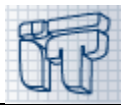


NOMBRE:
APELLIDOS:
NIA:
GRUPO:

2ª Parte: Problemas (7 puntos sobre 10)

Duración: 70 minutos
Puntuación máxima: 7 puntos
Fecha: 09 mayo 2019

Instrucciones para el examen:

- No se permite el uso de libros o apuntes, ni tener teléfonos móviles u otros dispositivos electrónicos encendidos. Incumplir cualquiera de estas normas puede ser motivo de expulsión inmediata del examen.
- Rellena tus datos personales antes de comenzar a realizar el examen.
- Utiliza el espacio de los recuadros en blanco para responder a bolígrafo a cada uno de los apartados de los problemas (no usar lápiz para las respuestas finales).

NOTA: No está permitido crear más atributos ni métodos de los que figuran en el enunciado (no son necesarios).

Problema 1 (Listas / pilas / colas /o colas dobles)

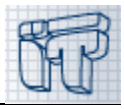
Dadas las clases Node y MyBasicLinkedList que se muestran a continuación, y asumiendo que todos sus métodos están implementados correctamente.

```
public class Node<E> {
      private E info;
      private Node<E> next;
      public Node(E info, Node<E> next) {
            this.info = info;
            this.next = next;
      }
      public Node(E info) {
            this(info, null);
      }
      public Node() {
            this(null, null);
      }
      public E getInfo(){…}
      public Node<E> getNext(){…}
      public void setInfo(E info){…}
      public void setNext(Node<E>next)
{…}
}
```

```
public class MyBasicLinkedList<E> {
      private Node<E> first;
      public MyBasicLinkedList() {
            this.first = null;
      }
      public Node<E> getFirst(){…}
      public void setFirst(Node<E>
first){…}
      public boolean isEmpty(){…}
      public void insert(E info){…}
      public int size(){…}
      public void print(){…}



}
```

Apartado 1:

Programe el método `void insertInSecondPlace(E info)` de la clase `MyBasicLinkedList`, que inserte un nodo con la información que se pasa como parámetro en la segunda posición de la lista.



```
Apartado 1 (1 punto)
```

Apartado 2:

Programe el método `public E removeSecond()` de la clase `MyBasicLinkedList`, que elimina el nodo colocado en segunda posición de la lista y devuelve como resultado su contenido.

```
Apartado 2 (1 punto)
```

Problema 2 (Árboles binarios)

Dada la interfaz `BTree<E>` y las clases `LBNode<E>` y `LBTree<E>`. Asumiendo que todos los métodos de las clases están implementados correctamente y que la clase `LBTree` tiene un constructor `LBTree(E info)`

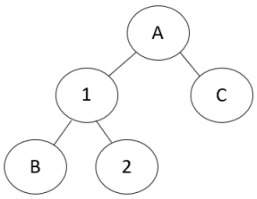
```java
public interface BTree<E> {

   static final int LEFT = 0;
   static final int RIGHT = 1;

   public boolean isEmpty();
   public E getInfo();
   public BTree<E> getLeft();
   public BTree<E> getRight();
   public void insert(BTree<E> tree, int side);
   public BTree<E> extract(int side);
   public String toStringPreOrder();
   public String toStringInOrder();
   public String toStringPostOrder();
   public String toString();
   public int size();
   public int height();
   public boolean equals(BTree<E> tree);
   public boolean find(BTree<E> tree);
}
```

```java
public class LBNode<E> {

   private E info;
   private BTree<E> left;
   private BTree<E> right;

   public LBNode(E info, BTree<E> left, BTree<E> right) {
       this.info = info;
       this.left = left;
       this.right = right;
   }
   public E getInfo() { return info;   }
   public void setInfo(E info) {this.info = info;}
   public BTree<E> getLeft() {return left;}
   public void setLeft(BTree<E> left) {this.left = left;}
   public BTree<E> getRight() {return right;}
   void setRight(BTree<E> right) {
       this.right = right;
   }
}
```

```java
public class LBTree<E> implements BTree<E> { ... }
```

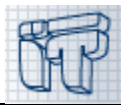
Apartado 1:

Implemente el método **recursivo** `int contarDigitos(BTree<Character> árbol)` que dado un árbol, devuelva como resultado el número de nodos del árbol que contienen como información un `Character` correspondiente a un dígito. Puede asumir que el método `contarDigitos` está en una clase independiente llamada `Test`.



Nota: Para hacerlo puede utilizar el método estático `isDigit()`, de la clase `Character` que comprueba si el carácter pasado como parámetro es un dígito. Si es un dígito devuelve `true`, en caso contrario devuelve `false`.

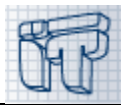
Ejemplo: Dado el siguiente árbol de la figura, el método devolvería como resultado: 2 ya que hay dos nodos que representan un número



```
Apartado 1 (1,25 puntos)
private static int contarDigitos(BTree<Character> arbol) {
```

Apartado 2:

Implemente un método `main` que haga lo siguiente: (1) Crea un árbol con los nodos de los dos primeros niveles del árbol `[A, 1, C]`, (2) realiza los procesos de inserción necesarios para construir el subárbol formado por los tres nodos indicados. (3) Imprima el árbol en pre-orden, (4) imprima el número de dígitos que contiene el árbol creado. Puedes asumir que el método `main` y el método `contarDigitos` implementado en el apartado anterior están dentro de la misma clase.



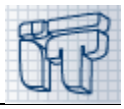
Apartado 2 (1,25 puntos)

Problema 3 (Algoritmos de ordenación y búsqueda)

Un centro comercial ha decidido desarrollar un software para optimizar su almacén. Para hacerlo implementa el algoritmo `Selection Sort` que dado `Array` con el nombre de un conjunto de productos los ordena en orden alfabético creciente.

```
import java.util.Arrays;

public class Problema3 {
  public static void main(String[] args){
```

Nota 1: Para comparar Strings se puede realizar con el método `compareTo()` de



```
    String[] productos = new String[] {"Flan",
        "Esponja", "Arroz", "Detergente",
"Bombones"};

    selectionSort(productos);

System.out.println(Arrays.toString(productos));
    //Imprime el array ordenado
  }
  public static void swap (String[] a, int i,
int j){
    String aux = a[i];
    a[i] = a[j];
    a[j] = aux;
  }
}
```

la clase `String`.

Nota 2: El algoritmo puede hacer uso del método `swap(String[] s, int i, int j)`, que permite intercambiar los elementos de las posiciones `i` y `j` del array `s`.

Ejemplo: Tras ejecutar el código el array productos quedaría así: `{Arroz, Bombones, Detergente, Esponja, Flan}`

Apartado 1:

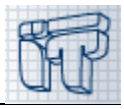
Implementa el método `SelectionSort` (que no debe devolver nada y debe ser estático) para que haga lo requerido.

Apartado 1 (2 puntos)

.

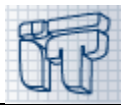
Apartado 2:

Muestre la evolución del algoritmo SelectionSort, después de cada intercambio (llamada al método swap), teniendo en cuenta el array que se declara en el main de la clase `Problema3`



Apartado 2 (0.5 puntos)

Flan	Esponja	Arroz	Detergente	Bombones



Criterios de corrección

Problema 1. Apartado 1 (1 punto)

- (0,25) Considerar caso especial (lista vacía)-condición del if (0 si algún fallo)
- (0,25) Creación del nodo nuevo (0 si algún fallo)
- (0,25) Enlazar el nodo nuevo con el tercero (0 si algún fallo)
- (0,25) Enlazar el primer nodo con el nuevo.

Problema 1. Apartado 2 (1 punto)

- (0,25) Devolver null si la lista está vacía
- (0,25) Condición del if
- (0,25) Almacenar y devolver el valor correcto en el caso genérico
- (0,25) Enlazar el primero con el siguiente al nuevo.

Problema 2. Apartado 1 (1,25 puntos)

- (0,25) Caso base correcto
- (0,5) Caso cuando el nodo actual es un dígito
- (0,5) Caso cuando el nodo actual no es un dígito.

Problema 2. Apartado 2 (1,25 puntos)

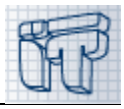
- (0,25) Creación de los árboles individuales (0 si presenta cualquier error)
- (0,25) Inserción en el orden correcto (0 si presenta cualquier error)
- (0,25) Llamada al método imprimir en pre-orden. (0 si presenta cualquier error)
- (0,5) Llamar al método contarDigitos
- (-0,1) si declaran mal el método main

Problema 3. Apartado 1 (2 puntos)

- (0,2) Declaración correcta del método.
  - Penalizar si no ponen `void` y/o `static` (-0.1)
  - Penalizar si no ponen el argumento o si se equivocan en el tipo del `Array` (-0.1)
- (0,4) Primer bucle for.
  - Penalizar si ponen `size()` en lugar de `length` (-0.1)
  - Si los límites no están bien definidos no asignar puntos.
- (0.2) Declaración e inicialización de la variable m
- (0.5) Segundo bucle for
  - Penalizar si ponen `size()` en lugar de `length` (-0.1)
  - Si los límites no están bien definidos no asignar puntos.
- (0.5) Líneas dentro del segundo for
  - Penalizar el mal uso o no uso del compareTo (-0.2)
  - Penalizar si pone en la comparación del compareTo mayor que 0 en lugar de menor (-0.1)
  - Penalizar si no actualiza la variable m (-0.1)
- (0.2) Llamar al método swap correctamente

Problema 3. Apartado 2 (0.5 puntos)

- (0,5) Si demuestran conocimiento sobre los intercambios de las posiciones del array.
  - Penalizar por cada intercambio mal realizado (-0.1)



Solución:

Problema 1. Apartado 1 (1 punto)

```
  public void insertInSecondPlace(E info){
    if(first != null){
      Node<E> nuevo = new Node<E>(info);
        nuevo.setNext(first.getNext());
        first.setNext(nuevo);
      }
  }
```

Problema 1. Apartado 2 (1 punto).

```
  public E removeSecond (){
    E resultado = null;
    if(first != null && first.getNext() != null){
      resultado = first.getNext().getInfo();
        first.setNext(first.getNext().getNext());
      }
      return resultado;
  }
```

Problema 2. Apartado 1 (1,25 puntos)

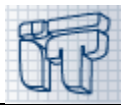
```
public static int contarDigitos(BTree<Character> arbol) {
  int resultado = 0;
  if(arbol.isEmpty()){
    resultado = 0;
  }else if(Character.isDigit(arbol.getInfo())){
    resultado = 1 + contarDigitos(arbol.getLeft()) +
                    contarDigitos(arbol.getRight());
  }else{
    resultado = contarDigitos(arbol.getLeft()) + contarDigitos(arbol.getRight());
  }
  return resultado;
}
```

Problema 2. Apartado 2 (1,25 puntos)

```
public static void main(String[] args){
  BTree<Character> miArbolA = new LBTree<Character>('A');
  BTree<Character> miArbol1 = new LBTree<Character>('1');
  BTree<Character> miArbolC = new LBTree<Character>('C');
  miArbolA.insert(miArbol1, BTree.LEFT);
  miArbolA.insert(miArbolC, BTree.RIGHT);
  System.out.println(miArbolA.toStringPreOrder());
  System.out.println("numDigits: " + contarDigitos(miArbolA));
}
```

Problema 3. Apartado 1 (2 puntos)

```
       public static void selectionSort (String[] s) {
              for (int i=0; i<s.length; i++) {
                     int m = i;
                     for (int j=i; j<s.length; j++) {
                            if (s[j].compareTo(s[m]) < 0){
                                   m = j;
                            }
                     }
                     swap(s, i, m);
                     System.out.println(Arrays.toString(s)); //Imprime el array
              }
       }
```



Problema 3. Apartado 2 (0.5 puntos)

Flan	**Esponja**	**Arroz**	**Detergente**	**Bombones**
Arroz	Esponja	Flan	Detergente	Bombones
Arroz	Bombones	Flan	Detergente	Esponja
Arroz	Bombones	Detergente	Flan	Esponja
Arroz	Bombones	Detergente	Esponja	Flan
Arroz	Bombones	Detergente	Esponja	Flan