

Familia Profesional Informática y Telecomunicaciones		Nombre del Ciclo Formativo Título de Técnico Superior en Desarrollo de Aplicaciones Web			
Centro Educativo IES Campanillas (sede PTA)		Módulo Profesional Programación Código: 0485 N.º de créditos ECTS: 14		Profesor Juan Antonio Jiménez Morales	
Curso lectivo: 2019/2020	Grupo: 1º DAW	Trimestre: Terº – Control 6	Modelo: Único	Fecha: 16/04/2020	Pág. 1/5

INSTRUCCIONES

- ➔ El alumno debe entregar una carpeta con las soluciones al examen cuyo nombre debe estar formado por “Ex” seguido del número de lista, seguido de las iniciales. Por ejemplo, Facundo Romuedo Piladro que es el número 8 de la lista entregaría una carpeta con nombre **Ex08frp**.
- ➔ Los ficheros o carpetas correspondientes a las soluciones se deben nombrar igual que la carpeta junto con el número del ejercicio, por ejemplo **Ex08frp1.java, Ex08frp2.java, etc.**
- ➔ En los comentarios de cada programa **se debe indicar el nombre completo**, la fecha y - si procede - el turno. También debe indicar una breve descripción de lo que hace el programa.
- ➔ Únicamente se necesita entregar el código fuente en java, **no se deben entregar los archivos con la extensión .class**.

EJERCICIOS

1. [3 puntos] Implemente en JSP el “Juego de la Vida”. Este juego consiste en simular la propagación de la vida sobre los puntos de una matriz bidimensional. Para ello, se parte de una posición inicial (tablero vacío) y se aplican una serie de reglas que indican si a cada paso (generación) las celdas toman vida, la mantienen o se mueren. Para representar esta matriz de puntos utilizaremos una tabla en la que en cada celda habrá un checkbox, el cual tendrá vida si está marcado y no tendrá vida si está en blanco.

Por tanto, partiremos de un estado inicial, e iremos evolucionando a la siguiente generación (siguiente estado) a partir del estado actual, y según las siguientes reglas:

- a) Una célula muerta con exactamente 3 células vecinas vivas "nace" (es decir, al turno siguiente estará viva).
- b) Una célula viva con 2 o 3 células vecinas vivas sigue viva, en otro caso muere (por "soledad" o "superpoblación").

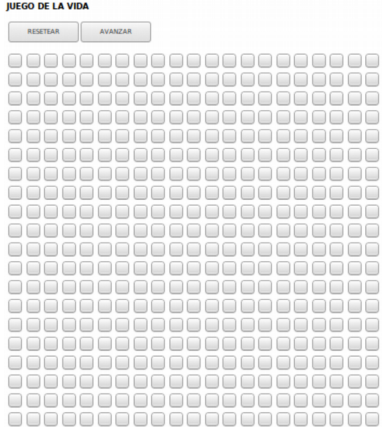
Para implementar este juego, se le sugiere que envíe el estado de cada celda como variables de un formulario. Podrá recoger el valor de estas variables desde JSP y “cargarlas” en una matriz bidimensional, la cual podrá utilizar para calcular el valor del siguiente estado. Se le sugiere que utilice una función en JSP para a partir del estado actual recibido calcular el nuevo estado.

Una vez obtenido el nuevo estado, puede presentar por pantalla la matriz con la vida en cada una de sus celdas.


Tenga en cuenta que la matriz será siempre cuadrada y que todo el ejercicio debe estar parametrizado a partir del valor de una constante definida en el programa JSP, de manera que cambiando el valor de dicha constante el juego se adapte, sin necesidad de tocar nada más en el código, al nuevo tamaño.

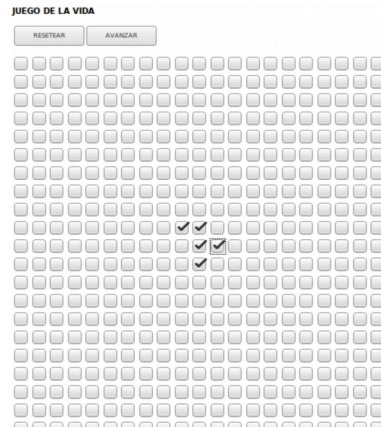
Debido al uso de una “matriz” de checkboxes, queda abierta la posibilidad de crear vida o quitarla en las celdas que el usuario desee después de evolucionar a un estado y antes de ir al siguiente. Es por este método por el que se establecerá la configuración inicial de celdas con vida.

Ejemplo (para una matriz de 21x21):



Tras introducir el estado inicial,
podría quedar así:





Familia Profesional Informática y Telecomunicaciones		Nombre del Ciclo Formativo Título de Técnico Superior en Desarrollo de Aplicaciones Web			
Centro Educativo IES Campanillas (sede PTA)		Módulo Profesional Programación Código: 0485 N.º de créditos ECTS: 14		Profesor Juan Antonio Jiménez Morales	
Curso lectivo: 2019/2020	Grupo: 1º DAW	Trimestre: Terº – Control 6	Modelo: Único	Fecha: 16/04/2020	Pág. 2/5

Ahora, tras pulsar sucesivamente en AVANZAR, obtendríamos los siguientes estados:



Y continuaría sucesivamente hasta que se estabilizara la población. El usuario podrá, alcanzado cualquier estado, cambiarlo a conveniencia, ya que puede interactuar sobre las casillas de la tabla (tal y como se hizo para establecer el estado inicial a partir de "la nada").

Familia Profesional Informática y Telecomunicaciones		Nombre del Ciclo Formativo Título de Técnico Superior en Desarrollo de Aplicaciones Web			
Centro Educativo IES Campanillas (sede PTA)		Módulo Profesional Programación Código: 0485 N.º de créditos ECTS: 14		Profesor Juan Antonio Jiménez Morales	
Curso lectivo: 2019/2020	Grupo: 1º DAW	Trimestre: Terº – Control 6	Modelo: Único	Fecha: 16/04/2020	Pág. 3/5

2. [3 puntos] Escriba un programa para implementar diccionarios de sinónimos. Para ello:

Debe implementar la clase `DiccionarioSinonimos` con las siguientes especificaciones:

- Respecto a los atributos de instancia, debe utilizar:
 - un `String` para almacenar el nombre del diccionario
 - un `HashMap`, consistente en un `String` como valor clave y un `ArrayList` de `Strings` como lista de sinónimos asociada a dicho valor clave.
- `Constructor`: se le indicará el nombre del diccionario. El `HashMap` estará inicialmente vacío.
- `toString`: devuelve la representación como cadena de caracteres de todo el diccionario (véase ejemplo más adelante)
- `aniadeSinonimo`: se le indicará la palabra a la que se quiere añadir un sinónimo, y el sinónimo a añadir. Tenga en cuenta que:
 - Si la palabra a la que se desea añadir un sinónimo no está en el `HashMap`, deberá añadir la entrada correspondiente, inicializar convenientemente el `ArrayList` asociado como valor y añadir en el `ArrayList` dicho sinónimo.
 - No podrá añadir el sinónimo a dicha palabra si ya lo tiene asociado. En cualquier otro caso, lo añadirá. Devolverá `false` si no ha podido añadirlo y `true` en caso contrario.
- `dameSinonimos`: devolverá la lista de sinónimos para una palabra dada. Si no hay sinónimos para una palabra dada, devolverá el mensaje: NO HAY.
- `eliminaSinonimo`: se le indicará la palabra a la que se desea eliminar un sinónimo, y el sinónimo que se desea eliminar. Tenga en cuenta que si la palabra se queda sin sinónimos, deberá eliminar la entrada del `HashMap`.
- `esSinonimo`: se le indicarán 2 palabras y devolverá `true` si la segunda aparece en el listado de sinónimos de la primera, `false` en caso contrario.

Una vez implementada esta clase, debería producirse la siguiente salida para el código JAVA facilitado:

```
public static void main (String[] args) {
    DiccionarioSinonimos dicci1;
    dicci1 = new DiccionarioSinonimos("Primer diccionario");
    dicci1.aniadeSinonimo("grande","enorme");
    dicci1.aniadeSinonimo("grande","inmenso");
    dicci1.aniadeSinonimo("pequeño","diminuto");
    System.out.print(dicci1);
    System.out.println("Sinonimos de grande: "+dicci1.dameSinonimos("grande"));
    if (dicci1.esSinonimo("grande", "enorme")) {
        System.out.println("'grande' y 'enorme' son sinonimos.");
    } else {
        System.out.println("'grande' y 'enorme' NO son sinonimos.");
    }
}
```

```
Primer diccionario
=====
pequeño:
    | diminuto
grande:
    | enorme | inmenso
=====
Sinonimos de grande: | enorme | inmenso
'grande' y 'enorme' son sinonimos.
```

Familia Profesional Informática y Telecomunicaciones		Nombre del Ciclo Formativo Título de Técnico Superior en Desarrollo de Aplicaciones Web			
Centro Educativo IES Campanillas (sede PTA)		Módulo Profesional Programación Código: 0485 N.º de créditos ECTS: 14		Profesor Juan Antonio Jiménez Morales	
Curso lectivo: 2019/2020	Grupo: 1º DAW	Trimestre: Terº – Control 6	Modelo: Único	Fecha: 16/04/2020	Pág. 4/5

3. [2,5 puntos] Gestión de un Zoológico. A continuación se presenta la jerarquía de clases que representa los animales de un posible zoológico:

La clase `Animal` es una clase abstracta con cuatro atributos privados de instancia:

- Una cadena de caracteres indicando la `especie` (león, águila, abeja)
- Una cadena de caracteres indicando el `nombre` del animal concreto
- Un dato numérico real indicando el `peso` en kg.
- Un dato numérico entero indicando el `número` de `jaula` que se asigna al animal.

Además, la clase `Animal` debe imponer sobre sus clases derivadas la implementación de un método (que en ella no existirá realmente; será un “método virtual”) denominado `MuestreComoCadena`: este método, que deberá ser implementado obligatoriamente en las clases derivadas de esta clase abstracta (esta imposición se adquiere por herencia), será definido a través de una interfaz denominada `AnimalComoCadena`, la cual debe ser “cumplida” por la clase `Animal`. La implementación de este método en cada una de las clases derivadas consistirá en mostrar por consola lo que devuelva el método `toString` de dicha clase.

De la clase `Animal` se derivan las clases `Mamifero`, `Ave` e `Insecto`.

La clase `Mamifero` no añade nuevos atributos de instancia, aunque deberá implementar el método `MuestreComoCadena` según las condiciones anteriormente indicadas.

La clase `Ave` tiene dos nuevos atributos privados de instancia:

- Una cadena de caracteres indicando el `color` predominante del `plumaje`
- Un dato numérico real indicando la `altura` máxima de `vuelo`.

La clase `Insecto` tiene un nuevo atributo privado de instancia, de tipo booleano, llamado `vuela`, que indica si el insecto `vuela` o no.

Para realizar este ejercicio se pide lo siguiente:

- Crear las 4 clases indicadas, junto con la interfaz. Debe implementar constructores de manera que se optimice el código, dándole valores iniciales a todos los atributos de los objetos.
- Implementar en cada clase el método `toString`, el cual devuelve la representación del objeto como cadena de caracteres (véase ejemplo posterior). También debe lograr una optimización del código.
- Implementar en cada clase instanciable el método definido en la interfaz, según lo indicado anteriormente.
- Implementar un programa de prueba, en el que se muestre el funcionamiento de este ecosistema de clases.

Familia Profesional Informática y Telecomunicaciones		Nombre del Ciclo Formativo Título de Técnico Superior en Desarrollo de Aplicaciones Web			
Centro Educativo IES Campanillas (sede PTA)		Módulo Profesional Programación Código: 0485 N.º de créditos ECTS: 14		Profesor Juan Antonio Jiménez Morales	
Curso lectivo: 2019/2020	Grupo: 1º DAW	Trimestre: Terº – Control 6	Modelo: Único	Fecha: 16/04/2020	Pág. 5/5

4. [1,5 puntos] Rehaga el ejercicio n.º 1 utilizando un ArrayList de ArrayList para almacenar los estados de las casillas, lo que sería pseudoequivalente a un “ArrayList bidimensional”.