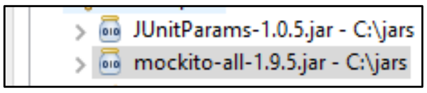
UECS2354 SOFTWARE TESTING
LAB TEST

MARKING SCHEME **JAN 2020**

Setup [2 marks]

1. Place jars file in desired location and build path is configured accordingly. **[2 marks]**



JUnitParams-1.0.5.jar - C:\jars
mockito-all-1.9.5.jar - C:\jars

Part 1 **[20 marks]**

1. Correctly implemented exception handling. (4 marks)

Criteria	**Mark**
Invalid age (should be less than 0)	1
Invalid month (should be 0 and less; 13 and more)	2
Any error handling method will be accepted	1

Sample code:

class: AirlineTicketingSystem
method: getTicketPrice()

```
            if(age < 0 || month < 1 || month > 12) // 3 marks
                  throw new IllegalArgumentException();   // 1 mark
```

2. Using correct test data based on Boundary Value Analysis. (7 marks)

Criteria	**Mark**
Should have 21 combination of test data (refer to the BVA table)	7
Wrong expected result or expected result which is not predetermine beforehand	Minus 2M

Sample code:

class: TestAirlineTicketingSystem

```
// 7 marks
            private Object[] getParamTestTicketPrice() {
                  return new Object[] {
                              new Object[] {100,0,1, 85},
                              new Object[] {100,0,11,85},
                              new Object[] {100,0,12,100},
                              new Object[] {100,6,1,85},
                              new Object[] {100,6,11,85},
                              new Object[] {100,6,12,100},
                              new Object[] {100,7,1,90},
                              new Object[] {100,7,11,90},
                              new Object[] {100,7,12,110},
                              new Object[] {100,18,1,90},
                              new Object[] {100,18,11,90},
                              new Object[] {100,18,12,110},
                              new Object[] {100,19,1,100},
                              new Object[] {100,19,11,100},
                              new Object[] {100,19,12,125},
                              new Object[] {100,55,1,100},
                              new Object[] {100,55,11,100},
                              new Object[] {100,55,12,125},
                              new Object[] {100,56,1,90},
                              new Object[] {100,56,11,90},
                              new Object[] {100,56,12,115}
                 };
        }
```

3. Correctly implemented test method for valid cases. (4 marks)

Criteria	**Mark**
Correct use of @Parameters - Can use any parameterized test approach	1
Correctly implemented test method	3
Wrong assert method, wrong use of delta	Minus 0.5 each mistake
Test error or failure	Minus 1

Sample code:

class: TestAirlineTicketingSystem

```
@Test
// 1 mark
@Parameters(method="getParamTestTicketPrice")
public void testGetTicketPrice(double normalPrice, int age, int month, double
expectedPrice) {
             // 1 mark
             AirlineTicketingSystem ATS = new AirlineTicketingSystem();

             // 2 marks assert method
             double actualPrice = ATS.getTicketPrice(normalPrice, age, month);
             assertEquals(expectedPrice, actualPrice,0.001);
}
```

4. Correctly implemented test method for invalid cases. (5 marks)

Criteria	**Mark**
Correct use of @Parameters - Can use any parameterized test approach Correct test data (refer to BVA table)	4
Correctly implemented test method for invalid value	1
Test error or failure	Minus 1

Sample code:

class: TestAirlineTicketingSystem

```
// 0.5 mark for expected=
@Test(expected=IllegalArgumentException.class)
// 4 marks for invalid cases
@Parameters(method="getParamIllegal")
public void testInvalidValues(double normalPrice, int age, int month) {
       // 0.5 mark
       AirlineTicketingSystem ATS = new AirlineTicketingSystem();
       ATS.getTicketPrice(normalPrice, age, month);
}
```

```
	private Object[] getParamIllegal(){
		return new Object[]{
			// month < 0
			new Object[] {100, -1, -1},
			new Object[] {100, 0, -1},
			new Object[] {100, 6, -1},
			new Object[] {100, 7, -1},
			new Object[] {100, 18, -1},
			new Object[] {100, 19, -1},
			new Object[] {100, 55, -1},
			new Object[] {100, 56, -1},
			// month = 1-11
			new Object[] {100, -1, 1},
			new Object[] {100, -1, 11},
			// month = 12
			new Object[] {100, -1, 12},
			// month >12
			new Object[] {100, -1, 13},
			new Object[] {100, 0, 13},
			new Object[] {100, 6, 13},
			new Object[] {100, 7, 13},
			new Object[] {100, 18, 13},
			new Object[] {100, 19, 13},
			new Object[] {100, 55, 13},
			new Object[] {100, 56, 13}
		};
	}
```

Part 2 **[18 marks]**

1. Correctly implemented constructors. (3 marks)

Criteria	**Mark**
Correctly implemented constructors (this will affect the test code implementation)	3
Missing or incomplete constructors	Minus 0.5

Sample code

class: AirlineTicketingSystem

```
// constructors
	// 1 mark
	public AirlineTicketingSystem(){
		tr = new Traveler();
	}

	// 2 marks
	public AirlineTicketingSystem(Traveler tr) {
		this.tr=tr;
	}
```

2. Correctly implemented exception handling. (2 marks)

Criteria	**Mark**
Invalid point (should be less than 0)	1
Any error handling method will be accepted	1

Sample code:

class: AirlineTicketingSystem
method: updateCategory()

```
// 2 marks
      if (point<0)
            throw new IllegalArgumentException();
```

3. Correctly implemented test method for getReward(). (5 marks)

Criteria	**Mark**
Correct use of @Parameters - Can use any parameterized test approach Correct test data	1
Correctly implemented test method	4
Missing Mockito methods	Minus 1 each
wrongly implemented Mockito methods	Minus 0.5
Wrong assert method	Minus 0.5
Test error or failures	Minus 1
No test double (stub)	Minus 2

Sample code:

class: TestAirlineTicketingSystem

```
// Methods to test getReward() and updateCategory()
@Test
// 1 mark
@Parameters({"Peter,Normal Flyer,Free meal upgrade","Peter,Seasonal Flyer,Free
seat upgrade","Peter,Frequent Flyer,Free upgrade to business class"})
public void testGetReward(String username, String category, String expected) {
      // 2 marks
      Traveler trmock = mock(Traveler.class);
      when(trmock.initializeTraveler(anyString())).thenReturn(category);

      // 2 marks
      AirlineTicketingSystem ATS = new AirlineTicketingSystem(trmock);
      ATS.startBooking(username);
      assertEquals(expected, ATS.getReward());
            }
```

UECS2354 SOFTWARE TESTING
LAB TEST

MARKING SCHEME **JAN 2020**

4. Correctly implemented test method for updateCategory(). (8 marks)

Criteria	**Mark**
Correct use of @Parameters - Can use any parameterized test approach Correct test data, Refer to BVA table	1
Correctly implemented test method for valid – with test double	5
Correctly implemented test method for invalid value – with test double	2
Missing Mockito methods or no verify() method	Minus 1 each
wrongly implemented Mockito methods	Minus 0.5
Test error or failures	Minus 1
No test double (stub)	Minus 2

Sample code:

class: TestAirlineTicketingSystem

```
@Test
// 1 marks
@Parameters({"0,Normal Flyer","1000,Normal Flyer","1001,Seasonal
Flyer","50000,Seasonal Flyer","50001,Frequent Flyer"})
public void testUpdateCategory(double point, String expected) {
	// 2 marks
	Traveler trmock = mock(Traveler.class);
	when(trmock.getPoint()).thenReturn(point);
	// 1 marks
	AirlineTicketingSystem ATS = new AirlineTicketingSystem(trmock);
	ATS.startBooking("Peter");
	ATS.updateCategory();
	// 2 marks
	verify(trmock).updateCategory(expected);
}

	// 2 marks
	@Test(expected=IllegalArgumentException.class)
	public void TestInvalidUpdateCategory() {
	Traveler trmock = mock(Traveler.class);
	when(trmock.getPoint()).thenReturn(-1.0);

	AirlineTicketingSystem ATS = new AirlineTicketingSystem(trmock);
	ATS.startBooking("Peter");
	ATS.updateCategory();
}
```

UECS2354 SOFTWARE TESTING

LAB TEST

MARKING SCHEME **JAN 2020**

BVA – getTicketPrice()

Month	← 0					1 - 11					12					13 →				
	(0)					(1, 11)	(1, 11)	(1, 11)	(1, 11)	(1, 11)	(12)	(12)	(12)	(12)	(12)	(13)				
Age	← -1	0-6	7-18	19-55	>56	← -1	0-6	7-18	19-55	>56	← -1	0-6	7-18	19-55	>56	← -1	0-6	7-18	19-55	>56
	(-1)	(0,6)	(7,18)	(19,55)	(56)	(-1)	(0,6)	(7,18)	(19,55)	(56)	(-1)	(0,6)	(7,18)	(19,55)	(56)	(-1)	(0,6)	(7,18)	(19,55)	(56)
NP							*100*	*100*	*100*	*100*		*100*	*100*	*100*	*100*					
FP	INV					INV	NP*0.85	NP*0.9	NP	NP*0.9	INV	NP	NP*1.1	NP*1.25	NP*1.15	INV				
ER							85	90	100	90		100	110	125	115					

BVA – updateCategory()

Range	← -1	0 -1000	1001 – 50,000	50001 →
TD	(-1)	(0,1000)	(1001, 50000)	(50001)
ER	INV	Normal Flyer	Seasonal Flyer	Frequent Flyer