Programación I

Examen Convocatoria Ordinaria 11-Enero-2016

Hace mucho tiempo en una galaxia muy muy lejana....

Tras la caída del Imperio Galáctico a manos de los rebeldes, el orden galáctico paso de nuevo a manos de la República Galáctica. Sin embargo, un último reducto imperial, liderado por un Maestro Sith que se hace llamar Snoke, formó la Primera Orden, los auto-proclamados herederos del Imperio galáctico. La rebelión pasó a ser la Resistencia y está liderada por la general Leia Organa. Su hermano Luke Skywalker, último maestro Jedi de la galaxia, fundó la orden Jedi. Sin embargo, un Jedi, que ahora se hace llamar Kylo Ren, sucumbió al lado oscuro y la orden de los Caballeros de Ren acabaron con la vida de todos los Jedis menos con la de Luke.

Tras este catastrófico evento, Luke se exilió, alejándose de la República y fue en busca del mítico templo Jedi. Ante los últimos avances de Ren y Snoke, la general Leia Organa, convencida que ante esta nueva amenaza proveniente de la Primera Orden, necesitan encontrar a su hermano Luke Skywalker y único Maestro Jedi con vida, nos ha pedido que realicemos algunas tareas para ayudar a recuperar su posición en la galaxia. Para ello, en el programa principal disponemos de las siguientes estructuras de datos:

- aPlanetas, es un array de Strings que almacena la lista de planetas probables por los que ha podido pasar Luke en su exilio hasta el templo perdido de los Jedi.
- aRutas: es un array de objetos Ruta que indican las rutas de comercio. La definición de la clase es:
    - origen: es el planeta origen de la ruta.
    - destino: es el planeta destino de la ruta.
    - controladaPrimeraOrden: es un boolean que indica que si la ruta está bajo el control de la Primera Orden y por tanto son enlaces que Luke es complicado que haya tomado.
- aContObservaciones, almacena el número de veces que ha parecido ver a Luke en cada una de las rutas. Es un array bidimensional:

	Tatooine	Dagobah	Jakku	Naboo
Tatooine	0	2	0	5
Dagobah	1	0	3	1
Jakku	2	1	2	3
Naboo	3	0	0	0

La clase Ruta **la tenéis implementada** al final del examen.

Para llevar a cabo nuestro encargo, se debe implementar la clase Observación.

[2 puntos] Clase Observacion

La clase Observacion representa una observación de Luke en alguna ruta. Está compuesta por los siguientes atributos:

- Planeta. El nombre del planeta donde se le ha visto.

- rutaOrigen: nombre del planeta origen de la ruta.
- rutaDestino: nombre del planeta destino de la ruta
- descripciónObservación: cadena de texto con la descripción de la observación.
- fechaObservación: un entero con formato aaaammdd

Para manejar estos datos, la clase facilita los siguientes métodos:

- **Constructor** sin argumentos, que inicializa todos los atributos con valores válidos por defecto (cada programador debe elegir cuáles son estos valores)
- **Constructor** con parámetros, que inicializa todos los atributos con valores pasados como parámetro, comprobando que son valores válidos. En caso de que no lo sean, se inicializarán con valores por defecto (iguales a los del constructor anterior).
- Métodos **set y get** para todos los atributos de la clase,. IMPORTANTE: cada vez que se cambie el valor de un atributo de la clase, se debe comprobar que el nuevo valor a asignar sea válido, en caso contrario se dejará el valor que tenía antes (no se debe mostrar ningún mensaje por pantalla)
- Método **leer**, que lee por teclado todos los atributos de la clase y comprueba que sean correctos. En caso de no serlo los seguirá pidiendo una y otra vez..
- Método **mostrar**, que muestra todos los atributos del objeto por pantalla.

[1 puntos] Método actualizarObservaciones

En base a las observaciones que están almacenadas en un array que se pasa como parámetro (descrito para el main como aContObservaciones), se actualizará en ese array en la posición correspondiente a la ruta el valor, añadiendo la observación en curso. Para ello, también necesitaremos del array aPlanetas como parámetro para saber las posiciones en el array bidimensional.

Clase Gestion

Una vez creada la clase Observacion, se deben programar los siguientes métodos para gestionar la búsqueda de Luke Skywalker

[1,5 puntos] Método cargarObservaciones

Este método lee por teclado los datos de todas las observaciones, que serán un máximo de 1000, y los almacena en el array *aObservaciones* en orden descendiente de fecha (si no se sabe hacer en orden la inserción, se obtiene un máximo de 0,75 ptos). Después de introducir cada observación se preguntará al usuario si quiere introducir más (contemplar las mayúsculas y minúsculas). El proceso finaliza al introducir 1000 observaciones o cuando el usuario indique que no quiere introducir más.

[2’5 puntos] Método obtenerProbabilidadesLuke

Para calcular la probabilidad, se evalúa de la siguiente forma. Se recorre el array tal y como está ordenado por fecha y se calcula por cada observaciones utilizando el método actualizarObservaciones. Una vez actualizadas, se obtiene el número de observaciones máxima de todas las rutas pero únicamente teniendo en cuenta las que no están controladas por el Primer Orden. Se dividen todas las observaciones por ese valor máximo (las controladas por el primer orden quedan a 0), generando un array de floats con las probabilidades que se devuelve al main.

[3 puntos] Método ordenarPosibilidades

Este método debe mostrar todas las observaciones (toda la información) por orden descendente de probabilidad calculada en el método anterior y en caso de empates la fecha más actual se mostrará antes.

Clase Ruta

```
public class Ruta {
	private String origen;
	private String destino;
	private boolean controladaPrimeraOrden;

	public Ruta(String origen, String destino, boolean
controladaPrimeraOrden) {
		this.origen = origen;
		this.destino = destino;
		this.controladaPrimeraOrden = controladaPrimeraOrden;
	}

	public Ruta() {
		this.origen = "";
		this.destino = "";
		this.controladaPrimeraOrden = false;
	}

	public String getOrigen() {
		return origen;
	}

	public void setOrigen(String origen) {
		this.origen = origen;
	}

	public String getDestino() {
		return destino;
	}

	public void setDestino(String destino) {
		this.destino = destino;
	}

	public boolean isControladaPrimeraOrden() {
		return controladaPrimeraOrden;
	}

	public void setControladaPrimeraOrden(boolean controladaPrimeraOrden){
		this.controladaPrimeraOrden = controladaPrimeraOrden;
	}

}
```

Clase Gestion

```
public class Gestion {
     public static void main(String[] args){
          String [] aPlanetas = {"Tatooine","Dagobah","Jakku","Naboo"};

          int[][] aContObservaciones = {
                    {0, 0, 0, 0},
                    {0, 0, 0, 0},
                    {0, 0, 0, 0},
                    {0, 0, 0, 0},
               };

          Ruta [] aRutas = {
               new Ruta("Tatooine", "Dagobah", false),
               new Ruta("Tatooine", "Jakku", true),
               new Ruta("Tatooine", "Naboo", false),
               new Ruta("Dagobah", "Tatooine", false),
               new Ruta("Dagobah", "Jakku", true),
               new Ruta("Dagobah", "Naboo", false),
               new Ruta("Jakku", "Tatooine", true),
               new Ruta("Jakku", "Dagobah", false),
               new Ruta("Jakku", "Naboo", true),
               new Ruta("Naboo", "Tatooine", false),
               new Ruta("Naboo", "Dagobah", true),
               new Ruta("Naboo", "Jakku", true),
          };

          Observacion aObservaciones = new Observacion[1000];
          cargarObservaciones(aObservaciones);
          float [][] aProbabilidades =
obtenerProbabilidadesLuke(aObservaciones,aContObservaciones, aRutas);
          ordenarPosibilidades(aObservaciones, aProbabilidades,aRutas);

     }
}
```