La pila, la coa i la llista estàn en la mateixa classe pila.
Pila

Caixa Blanca

Prove de caixa blanca sols hi ha un camí, per tant queden recollides amb les de caixa negra

Caixa Negre

- **Classe d'equivalències:** sols tendren una sola clase i agafam el valor 5 .

- **Errors típics:**
  - **Null**

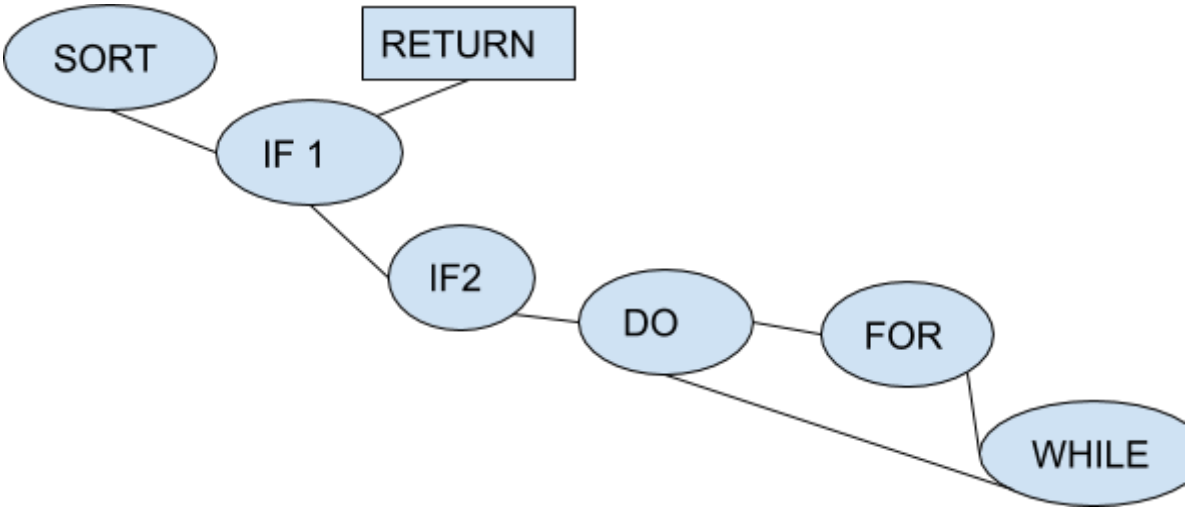
    (testpushnull) funciona
    (testpopnull): funciona
    (testTopNull): funciona
    (TestTopEmpty): return 'Is empty' quan no hi ha res
    TestPopEmpty: si no hi ha res, trona una excepció ArrayIndexOut'

  - **Valor molt alt:** (testpushlong) quan es defineix amb valor '9999999999' long de deu o més digits es posara una L al final del valor long i si es un valor integer no hem de posar L.
  - **Valor molt baix:** funciona de la mateixa forma que el valor molt alt
  - **Valor negatiu:** (TestPushNegative) funciona definit amb el valor maxim molt alt

Llista Ordenada

I*mplementar una classe i mètode amb Java que donats dues llistes ordenades torni una tercera llista on* *apareixen els valors de les dues primeres però ordenats. Exemple: llista1: 1,2,6. Llista2: 4,5,8. LLista3:* *1,2,4,5,6,8.* *Implementa els casos de prova amb Junit seguint les tècniques de la caixa blanca.* *Implementa els casos ed prova amb Junit seguint les tècniques de la caixa negra.*

- **Caixa Blanca**



- **Caixa Negra**
  - **Clase de equivalencies**

    Hem creat dos mètodes un que junten els valors i l'altre mètode, ordena de major a menor. En aquesta cas
    - **Juntar**: [5,6] [1,3] = [5,6,1,3]
    - **Ordenar**: [5,3,1,6] = [1,3,5,6]

    Només és per valors integers

  - **Error tipics**

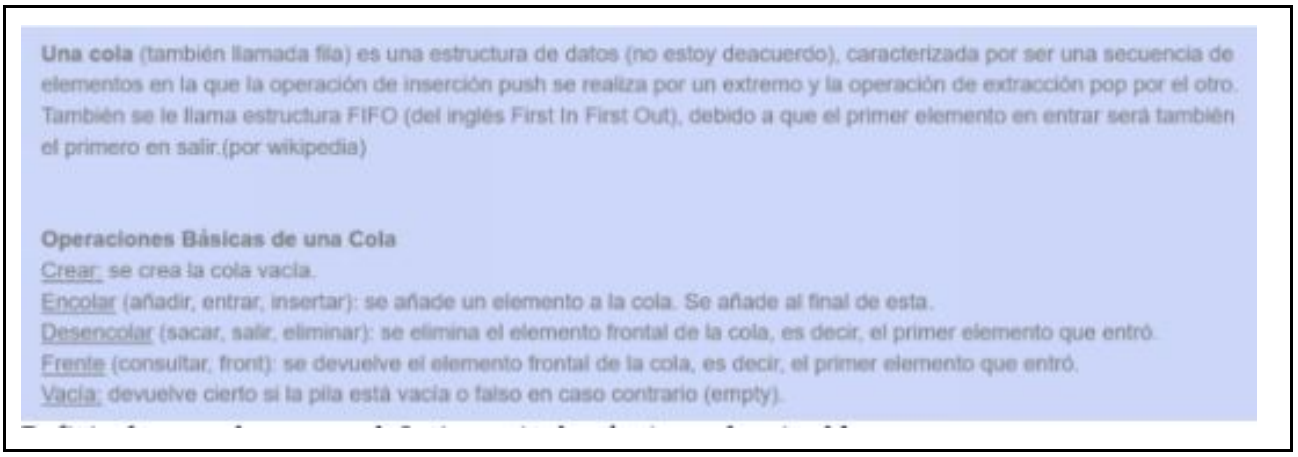
    Juntar:

    testConcat: prova que es junten (en positiu)
    testConcatNull: provam que si es nul, funciona com esperam
    testConcatEqual: provam que amb valors iguals funcionen
    testCobcatNegative: provan que amb valors negatius funcionen

    Ordenar:

    testSortt: prova que es junten (en positiu)
    testSortNull: provam que si es nul, funciona com esperam
    testSortEqual: provam que amb valors iguals funcionen
    testSortNegative: provan que amb valors negatius funcionen

CUA



- **Caixa Blanca**

  Prove de caixa blanca sols hi ha un camí, per tant queden recollides amb les de caixa negra

- **Caixa Negra**
  - **Classes d'equivalencies**

    Hem creat dos metodes un que eliminen i un altre que agafa l' ultim nombre
    - **Shift**: elimina el primer valor [5,6] = [6]
    - **Peek**: ens mostra la posició 0, [5.6] = [5] (es manté el 6)

Error tipics:

testShift: prova que es junten (en positiu)
testShiftNull: provam que si es nul, funciona com esperam
testShiftEqual: provam que amb valors iguals funcionen
testShiftNegative: provan que amb valors negatius funcionen

Ordenar:

testPeek: prova que es junten (en positiu)
testPeekNull: provam que si es nul, funciona com esperam
testPeekEqual: provam que amb valors iguals funcionen
testPeekNegative: provan que amb valors negatius funcionen

Salari

Método	Especificación
`float calculaSalarioBruto( TipoEmpleado tipo, float ventasMes, float horasExtra)`	El salario base será 1000 euros si el empleado es de tipo `TipoEmpleado.vendedor`, y de 1500 euros si es de tipo `TipoVendedor.encargado`. A esta cantidad se le sumará una prima de 100 euros si `ventasMes` es mayor o igual que 1000 euros, y de 200 euros si fuese al menos de 1500 euros. Por último, cada hora extra se pagará a 20 euros. Si `tipo` es `null`, o `ventasMes` o `horasExtra` toman valores negativos el método lanzará una excepción de tipo `BRException`.
`float calculaSalarioNeto( float salarioBruto)`	Si el salario bruto es menor de 1000 euros, no se aplicará ninguna retención. Para salarios a partir de 1000 euros, y menores de 1500 euros se les aplicará un 16%, y a los salarios a partir de 1500 euros se les aplicará un 18%. El método nos devolverá `salarioBruto * (1-retencion)`, o `BRExcepcion` si el salario es menor que cero.

Método a probar	Entrada	Salida esperada
calculaSalarioNeto	2000	1640
calculaSalarioNeto	1500	1230
calculaSalarioNeto	1499.99	1259.9916
calculaSalarioNeto	1250	1050
calculaSalarioNeto	1000	840
calculaSalarioNeto	999.99	999.99
calculaSalarioNeto	500	500
calculaSalarioNeto	0	0
calculaSalarioNeto	-1	BRException
calculaSalarioBruto	vendedor, 2000 euros, 8h	1360
calculaSalarioBruto	vendedor, 1500 euros, 3h	1260
calculaSalarioBruto	vendedor, 1499.99 euros, 0h	1100
calculaSalarioBruto	encargado, 1250 euros, 8h	1760
calculaSalarioBruto	encargado, 1000 euros, 0h	1600
calculaSalarioBruto	encargado, 999.99 euros, 3h	1560
calculaSalarioBruto	encargado, 500 euros, 0h	1500
calculaSalarioBruto	encargado, 0 euros, 8h	1660
calculaSalarioBruto	vendedor, -1 euros, 8h	BRException
calculaSalarioBruto	vendedor, 1500 euros, -1h	BRException
calculaSalarioBruto	null, 1500 euros, 8h	BRException

- **Caixa Blanca**

  Definim dos camins, el mètode Vendedor y el metode Encargado, definits ambs un valor (salari) fixos que pot augmentar depenent de les hores efectuades

- **Caixa Negra**
  - **Classes d'equivalencies**
    - **Valors negatius**
    - **Valors maxims (1200, 1500) Salari**

Error típics:

Venedor

testVenedorNull: provam que si es nul, torna un exception

- testVenedorNegative: provan que amb valors negatius funcionen

Encarregat

testEncarregatNull: provam que si es nul, torna un exception
testEncarregatNegative: provan que amb valors negatius funcionen

Ventes

testVentesNull: provam que si es nul, torna un exception
testVentesNegative: provan que amb valors negatius funcionen

Salario Bruto

testVentesNull: provam que si es nul, torna un exception
testSalarioNegative: torna un exception