Parcial 2 - Teoría - PRG - ETSInf - Curso 2012-13.
7 de junio de 2013. Duración: 2 horas.

1. 2 puntos Suponiendo ya implementada una excepción de usuario llamada `PilaVaciaException` que deriva de la clase `Exception`, es posible definir la clase `PilaIntEnla`, para que cuando se ejecuten los métodos que no pueden operar sobre una Pila vacía lancen la excepción señalada (en lugar de indicar que es una precondición de ejecución de los mismos).

   Se pide: Implementa los métodos de la clase `PilaIntEnla` que necesiten operar sobre una Pila no vacía para que cuando se ejecuten sobre una Pila vacía lancen la excepción `PilaVaciaException`.

Solución:

```
/** Desapila y devuelve el valor en la cima de la Pila */
public int desapilar() throws PilaVaciaException {
    if (talla==0) throw new PilaVaciaException("Pila vacía");
    else {
      int x = this.cima.dato;
      this.cima = this.cima.siguiente;
      this.talla--;
      return x;
    }
}

/** Devuelve el valor en la cima de la Pila */
public int cima() throws PilaVaciaException {
    if (talla==0) throw new PilaVaciaException("Pila vacía");
    return this.cima.dato;
}
```

2. 2.5 puntos Dada una secuencia enlazada de enteros `sec` y un entero `x`, **se pide** un método con el siguiente perfil:

```
public static int ultimaAparicionDe(NodoInt sec, int x)
```

   que devuelva la posición de la última aparición de `x` en la secuencia y -1 si no está, entendiendo 0 como la primera posición.

Solución:

```
public static int ultimaAparicionDe(NodoInt sec, int x) {
   NodoInt actual = sec;
   int contador = 0, ultimaAparicion = -1;
   while(actual!=null) {
         if (actual.dato==x) ultimaAparicion = contador;
         contador++;
         actual = actual.siguiente;
   }
   return ultimaAparicion;
}
```

3. 2.5 puntos Dada una pila de enteros no vacía, **se pide** un método **recursivo** con el siguiente perfil:

```
public static void borraBase(PilaIntEnla p)
```

que la modifique eliminando el elemento de su base (el elemento más antiguo de la pila).

Solución:

```
/** p es una Pila no vacía */
public static void borraBase(PilaIntEnla p) {
   if (p.talla()>1) {
      int x = p.desapilar();
      borraBase(p);
      p.apilar(x);
   }
   else p.desapilar();
}
```

4. 3 puntos Dadas dos listas con punto de interés de enteros, `lista1` y `lista2`, ambas con sus elementos en orden estrictamente creciente, **se pide** un método con el siguiente perfil:

```
/** lista1, lista2 están en orden estrictamente creciente */
public static ListaPIIntEnla union(ListaPIIntEnla lista1, ListaPIIntEnla lista2)
```

que calcule la unión de las dos listas. La lista resultante también deberá quedar en orden estrictamente creciente.

Ejemplo:
Sea una `lista1` con los valores: `1, 3, 5, 7, 9, 11, 13, 15, 17, 19`
Sea una `lista2` con los valores: `1, 4, 7, 10, 13, 16, 19, 21, 27`
El resultado de `union(lista1,lista2)` debe ser una lista con los valores:
`1, 3, 4, 5, 7, 9, 10, 11, 13, 15, 16, 17, 19, 21, 27`

Solución:

```
/** lista1, lista2 están en orden estrictamente creciente */
public static ListaPIIntEnla union(ListaPIIntEnla lista1, ListaPIIntEnla lista2) {
    ListaPIIntEnla li = new ListaPIIntEnla();
    lista1.inicio();  lista2.inicio();
    while(!lista1.esFin() && !lista2.esFin()) {
        int i = lista1.recuperar(), j = lista2.recuperar();
        if (i<j) { li.insertar(i); lista1.siguiente(); }
        else if (i>j) { li.insertar(j); lista2.siguiente(); }
        else { li.insertar(i); lista1.siguiente(); lista2.siguiente(); }
    }

    while(!lista1.esFin()) {
        li.insertar(lista1.recuperar());
        lista1.siguiente();
```

```
    }

    while(!lista2.esFin()) {
        li.insertar(lista2.recuperar());
        lista2.siguiente();
    }
    return li;
}
```