4. Avaliación

4.1 Criterios de avaliación

Criterios de avaliación seleccionados para esta actividade	Instrumento de avaliación	Peso cualificación (%)
▪ CA3.2 - Definíronse casos de proba.	▪ PE.2 - Proba escrita práctica de definición de casos de proba utilizando técnicas de caixa branca e negra.	9
▪ CA3.6 - Efectuáronse probas unitarias de clases e funcións.	▪ PE.6 - Proba escrita práctica de xeración de probas unitarias no contorno de desenvolvemento.	28
▪ CA3.7 - Executáronse probas automáticas.	▪ PE.7 - Proba escrita práctica de execución automática de probas unitarias no contorno de desenvolvemento.	19
▪ CA3.8 - Documentáronse as incidencias detectadas.	▪ PE.8 - Proba escrita práctica de interpretación de resultados da execución de probas unitarias e documentación de incidencias detectadas.	8

4.2 Exemplo de proba escrita

Introdución

Proponse unha proba escrita combinada cos instrumentos de avaliación PE. 2, PE. 6, PE. 7 e PE.8. A correspondencia dos distintos apartados da proba cós criterios de avaliación é a seguinte:

- CA3.2: Apartado a.
- CA3.6: Apartado b.
- CA3.7: Apartado c.
- CA3.8: Apartado c.

O alumnado realizará o exame nunha máquina virtual que ten instalado o software necesario. O alumno irá colocando nun documento as imaxes ou texto correspondentes a cada pregunta. Ao finalizar a proba, entregará o proxecto NetBeans coa carpeta que conteña a proba JUnit e o documento impreso que firmara e paxinará convenientemente; a máquina virtual quedará no ordenador coa sesión pechada para posibles consultas do profesorado.

Texto da proba

Utilizarase a clase Java *Divisores* do proxecto NetBeans *divisores*. O proxecto está composto pola clase *Divisores* e a clase *Main*. A clase *Divisiores* ten o método *obterDivisores* que recibe un número de tipo byte e devolve unha cadea formada polos divisores do número separados por un espazo en branco. A clase *Main* permite teclear un número de tipo byte e visualiza os divisores dese número que obtén da clase *Divisores*. Información que pode influír na definición das probas:

- O tipo *byte* admite os valores: -128 a 127.
- Tódolos números enteiros maiores que 0 teñen como mínimo dous divisores: 1 e o propio número agás o 1 que só ten como divisor:1. Os números primos teñen como divisores só o 1 e o propio número, como por exemplo 127.

Realizar a proba da clase *Divisores* seguindo os pasos:

a) Definir os casos de proba indicando para cada un: identificación do caso, valores de entrada, xustificación da existencia do caso (identificación da clase de equivalencia, análise de valor límite, conxectura de erros, ou camiño que dá lugar a ese caso), e a saída esperada. Entregaranse os casos de proba, grafo de fluxo, cálculos da complexidade ciclomática de McCabe, definición de camiños, definición de clases de equivalencia, resultados da análise de valores límite e resultados da conxectura de erros.

b) Xerar a proba mediante JUnit e utilizando NetBeans 8.0.1.

c) Executar a proba. Entregar resultado da execución e documentación das incidencias encontradas.

Solución

Pregunta a)

Probas de caixa negra. Clases de equivalencia:

Entrada	Clases válidas	Clases non válidas
byte n	(1) 0<Numero enteiro<=127	(2) Número enteiro <=0

Probas de caixa negra. Análise do valor límite:

AVL1: 0

AVL2: 1

AVL3: 127. Número primo. O bucle repítese o número máximo de veces.

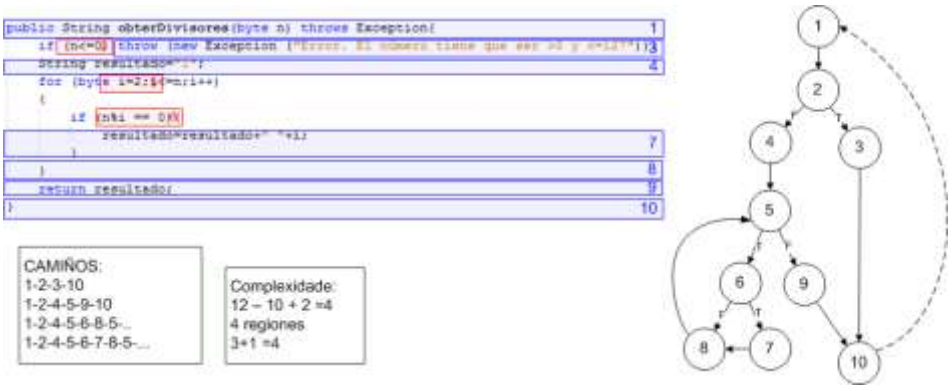
Probas de caixa negra. Conxectura de erros:

CE1: 1 que só ten un divisor e fai que o bucle non se execute.

CE2: 2 número primo que só ten dúas divisores e fai que o bucle se repita 1 vez.

CE3: 126 número non primo. O bucle repítese repita 126 veces.

Probas de caixa branca. Grafo de fluxo, complexidade de McCabe e camiños:



Casos de proba:

Identificación do caso de proba	Entrada	Identificación de Clase/camiño/AVL/CE	Saída
	byte n		
C1	1	(1)/AVL2/CE1	"1"
C2	127	AVL3	"1 127"
C3	2	CE2	"1 2"
C4	126	CE3	"1 2 3 6 7 9 14 18 21 42 63 126"

C5	0	(2)/AVL1	Error

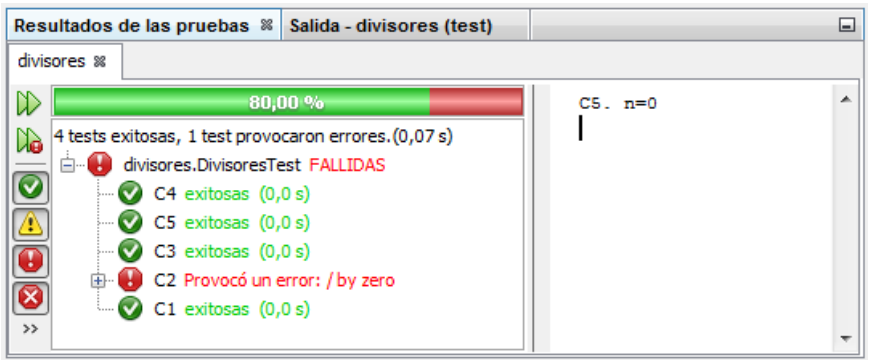
Procedemento de proba: utilizarase JUnit en NetBeans 8.0.1.

```
package divisores;

import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author Profesor
 */
public class DivisoresTest {
    /**
     * Test of obterDivisores method, of class Divisores.
     * @throws java.lang.Exception
     */
    @Test
    public void C1() throws Exception{
        byte n=1;
        Divisores instance = new Divisores();
        String expResult = "1";
        String result = instance.obterDivisores(n);
        assertEquals(expResult, result);
    }
    @Test
    public void C2() throws Exception{
        byte n=127;
        Divisores instance = new Divisores();
        String expResult = "1 127";
        String result = instance.obterDivisores(n);
        assertEquals(expResult, result);
    }
    @Test
    public void C3() throws Exception{
        byte n=2;
        Divisores instance = new Divisores();
        String expResult = "1 2";
        String result = instance.obterDivisores(n);
        assertEquals(expResult, result);
    }
    @Test
    public void C4() throws Exception{
        byte n=126;
        Divisores instance = new Divisores();
        String expResult = "1 2 3 6 7 9 14 18 21 42 63 126";
        String result = instance.obterDivisores(n);
        assertEquals(expResult, result);
    }
    @Test(expected = Exception.class)
    public void C5() throws Exception{
        byte n=0;
        System.out.println("C5. n=0");
        Divisores instance = new Divisores();
        String result = instance.obterDivisores(n);
    }
}
```

Execución, avaliación e resumo da proba:



O caso C2 dá o erro de división por 0 xa que a variable i do bucle é de tipo byte e eso fai que cando i vale 127 e se incrementa unha unidade, pase a valer 128, é dicir, 0; ao continuar no bucle divídese por 0.