this.getM_endTime().isBefore(temp.getM_startTime())	this.getM_endTime().equals(temp.getM_startTime())
Boundary	this.getM_startTime().isAfter(temp.getM_endTime())	this.getM_startTime().isAfter(temp.getM_endTime())
Boundary	this.getM_startTime().isAfter(temp.getM_endTime())	this.getM_startTime().isBefore(temp.getM_endTime())
Boundary	this.getM_startTime().isAfter(temp.getM_endTime())	this.getM_startTime().equals(temp.getM_endTime())
Compound Boundary	this.getM_endTime().isBefore(temp.getM_startTime()) is false and this.getM_startTime().isAfter(temp.getM_endTime()) is true	!this.getM_endTime().isBefore(temp.getM_startTime()) and this.getM_startTime().isAfter(temp.getM_endTime())
Compound Boundary	this.getM_endTime().isBefore(temp.getM_startTime()) is true and this.getM_startTime().isAfter(temp.getM_endTime()) is false	this.getM_endTime().isBefore(temp.getM_startTime()) and !this.getM_startTime().isAfter(temp.getM_endTime())

Separate Tests

Test Condition	Condition Satisfied	Assertion

this.startTime = 1
this.endTime = 3
other.startTime = 2
other.endTime = 4
	CC, b3, b6	assertTrue

this.startTime = 3
this.endTime = 4
other.startTime = 1
other.endTime = 2
	C1, b3, b5	assertFalse

this.startTime = 1
this.endTime = 2
other.startTime = 3
other.endTime = 4
	C2, b2, b6	assertFalse

this.startTime = 1
this.endTime = 2
other.startTime = 2
other.endTime = 3
	b4	assertTrue

this.startTime = 2
this.endTime = 3
other.startTime = 1
other.endTime = 2
	b7	assertTrue

Make a new class that is not type PhotoTime that implements weightedJobSchedulable	b1	AssertionError

linkPredecessor

Conditions

Goal	Notes	Condition
Code Coverage	Call setter	m_linkedPhotoTime = set value after called
Branch Coverage	Error checking	_newLink == null

Separate Tests

Test Condition	Condition Satisfied	Assertion
Valid weightedJobSchedulable object	CC	No error thrown
null	b1	NullPointerException

compareTo

Conditions

Goal	Notes	Condition
Branch Coverage	Error checking	_otherPhtoTime == null
Branch Coverage	this.getM_endTime().isBefore(_otherPhotoTime.getM_endTime()) is true	this.getM_endTime().isBefore(_otherPhotoTime.getM_endTime())
Branch Coverage	this.getM_endTime().isBefore(_otherPhotoTime.getM_endTime()) is false	!this.getM_endTime().isBefore(_otherPhotoTime.getM_endTime())
Branch Coverage	this.getM_endTime().isAfter(_otherPhotoTime.getM_endTime()) is true	this.getM_endTime().isAfter(_otherPhotoTime.getM_endTime())
Branch Coverage	this.getM_endTime().isAfter(_otherPhotoTime.getM_endTime()) is false	!this.getM_endTime().isAfter(_otherPhotoTime.getM_endTime())
Branch Coverage	this.getValue().compareTo(_otherPhotoTime.getValue()) == 0	this.getValue().compareTo(_otherPhotoTime.getValue()) == 0
Branch Coverage	this.getValue().compareTo(_otherPhotoTime.getValue()) == 0	this.getValue().compareTo(_otherPhotoTime.getValue()) != 0
Boundary	this.getValue > _otherPhotoTime.getValue() is true	this.getValue > _otherPhotoTime.getValue()
Boundary	this.getValue < _otherPhotoTime.getValue() is true	this.getValue < _otherPhotoTime.getValue()
Boundary	this.getValue == _otherPhotoTime.getValue() is true	this.getValue == _otherPhotoTime.getValue()

Separate Tests

Test Condition	Condition Satisfied	Assertion
_otherPhotoTime = null	B1	NullPointerException

this.startTime = 1
this.endTime = 2
other.startTime = 1
other.endTime = 3
	B2	Equals -1

this.startTime = 1
this.endTime = 3
other.startTime = 1
other.endTime = 2
	B3, B4	Equals 1

This.value = 1
Other.value = 2
	B5, b1	Equals -1

This.value = 2
Other.value = 1
	B6, b2	Equals 1

This.value = 1
Other.value = 1
Landmark = “a”
Landmark = “b”
	B7, b3	Equals -1

Photo Schedule

PhotoSchedule

Conditions

Goal	Notes	Condition
Code coverage	All conditions true	_photoTimes != φ
Branch coverage	_photoTime is empty	_photoTimes = φ

Separate Tests

Test Condition	Condition Satisfied	Assertion

S = {X1}
X1 = some valid PhotoTime
	CC	No errors thrown

S = {}	Branch coverage	No errors thrown

validateStartEndTime

Conditions

Goal	Notes	Condition
Code Coverage	All conditions true	_startTime.isBefore(_endTime)
Branch coverage	_startTime.isBefore (_endTime) false	_startTime.isAfter(_endTime) _startTime.equals(_endTime)
Boundary	_startTime.isBefore(_endTime)	_startTime.isBefore(_endTime)
Boundary	_startTime.isAfter(_endTime)	_startTime.isAfter(_endTime)
Boundary	_startTime.equals(_endTime)	_startTime.equals(_endTime)

Separate Tests

Test Condition	Condition Satisfied	Assertion
startTime = 1 endTime = 2	CC, b1	No errors thrown
startTime = 2 endTime = 1	B1, b2	AssertionError
startTime = 1 endTime = 1	B1, b3	AssertionError

of

Conditions

Goal	Notes	Condition
Code Coverage	Call builder	No error thrown
Branch Coverage	Error checking	_startTime = null
Branch Coverage	Error checking	_endTime = null

Separate Tests

Test Condition	Condition Satisfied	Assertion
Valid inputs	CC	No error thrown
_startTime = null	B1	NullPointerException thrown
_endTime = null	B2	NullPointerException thrown

of

Conditions

Goal	Notes	Condition
Code Coverage	Call builder	No error thrown
Branch Coverage	Error checking	_phototimes = null
Branch Coverage	Error checking	_startTime = null
Branch Coverage	Error checking	_endTime = null

Separate Test

Test Condition	Condition Satisfied	Assertion
Valid inputs	CC	No error thrown
_photoTimes = null	B1	NullPointerException thrown
_startTime = null	B2	NullPointerException thrown
_endTime = null	B3	NullPointerException thrown

addPhotoTime

Conditions

Goal	Notes	Condition
Code coverage	All inputs true	_photo.getM_startTime().isAfter(m_startTime) and _photo.getM_endTime().isBefore(m_endTime)
Branch coverage	_photo is null	NullPointerException thrown
Branch coverage	_photo.getM_startTime().isAfter(m_startTime) and _photo.getM_endTime().isBefore(m_endTime)	!_photo.getM_startTime().isAfter(m_startTime) and _photo.getM_endTime().isBefore(m_endTime)
Branch coverage	_photo.getM_startTime().isAfter(m_startTime) and _photo.getM_endTime().isBefore(m_endTime)	_photo.getM_startTime().isAfter(m_startTime) and !_photo.getM_endTime().isBefore(m_endTime)
Branch coverage	_photo.getM_startTime().isAfter(m_startTime) and _photo.getM_endTime().isBefore(m_endTime)	!_photo.getM_startTime().isAfter(m_startTime) and !_photo.getM_endTime().isBefore(m_endTime)
Boundary coverage	_photo.getM_startTime().isAfter(m_startTime)	_photo.getM_startTime().isAfter(m_startTime)
Boundary coverage	_photo.getM_startTime().isAfter(m_startTime)	_photo.getM_startTime().isBefore(m_startTime)
Boundary coverage	_photo.getM_startTime().isAfter(m_startTime)	_photo.getM_startTime().equals(m_startTime)
Boundary coverage	_photo.getM_endTime().isBefore(m_endTime)	_photo.getM_endTime().isBefore(m_endTime)
Boundary coverage	_photo.getM_endTime().isBefore(m_endTime)	_photo.getM_endTime().isAfter(m_endTime)
Boundary coverage	_photo.getM_endTime().isBefore(m_endTime)	_photo.getM_endTime().equals(m_endTime)

Separate Test

Test Condition	Condition Satisfied	Assertion
_photo = null	B1	NullPointerException

this.startTime = 2
this.endTime = 5
photo.startTime = 3
photo.endTime = 4
	CC, b1, b5	Assert true

this.startTime = 2
this.endTime = 5
photo.startTime = 1
photo.endTime = 4
	B1, b2, b4	Assert false

this.startTime = 2
this.endTime = 5
photo.startTime = 3
photo.endTime = 6
	B2, b1, b5	Assert false

this.startTime = 2
this.endTime = 5
photo.startTime = 1
photo.endTime = 6
	B3, b2, b5	Assert false

this.startTime = 1
this.endTime = 3
photo.startTime = 1
photo.endTime = 2
	b3	Assert false

this.startTime = 1
this.endTime = 3
photo.startTime = 2
photo.endTime = 3
	b6	Assert false

removePhotoTime

Conditions

Goal	Notes	Condition
Code coverage	All conditions true	_photo is in the list
Branch Coverage	Error checking	_photo = null
Branch Coverage	.remove is false	_photo is not in the set

Separate Test

Test Condition	Condition Satisfied	Assertion
photo is valid and already in the list	CC	Assert True
photo is null	B1	NullPointerException
photo is not in the list	B2	Assert False

schedule

Conditions

Goal	Notes	Condition
Code coverage	Call routine	No errors thrown

Separate Test

Test Condition	Condition Satisfied	Assertion
Call routine	CC	No errors thrown

Weighted Job Schedule

linkBestPredecessor

Conditions

Goal	Notes	Condition
Code coverage	All conditions true	!_focusList.get(endIndex).doesOverlap(job) And totalValue > _valueList.get(endIndex)
Branch coverage	!_focusList.get(endIndex).doesOverlap(job) And totalValue > _valueList.get(endIndex)	_focusList.get(endIndex).doesOverlap(job) And totalValue > _valueList.get(endIndex)
Branch coverage	!_focusList.get(endIndex).doesOverlap(job) And totalValue > _valueList.get(endIndex)	!_focusList.get(endIndex).doesOverlap(job) And !totalValue > _valueList.get(endIndex)
Branch coverage	!_focusList.get(endIndex).doesOverlap(job) And totalValue > _valueList.get(endIndex)	_focusList.get(endIndex).doesOverlap(job) And !totalValue > _valueList.get(endIndex)
Boundary Coverage	totalValue > _valueList.get(endIndex)	totalValue > _valueList.get(endIndex)
Boundary Coverage	totalValue > _valueList.get(endIndex)	totalValue < _valueList.get(endIndex)
Boundary Coverage	totalValue > _valueList.get(endIndex)	totalValue == _valueList.get(endIndex)

Single Loop

Test Condition	Condition Satisfied	Assertion

End index = 5
S1 = {X1, X2, X3, X4, X5, X6}
S2 = {1, 2, 1, 1, 2}
{Start, End, Value}
X1 = (1, 2, 1)
X2 = (1, 2, 2)
X3 = (1, 3, 1)
X4 = (1, 4, 1)
X5 = (1, 4, 2)
X6 = (4, 5, 2)
	CC, B1, B2, B3, b1, b2, b3	

maxValueIndex

Conditions

Goal	Notes	Condition
Code coverage	All conditions true	_valueList[i] > max
Branch coverage	_valueList.get(i) > max is false	_valueList[i] <= max
Boundary coverage	_valueList.get(i) > max is false	_valueList[i] < max
Boundary coverage	_valueList.get(i) > max is false	_valueList[i] == max

Single Loop

Test Condition	Condition Satisfied	Assertion

S = {X1, X2, X3, X4}
X1 = 1, X2 = 2, X3 = 2, X4 = 1
	CC, B1, b1, b2	assertEquals 1

optimalSchedule

Conditions

Goal	Notes	Condition
Code coverage	All conditions true	First iteration link != null
Branch coverage	While condition false	Link.getLinkedObject = null

Single Loop

Test Condition	Condition Satisfied	Assertion
X1 linked to X2 X2 linked to null X1, X2 valid inputs	CC, B1	List = X2, X1

weightedJobSchedule

Conditions

Goal	Notes	Condition
Code coverage	All conditions true	sortedSet is valid, size > 0
Branch Coverage	Error checking	_sortedSet = null
Branch Coverage	Error checking	_sortedSet.size() <= 0
Boundary	_sortedSet.size() > 0	_sortedSet.size > 0
Boundary	_sortedSet.size() > 0	_sortedSet.size <= 0

Single Loop

Test Condition	Condition Satisfied	Assertion
S = {X1}	CC, b1	= {X1}
S = {}	B2, b2	null
S = null	B1	NullPointerException

Stress Test

1. Create a PhotoSchedule object with an empty list
2. Create 10,000 new PhotoTimes and add them to the schedule
    a. Each PhotoTime will have a name i
    b. Each PhotoTime will have a new random number between 0-100 inclusive for start
    c. Each PhotoTime will have a new random number between 0-100 inclusive for end
    d. Each PhotoTime will have a new random number between 0-100 inclusive for priority
3. Because a start time must be before the end time, the probability that the PhotoTime successfully adds to the PhotoSchedule is 48.5% (derived from generating 10,000 lists)
4. The size of the array in the PhotoSchedule should be between size 4600 and 5100
5. The average length of a solution over a list of size 4850 is 37 (generated by testing 1000 runs)
6. The size of the solution should be between 32 and 42