PRG - ETSInf. TEORÍA. Curso 2014-15. Parcial 2.
11 de junio de 2015. Duración: 2 horas.

1. 2.5 puntos Se pide implementar un método estático tal que:

   - Ha de recibir como argumento un `String`, que se considera que contiene la ruta y nombre de un fichero de texto.
   - Se supondrá que cada línea del fichero recibido contiene una representación válida de un número entero o algo que no lo sea. Se desconoce cuántas líneas hay en el fichero.
   - Propagará la excepción `FileNotFoundException` si no pudiera abrir dicho fichero.
   - Sumará todos los números enteros contenidos en el fichero. Devolverá dicha suma.
   - Si se lee una representación válida de un número entero, dicho número se sumará al resultado a devolver.
   - Si se lee una representación no válida de un número entero, la excepción `InputMismatchException` producida se deberá capturar, mostrando en la consola de error un mensaje que incluya el nombre de la excepción y el valor de esa representación no válida. Esta circunstancia no debe impedir que continúe la lectura del fichero.

Solución:

```
public static int sumar (String f) throws FileNotFoundException {
     int suma = 0;
     Scanner sc = new Scanner(new File(f));
     while ( sc.hasNextLine() )
        try {
           suma += sc.nextInt();
        } catch ( InputMismatchException e ) {
           System.err.println(e+"::"+sc.nextLine());
        }
     sc.close();
     return suma;
  }
```

2. 3 puntos Añadir a la clase `ColaIntEnla` un método de perfil

   ```
   public void colar(int x)
   ```

   tal que:

   - Busque la primera ocurrencia del elemento `x` dentro de la cola y en caso de éxito en la búsqueda, haga que dicho elemento se "cuele"delante del todo y por tanto se quede como el primero de la cola.
   - En caso de fracaso en la búsqueda, la cola se queda como estaba.

   NOTA: Sólo se permite acceder a los atributos de la clase, quedando terminantemente prohibido el acceso a sus métodos.

Solución:

```
/** Si x está en la cola, lo pone el primero en cola. */
public void colar(int x)
   NodoInt aux = primero, ant = null;
   while (aux != null && aux.dato != x) {
      ant = aux;
```

```
      aux = aux.siguiente;
   }

   if (aux != null && aux != primero) {
      ant.siguiente = aux.siguiente;
      aux.siguiente = primero;
      primero = aux;

      if (aux == ultimo) ultimo = ant;
   }
}
```

3. 2 puntos Considérese la clase `ListaPIIntEnla`, con todos los métodos conocidos y, además, el método siguiente, que se supone también implementado:

```
/** Devuelve true si n se encuentra en algún nodo de la lista,
 *  false en caso contrario.
 */
 public boolean contiene (int n)
```

Se pide implementar un método estático (se supone que en una clase distinta de `ListaPIIntEnla`) tal que:

- Reciba como argumentos dos objetos de la clase `ListaPIIntEnla`, llamémosles `a` y `b`.
- Ha de insertar en la lista `a` solo los datos almacenados en la lista `b` que no se encuentren previamente almacenados en la lista `a`.
- La inserción en la lista `a` se hará delante del elemento señalado por el `PI`, manteniéndose la posición del `PI`.
- En la lista `b` se puede modificar la posición de su `PI`.
- En la implementación, se debe usar el método `contiene`.

Solución:

```
public static void insertar_nuevos(ListaPIIntEnla a, ListaPIIntEnla b) {
    b.inicio();
    while ( !b.esFin() ) {
       int i = b.recuperar();
       if ( !a.contiene(i) ) a.insertar(i);
       b.siguiente();
    }
}
```

4. 2.5 puntos Dada una `PilaIntEnla p` y un entero `x`, se pide escribir un método estático (se supone que en una clase distinta de `PilaIntEnla`), tal que:

- Calcule y devuelva el número de apariciones de `x` en `p`.
- Debe dejar la pila `p` en el estado en que se encontraba inicialmente.

Solución:

```
public static int numAparicionesEnPila(PilaIntEnla p, int x){
   int n = 0;
   if (!p.esVacia()){
```

```
            int aux = p.desapilar();
            n = numAparicionesEnPila(p,x);
            if ( aux == x) n++;
            p.apilar(aux);
        }
        return n;
    }
```

Alternativamente:

```
    public static int numAparicionesEnPila(PilaIntEnla p, int x){
        int[] aux = new int[p.talla()];
        int n = 0, i = 0;
        while(!p.esVacia()){
            aux[i] = p.desapilar();
            if (aux[i] == x) n++;
            i++;
        }
        for ( i = aux.length-1; i>=0; i--) p.apilar(aux[i]);
        return n;
    }
```