ESTRUCTURA DE DATOS Y ALGORITMOS **29-Enero-2009**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**
Nombre y apellidos: _____ Nota:

NOTA FINAL: Nota Practica (1 punto) + Nota Examen (9 punto)

- Es indispensable aprobar el examen (4,5 puntos) para aprobar la asignatura (5 puntos)
- La práctica es opcional
- Duración: 3 horas
- No está permitido el uso de apuntes, libros o móviles
- **Todos los alumnos implicados en una copia** de un ejercicio **tendrán una nota final de 0**. El alumno es responsable de velar por su examen. Es decir **tanto el que copia como el que se deja copiar (ya sea de manera activa o pasiva) recibirán el mismo castigo sin que exista atenuante alguno**
- Publicación de las notas: 15-2-2009
- Revisión del examen: 16-2-2009, 15:00 - 18:00

1. Ejercicio (2 puntos)

a) Una función hash:
   A. Asigna a una clave una posición en una tabla
   B. Coloca el valor asociado a una clave dentro de una tabla
   C. Resuelve los conflictos provocados por claves idénticas
   D. Cambia el valor de una clave al de un índice dentro de la tabla

b) Se han insertado un conjunto de datos en una estructura de datos para su procesamiento. Debido a la característica de la estructura de datos, después de la inserción los datos están siendo procesados en orden inverso al orden de inserción. ¿Qué estructura de datos se está utilizando?

c) Si consideramos básica la operación ***vacíar*** sobre pilas que vacía completamente una pila y la implementamos en Java:
   A. La operación implementada con arrays requiere del recorrido de todo el array y mediante listas enlazadas de solo una operación.
   B. La operación implementada con arrays y mediante listas enlazadas requieren una sola operación.
   C. La operación implementada con arrays y mediante listas enlazadas requiere recorrer todo la estructura.
   D. No es válida ninguna de las anteriores.

d) Explica cuando, como y donde se debe implementar el método *"equals"*.

ESTRUCTURA DE DATOS Y ALGORITMOS **29-Enero-2009**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**
Nombre y apellidos: _____ Nota:

NOTA FINAL: Nota Practica (1 punto) + Nota Examen (9 punto)

- Es indispensable aprobar el examen (4,5 puntos) para aprobar la asignatura (5 puntos)
- La práctica es opcional
- Duración: 3 horas
- No está permitido el uso de apuntes, libros o móviles
- **Todos los alumnos implicados en una copia** de un ejercicio **tendrán una nota final de 0**. El alumno es responsable de velar por su examen. Es decir **tanto el que copia como el que se deja copiar (ya sea de manera activa o pasiva) recibirán el mismo castigo sin que exista atenuante alguno**
- Publicación de las notas: 15-2-2009
- Revisión del examen: 16-2-2009, 15:00 - 18:00

1. Ejercicio (2 puntos)

a) Si consideramos básica la operación ***vacíar*** sobre pilas que vacía completamente una pila y la implementamos en Java:
   A. La operación implementada con arrays requiere del recorrido de todo el array y mediante listas enlazadas de solo una operación.
   B. La operación implementada con arrays y mediante listas enlazadas requieren una sola operación.
   C. La operación implementada con arrays y mediante listas enlazadas requiere recorrer todo la estructura.
   D. No es válida ninguna de las anteriores.

b) Se han insertado un conjunto de datos en una estructura de datos para su procesamiento. Debido a la característica de la estructura de datos, después de la inserción los datos están siendo procesados en orden inverso al orden de inserción. ¿Qué estructura de datos se está utilizando?

c) Una función hash:
   A. Asigna a una clave una posición en una tabla
   B. Coloca el valor asociado a una clave dentro de una tabla
   C. Resuelve los conflictos provocados por claves idénticas
   D. Cambia el valor de una clave al de un índice dentro de la tabla

d) Explica cuando, como y donde se debe implementar el método *"equals"*.

Nombre y apellidos: _____

NOTA FINAL: Nota Practica (1 punto) + Nota Examen (9 punto)

- Es indispensable aprobar el examen (4,5 puntos) para aprobar la asignatura (5 puntos)
- La práctica es opcional
- Duración: 3 horas
- No está permitido el uso de apuntes, libros o móviles
- **Todos los alumnos implicados en una copia** de un ejercicio **tendrán una nota final de 0**. El alumno es responsable de velar por su examen. Es decir **tanto el que copia como el que se deja copiar (ya sea de manera activa o pasiva) recibirán el mismo castigo sin que exista atenuante alguno**
- Publicación de las notas: 15-2-2009
- Revisión del examen: 16-2-2009, 15:00 - 18:00

2. *Ejercicio (3 puntos)*

Dada la siguiente especificación de las clases BTNode, BinTree y BinTreeItr:

```
public class BinTreeItr{
     BTNode current;
     BinTree bTree;
}

public class BinTree{
     BTNode root;
}
```

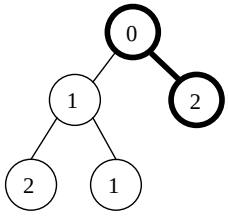
```
public class BTNode{
     Object content;
     BTNode left;
     BTNode right;
}
```

Nota: Todo método necesario tendrá que ser implementado.

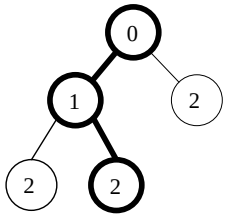
Especificar, diseñar e implementar un método recursivo que dado un árbol binario de enteros, calcule y devuelva cual es la longitud del camino que mayor coste supone recorrerla. Un **camino** es una secuencia de nodos en la que cada nodo es adyacente al siguiente. Cada nodo del árbol puede ser alcanzado (se llega a él) siguiendo un único camino que comienza en el nodo raíz y su contenido (el entero) representa el coste de visitarlo. Además, visitar los **nodos derechos cuesta el doble** que visitar los izquierdos, mientras el coste del nodo raíz siempre es 0.

```
int longitudCaminoCosteMaximo();
```

Ejemplos:



Longitud = 2



Longitud = 3

ESTRUCTURA DE DATOS Y ALGORITMOS **29-Enero-2009**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**

3. *Ejercicio (4 puntos)*

Desarrollar un sistema informático que modele el comportamiento de un mercado de trabajo simplificado, donde las personas pueden ser contratadas y despedidas por empresas de forma muy fácil y a menudo. Una persona puede estar trabajando en una empresa o no, y una empresa, como es lógico, tiene contratado a varias personas. Comentar que todas las personas tienen distinto dni, lo mismo que, todas las empresas tienen distinto nif.

Para ello suponemos disponibles la implementación de las siguientes clases, todas ellas equipadas con operaciones de acceso, igualdad (*equals*) y orden (implementan la interfaz *Comparable*, es decir, el método *compareTo*):

- *Persona* que sirve para almacenar y gestionar la información sobre una persona (dni, nombre, dirección, edad)
- *Empresa* que sirve para almacenar y gestionar la información sobre una empresa (nif, nombre, dirección).

Se pide:

a) Diseña e implementa la estructura de la clase *Mercado*, cuya misión es gestionar la información sobre el mercado de trabajo, teniendo en cuenta as características del problema y las estructuras de datos vistas en clase. También implementa las clases necesarias para representar las relaciones entre la información a gestionar. **Nota**: no implementes ni modifiques las clases Persona y Empresa.

b) Implementa las siguientes operaciones públicas de la clase *Mercado*:

- ***Constructor***: Genera un mercado vacío, sin ninguna información.
- ***contrata***: Altera un mercado, efectuando la contratación de cierta persona como empleado de cierta empresa.
- ***despide***: Altera un mercado, efectuando el despido de cierta persona que era antes empleado de cierta empresa.
- ***empleados***: Consulta los empleados de una empresa, devolviendo como resultado un array ordenado de personas.
- ***esEmpleado***: Averigua si es cierto o no que una persona dada es empleado de una empresa dada.

Nota: No es necesario implementar los métodos típicos de cualquier estructura de datos que hayamos visto en clase, siempre que se utilicen los nombres habituales.