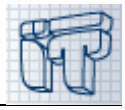


NOMBRE:
APELLIDOS:
NIA:
GRUPO:

Segundo Parcial

2ª Parte: Problemas (7 puntos sobre 10)

Duración: 80 minutos
Puntuación máxima: 7 puntos
Fecha: 7 de mayo de 2019

Instrucciones para el examen:

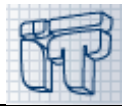
- No se permite el uso de libros o apuntes, ni tener teléfonos móviles u otros dispositivos electrónicos encendidos. Incumplir cualquiera de estas normas puede ser motivo de expulsión inmediata del examen.
- Rellena tus datos personales antes de comenzar a realizar el examen.

Ejercicio 1 (2 / 7 PUNTOS)

En una clase Main se pide implementar el método estático public static void invertir (int numero) que imprime por pantalla en orden inverso el número introducido. Por ejemplo, cuando el parámetro número = 1234, la salida por pantalla es 4321. Notas:

1. Suponga que el **número no sea negativo.**
2. **No se permite la conversión del número a String, debiéndose tratar en todo momento como un número entero.** Recuerda que existe la operación módulo (% en java) que te da el resto de dividir un número entre otro.

Solución (2 puntos)
public static void invertir (int numero)



Ejercicio 2 (2,5 / 7 PUNTOS)

Se dispone de la clase `QueueUc3m` que implementa la interfaz `Queue`:

```
public interface Queue<E> {
   boolean isEmpty();
   int size();
   void enqueue(E info);
   E dequeue();
   E front();
}
```

Suponiendo que todos los métodos de la interfaz ya están implementados en la clase `QueueUc3m`, se desea añadir un nuevo método a dicha clase con la siguiente firma, la cual deberá respetar en su solución del ejercicio:

```
public QueueUc3m<E> mixUp(QueueUc3m<E> q2);
```

Este método recibe otro objeto de la clase `QueueUc3m` y devuelve el resultado de la mezcla de las dos colas, es decir, se intercalan uno a uno los elementos de ambas colas. Por ejemplo, si tenemos las siguientes colas y suponiendo que el campo de información fuera numérico:

Q1: 1 2 3
Q2: 4 5 6 7 8

El resultado a devolver en la cola sería: 1 4 2 5 3 6 7 8

Se pide codificar dicho método utilizando **exclusivamente** los métodos de la clase `QueueUc3m`.

NOTA: Por simplicidad, no es necesario que las colas conserven los elementos originales, ni tampoco importa el orden de los elementos en la cola resultante.

Solución (2,5 puntos)

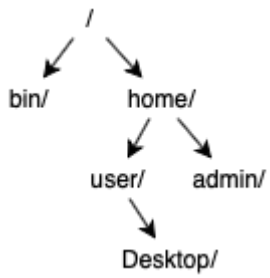
```
public QueueUc3m<E> mixUp(QueueUc3m<E> q2){









}
```

Ejercicio 3 (2,5 / 7 PUNTOS)

Tenemos un árbol binario que representa una estructura de directorios, en la que la información de cada nodo es el nombre del directorio como `String`, y en el que cada directorio puede tener a lo sumo un subdirectorio izquierdo, y otro subdirectorio derecho. La ruta absoluta de un directorio dado se calcula concatenando sucesivamente el nombre de todos los directorios que se encuentran en el camino desde la raíz hasta dicho directorio.

Se pide implementar un método `static void print(String prefix, BTree<String> folder)` recursivo que imprima línea a línea la ruta absoluta de todos los directorios usando un recorrido preorden a partir del directorio raíz.

Ejemplo del árbol y el resultado que se debe mostrar cuando se realiza la llamada `print("", tree)` siendo `tree` una referencia de tipo `BTree<String>` a la raíz del árbol:




```
/
/bin/
/home/
/home/user/
/home/user/Desktop/
/home/admin/
```

La definición de la interfaz `BTree<E>`, y la implementación de las clases `LBNode<E>` y `LBTree<E>` son las mismas que las presentadas en clase. Se recuerda los métodos disponibles:

```
public interface BTree<E> {
   static final int LEFT = 0;
   static final int RIGHT = 1;
   boolean isEmpty();
   E getInfo();
   BTree<E> getLeft();
   BTree<E> getRight();
   void insert(BTree<E> tree, int side);
   BTree<E> extract(int side);
   String toStringPreOrder();
   String toStringInOrder();
   String toStringPostOrder();
   String toString(); // pre-order
   int size();
   int height();
   boolean equals(BTree<E> tree);
   boolean find(BTree<E> tree);
}
```

Solución (2,5 puntos)

```
static void print(String prefix, BTree<String> folder) {









}
```

Soluciones

Problema 1 (2 puntos)

```
public static void invertir (int numero) {
    if (numero < 10)
            System.out.print(numero);
    else{
            System.out.print(numero%10);
            invertir (numero/10);
    }
}
```

RUBRICA

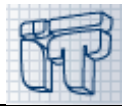
- (0,25) Caso trivial correcto (if numero<10) .
  - Penalizar con un 0 si no se pone el caso trivial correcto.
- (1,75) Caso Recursivo correcto.
  - Si se imprime correctamente quedándose con el resto entre el número y 10 (0,5)
  - Si se llama a la función de modo recursivo "invertir" correctamente (1,25)
  - Si no se sabe hacer en general el caso recursivo un 0.

Problema 2 (2,5 puntos)

```
public QueueUc3m<E> mixUp(QueueUc3m<E> q2){
  QueueUc3m<E> result= new QueueUc3m<E>();

    while((!this.isEmpty()) &&(!q2.isEmpty())) {
      result.enqueue(this.dequeue());
      result.enqueue(q2.dequeue());
    }
    if (this.isEmpty())
      while(!q2.isEmpty())
        result.enqueue(q2.dequeue()) ;
    else
      while(!this.isEmpty())
        result.enqueue(this.dequeue());
    return result;
}
```

RÚBRICA:

- Cero si la solución planteada no tiene sentido, o no se sabe hacer.
- 0,25 si declara e inicializa correctamente la cola para almacenar el resultado.
- 0,75 puntos si se distinguen bien los tres posibles casos (que las dos colas tengan elementos, que sólo los tenga la primera cola, o que sólo los tenga la segunda, siendo 0,25 cada distinción correcta de caso).
- 0,5 por cada caso de los anteriores correctamente resuelto (0,25 el bucle correcto y 0,25 encolar el elemento correctamente).
- Si no usa los métodos de la clase QueueUc3m, se puntúa según los criterios anteriores y se le resta 1/3 a la nota.



Problema 3 (2,5 puntos)

```
static void print(String prefix, BTree<String> folder) {
    if(!folder.isEmpty()) {
        String path = prefix + folder.getInfo().toString();
        System.out.println(path);
        print(path, folder.getLeft());
        print(path, folder.getRight());
    }
}
```

RÚBRICA:

- 0,75: llamar a los casos recursivos con los parámetros correctos.
- 0,50: tener en cuenta algún caso base para detener la recursión.
- 0,50: imprimir las rutas absolutas en preorden.
- 0,75: calcular la ruta absoluta de cada directorio.