Test case design - study point exercise

Program sourcecode: https://github.com/bigstepdenmark/testCaseDesignExercise

Af Ismail Cam

Equivalence Partitioning

1. Make equivalences classes for the input variable for this method that accepts the numbers 1 - 1000:

```
boolean isEven(int n)
{
    return n > 0 && n <= 1000 && n % 2 == 0;
}
```

```
/**
 * Test the given integer from 1-1000 is even.
 */
@org.junit.Test
public void isEven() throws Exception
{
    /*********************************
     * EQUIVALENCE PARTITIONING
     *********************************/
    // First invalid partition
    assertEquals( false, ctrl.isEven( 0 ) );

    // Valid partition
    assertEquals( true, ctrl.isEven( 500 ) );

    // Last invalid partition
    assertEquals( false, ctrl.isEven( 1001 ) );
}
```

Partitions	Input	Expected	Result
First invalid partition	0	false	false
Valid partition	500	true	true
Last invalid partition	1001	false	false

2. Make equivalences classes for an input variable that represents a mortgage applicant's salary. The valid range is $1,000 pr. month to $75,000 pr. month

```
boolean checkSalary(int amount)
{
    return amount >= 1000 && amount <= 75000;
}
```

```
/**
 * Test the given amount is valid.
 */
@org.junit.Test
public void checkSalary() throws Exception
{
    /**********************************
     * EQUIVALENCE PARTITIONING
     **********************************/
    // First invalid partition
    assertEquals( false, ctrl.checkSalary( 0 ) );

    // Valid partition
    assertEquals( true, ctrl.checkSalary( 1000 ) );

    // Last invalid partition
    assertEquals( false, ctrl.checkSalary( 75001 ) );
}
```

Partitions	Input	Expected	Result
First invalid partition	0	false	false
Valid partition	1000	true	true
Last invalid partition	75001	false	false

3. Make equivalences classes for the input variables for this method:

```
static int getNumDaysinMonth(int month, int year)
{
    try
    {
        YearMonth ym = YearMonth.of( year, month );
        return ym.lengthOfMonth();
    }
    catch( DateTimeException ex )
    {
        return 0;
    }
}
```

```
/**
 * Test the number of days in a specific month and year.
 */
@org.junit.Test
public void getNumDaysinMonth() throws Exception
{
    /***********************************
     * EQUIVALENCE PARTITIONING
     ***********************************/
    // First invalid partition
    assertSame( 0, Controller.getNumDaysinMonth( 0,0 ) );

    // Valid partition
    assertSame( 31, Controller.getNumDaysinMonth( 1,2017 ) );

    // Last invalid partition
    assertSame( 0, Controller.getNumDaysinMonth( 13,100000 ) );
}
```

Partitions	Input	Expected	Result
First invalid partition	0, 0	0	0
Valid partition	1, 2017	31	31
Last invalid partition	13, 100000	0	0

Boundary Value Analysis

1. Do boundary value analysis for equivalence partitioning exercise 1

```java
/**
 * Test the given integer from 1-1000 is even.
 */
@org.junit.Test
public void isEven() throws Exception
{
    /*********************************
     * BOUNDARY VALUE ANALYSIS
     *********************************/
    // First invalid partition
    assertEquals( false, ctrl.isEven( 0 ) );

    // Valid partition
    assertEquals( false, ctrl.isEven( 1 ) );
    assertEquals( true, ctrl.isEven( 1000 ) );

    // Last invalid partition
    assertEquals( false, ctrl.isEven( 1001 ) );
}
```

Partitions	Input	Expected	Result
First invalid partition	0	false	false
Valid partition (min)	1	false	false
Valid partition (max)	1000	true	true
Last invalid partition	1001	false	false

2. Do boundary value analysis for equivalence partitioning exercise 2

```java
/**
 * Test the given amount is valid.
 */
@org.junit.Test
public void checkSalary() throws Exception
{
    /**********************************
     * BOUNDARY VALUE ANALYSIS
     **********************************/
    // First invalid partition
    assertEquals( false, ctrl.checkSalary( 0 ) );

    // Valid partition
    assertEquals( true, ctrl.checkSalary( 1000 ) );
    assertEquals( true, ctrl.checkSalary( 75000 ) );

    // Last invalid partition
    assertEquals( false, ctrl.checkSalary( 75001 ) );
}
```

Partitions	Input	Expected	Result
First invalid partition	0	false	false
Valid partition (min)	1000	true	true
Valid partition (max)	75000	true	true
Last invalid partition	75001	false	false

3. Do boundary value analysis for equivalence partitioning exercise 3

```java
/**
 * Test the number of days in a specific month and year.
 */
@org.junit.Test
public void getNumDaysinMonth() throws Exception
{
    /***********************************
     * BOUNDARY VALUE ANALYSIS
     ***********************************/
    // First invalid partition
    assertSame( 0, Controller.getNumDaysinMonth( 0,0 ) );

    // Valid partition
    assertSame( 31, Controller.getNumDaysinMonth( 1,1975 ) );
    assertSame( 31, Controller.getNumDaysinMonth( 12,2017 ) );

    // Last invalid partition
    assertSame( 0, Controller.getNumDaysinMonth( 13,100000 ) );
}
```

Partitions	Input	Expected	Result
First invalid partition	0, 0	0	0
Valid partition (min)	1, 1975	31	31
Valid partition (max)	12, 2017	31	31
Last invalid partition	13, 100000	0	0

Decision tables

1. Make a decision table for the following business case:
   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								
Deductible has been meet	F	F	F	F	T	T	T	T
Actions								
Doctor's office visits	F	F	T	T	F	F	T	T
Hospital visits	F	T	F	T	F	T	F	T
Expected results								
Reimbursed	0 %	0 %	0 %	0 %	0 %	80 %	50 %	0 %

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								
Divisible by 4	F	F	F	F	T	T	T	T
Divisible by 100	F	F	T	T	F	F	T	T
Divisible by 400	F	T	F	T	F	T	F	T
Actions								
Leap year	No	Yes	No	Yes	Yes	Yes	No	Yes