Pràctica de proves d'optimització

José Carlos Salvador Gomes
Benjamín Caleb Huanca Quiroga
Agustín Fontaiña Santarelli

1. Pila

1.1. Caixa blanca

En aquest programa no farem cap prova de caixa blanca ja que els mètodes no hem utilitzat cap bucle, iteració, o condició.

1.2. Caixa negra

En aquest programa farem els següents casos de prova pel codi:

-Classes d'equivalència:

A cap dels mètodes d'aquest programa el valor numèric introduit afecta a l'operació efectuada. Per tant només existeix una classe d'equivalència per a cada mètode.

Push	-> testPush
Pop	-> testPop
isEmpty	-> testIsEmptyFalse, testIsEmptyTrue
Top	-> testTop

-Anàlisi dels valors límit:

Degut a que no hi ha classes d'equivalència no hi haurà un estudi de valors inicials i finals. Només comprovarem introduint un nombre molt gran, però ja que aquest cas també és un error típic no ho farem aquí.

-Estudi d'errors típics:

En el mètode "Push" els errors típics que provarem seran els següents:
·Introduir un caràcter diferent a un nombre (No compila):
Utilitzant testPush introduint un caràcter no numèric, o un nombre amb decimals i el propi netbeans ja no ho deixa compilar.

·No introduir cap valor. (No compila):
Utilitzant testPush introduint el valor null el programa ja no compila.

·Introduir un valor molt gran. (No compila):
Utilitzant testPush introduint un valor molt alt, el tipus de variable primitiu int, ja no ho compila.

·Utilitzar el mètode *pop* quan la pila es buida (Corregit).
Per corregirlo hem hagut d'afegir l'excepció al métode. I l'hem provat amb el mètode *popIsEmpty*.

·Utilitzar el mètode *top* quan la pila es buida (Corregit).
Per corregirlo hem hagut d'afegir l'excepció al métode.I l'hem provat amb el mètode *topIsEmpty*.

-Maneig d'interfície gràfica:

Ja que aquest programa no disposa d'una interfície gràfica no realitzarem cap cas de prova.

-Dades aleatòries:

En aquest programa hauríem d'utilitzar un programa que generi casos de prova amb dades aleatòries.

2. Ordenar dos arrays

2.1.Caixa blanca

Pel mètode "calculaSalarioBruto":

-Una de les llistes estigui buida o les dues estiguin buides: ***testSumOrdCB1***
Cas utilitzat : [] [4,5,6]

-Una de les llistes superi el màxim de longitud de l'array: ***testSumOrdCB2***
Cas utilitzat : [5,7,13,34,45,48,89] [4,5,6]

-La primera llista no està ordenada: ***testSumOrdCB3***
Cas utilitzat : [5,1,-7] [4,5,6]

-La segona llista no està ordenada: ***testSumOrdCB4***
Cas utilitzat : [4,5,6] [5,1,-7]

-El primer valor de la primera llista es menor que el primer valor de la segona llista i no és l'últim valor de la primera llista: ***testSumOrdCB5***
Cas utilitzat : [-7, 8, 16] [-6, 10, 15]

-El primer valor de la primera llista es menor que el primer valor de la segona llista i és l'últim valor de la primera llista: ***testSumOrdCB6***
Cas utilitzat : [-7] [-6, 10, 15]

-El primer valor de la primera llista es mayor que el primer valor de la segona llista i no és l'últim valor de la primera llista: ***testSumOrdCB7***
Cas utilitzat : [5, 10] [4, 6]

-El primer valor de la primera llista es mayor que el primer valor de la segona llista i és l'últim valor de la primera llista: ***testSumOrdCB8***
Cas utilitzat : [5, 10] [4]

2.2. Caixa negra

En aquest programa farem els següents casos de prova pel codi:

-Classes d'equivalència:

1. Introduir dues llistes ordenades: ***testSumOrdEquiv1()***
   Llista1 -> (1, 2, 3) Llista2 -> (4, 5, 6)

2. Introduir dues llistes ordenades pero invertides: ***testSumOrdEquiv2()***
   Llista1 -> (4, 5, 6) Llista2 -> (1, 2, 3)

3. Introduir dues llistes amb els valors desordenats: ***testSumOrdEquiv3()***
   Llista1 -> (1, 12, 43) Llista2 -> (1, 5, 36)

-Anàlisi dels valors límit:

El valor límit serà un número molt gran: El programa **no compila** ja que el tipus de variable del tipus int no permet números molt grans.

-Estudi d'errors típics:

Els errors típics que provarem seran els següents:

·No introduir cap valor en una llista (Corregit): ***testSumOrdErrT1***
Hem afegit al codi una comprovació de que els arrays tinguin contingut. Ara ens dona l'error "*Exception*".

·No introduir cap valor en cap de les dues llistes (Corregit): ***testSumOrdErrT2***
La correcció feta per l'error anterior ha corregit aquest també.

·Introduir un valor molt gran (No compila):
Hem provat a introduir un valor molt alt, i el tipus de variable primitiu int ja no ho compila.

·Introduir un valor decimal o un altre valor no vàlid (No compila):
Hem provat introduint un valor com per exemple "A" i ha donat, i el programa no ens ha permès compilar.

·Que les dues llistes siguin de diferent longitud: ***testSumOrdErrT3()***

·Introduir un valor negatiu: ***testSumOrdErrT4()***

·Introduir valors desordenats dins una llista (Corregit): ***testSumOrdErrT5()***
Hem afegit al codi una comprovació de que els arrays estiguin ordenats per arreglar aquest error. Ara ens dona l'error "*Exception*".

·Introduir més valors que el número màxim de valors en una mateixa llista (Corregit): ***testSumOrdErrT6()***

Hem afegit al codi una comprovació de que els arrays no sobrepasen la llargada màxima establerta. Ara ens dona l'error "*Exception*".

-Maneig d'interfície gràfica:

Ja que aquest programa no disposa d'una interfície gràfica no realitzarem cap cas de prova.

-Dades aleatòries:

En aquest programa hauríem d'utilitzar un programa que generi casos de prova amb dades aleatòries.

3. Coa

3.1.Caixa blanca

En aquest programa no farem cap prova de caixa blanca ja que els mètodes no hem utilitzat cap bucle, iteració, o condició.

3.2. Caixa negra

En aquest programa farem els següents casos de prova pel codi:

-Classes d'equivalència:

A cap dels mètodes d'aquest programa el valor numèric introduit afecta a l'operació efectuada. Per tant només existeix una classe d'equivalència per a cada mètode.

Push	-> testPush
Pop	-> testPop
isEmpty	-> testIsEmptyFalse, testIsEmptyTrue
Top	-> testTop

-Anàlisi dels valors límit:

Degut a que no hi ha classes d'equivalència no hi haurà un estudi de valors inicials i finals. Només comprovarem introduint un nombre molt gran, però ja que aquest cas també és un error típic no ho farem aquí.

-Estudi d'errors típics:

En el mètode "Push" els errors típics que provarem seran els següents:

·Introduir un caràcter diferent a un nombre (No compila):

Utilitzant testPush introduint un caràcter no numèric, o un nombre amb decimals i el propi netbeans ja no ho deixa compilar.

·No introduir cap valor. (No compila):

Utilitzant testPush introduint el valor null el programa ja no compila.

·Introduir un valor molt gran. (No compila):

Utilitzant testPush introduint un valor molt alt, el tipus de variable primitiu int, ja no ho compila.

·Utilitzar el mètode *pop* quan la pila es buida (Corregit).

Per corregirlo hem hagut d'afegir l'excepció al métode. I l'hem provat amb el mètode *popIsEmpty*.

·Utilitzar el mètode *top* quan la pila es buida (Corregit).

Per corregirlo hem hagut d'afegir l'excepció al métode.I l'hem provat amb el mètode *topIsEmpty*.

-Maneig d'interfície gràfica:

Ja que aquest programa no disposa d'una interfície gràfica no realitzarem cap cas de prova.

-Dades aleatòries:

En aquest programa hauríem d'utilitzar un programa que generi casos de prova amb dades aleatòries.

4. Calcular salari net / brut

4.1.Caixa blanca

Per aquest programa haurem de fer les proves de caixa blanca dels dos mètodes que conformen la classe.

Pel mètode "calculaSalarioBruto":

Dins aquest mètode tenim 5 ifs. Hem escrit en binari cada possibilitat, i aquests són els valors que hem introduït per cada prova:

	TipoEmpleado	VentasMes	HorasExtra	Mètode Utilitzat
00100:	Encargado	900	0	testSalBrutoCB1
00101:	Encargado	1875	0	testSalBrutoCB2
00110:	Encargado	1333	0	testSalBrutoCB3
01000:	Vendedor	900	0	testSalBrutoCB4
01001:	Vendedor	1875	0	testSalBrutoCB5
01010:	Vendedor	1333	0	testSalBrutoCB6
1:	null	-1	-1	testSalBrutoCB7

Pel mètode "calculaSalarioNeto":

Dins aquest mètode tenim 3 ifs. Hem escrit en binari cada possibilitat, i aquests són els valors que hem introduït per cada prova:

	SalarioBruto	Mètode Utilitzat
000:	1900	testSalNetoCB1
001:	1333	testSalNetoCB2
010:	900	testSalNetoCB3
1:	-1	testSalNetoCB4

4.2. Caixa negra

En aquest programa farem els següents casos de prova per cada un dels mètodes:

Pel mètode "calculaSalarioBruto":

-Classes d'equivalència:

-TipoEmpleado -> -*"Vendedor"* i *"encargado"*. ***testSalBrutoEquiv1***
-VentasMes ->
-No es sumarà res en el cas de que les ventes siguin menors a 1000 euros.
-Es sumaran 100 euros al Salario si ventasMes és major o igual que 1000 euros.
- Es sumarà 200 euros al Salario si ventasMes és major o igual que 1500 euros.
-HorasExtra -> -Nombres sencers positius.

-Anàlisi dels valors límit:

VentasMes:

- Per provar els casos dels límits 0 i 999,99 hem utilitzat:
testSalBrutoVL1 (Per comprovar amb 0).
testSalBrutoVL2 (Per comprovar amb 999.99).

- Per provar els casos dels límits 1000 i 1499,99 hem utilitzat:
testSalBrutoVL3(Per comprovar amb 1000).
testSalBrutoVL4 (Per comprovar amb 1499.99).

- Provarem a utilitzar 1500 per comprovar el límit entre 1500 i un nombre major a 1500 que no superi la capacitat del float. Aquest darrer cas no compila.
testSalBrutoVLNegatiu5 (Per comprovar amb 1500).

HorasExtra:

- Un nombre molt alt (No compila).
- El nombre 0. ***testSalBrutoVLZero***

-Estudi d'errors típics:

-Posar un número amb més de 6 decimals (No compila).
-Introduir un caràcter diferent a un nombre (No compila).
-No introduir cap valor (No compila).
-Posar un número molt gran (No compila).
-Introduir un valor negatiu: Amb el mètode ***testSalBrutoETNegatiu1*** i amb el mètode ***testSalBrutoETNegatiu2*** s'executarà l'excepció BRExcpetion.

-Maneig d'interfície gràfica:

Ja que aquest programa no disposa d'una interfície gràfica no realitzarem cap cas de prova.

-Dades aleatòries:

En aquest programa hauríem d'utilitzar un programa que generi casos de prova amb dades aleatòries.

Pel mètode "calculaSalarioNeto":

-Classes d'equivalència:

Salaris: 0 <= x < 1000
1000 <= x < 1500 -> 16%.
x >=1500 -> 18%.

-Anàlisi dels valors límit:

- Per provar els casos dels límits 0 i 999,99 hem utilitzat:
testSalNetoVL1(Per comprovar amb 0).
testSalNetoVL2(Per comprovar amb 999.99).

- Per provar els casos dels límits 1000 i 1499,99 hem utilitzat:
testSalNetoVL3(Per comprovar amb 1000).
testSalNetoVL4(Per comprovar amb 1499.99)

- Provarem a utilitzar 1500 per comprovar el límit entre 1500 i un nombre major a 1500 que no superi la capacitat del float. Aquest darrer cas no compila.
testSalNetoVL5(Per comprovar amb 1500).

-Estudi d'errors típics:

-Posar un número amb més de 6 decimals. (No compila).
-Introduir un caràcter diferent a un nombre. (No compila).
-No introduir cap valor. (No compila).
-Posar un número molt gran. (No compila).
-Introduir un valor negatiu: Per fer la prova d'aquest cas hem introduït el valor -1 amb el mètode ***testSalNetoETNegatiu1.***

-Maneig d'interfície gràfica:

Ja que aquest programa no disposa d'una interfície gràfica no realitzarem cap cas de prova.

-Dades aleatòries:

En aquest programa hauríem d'utilitzar un programa que generi casos de prova amb dades aleatòries.