PRÁCTICA 3: SIMULADOR "INVASIÓN"

OBJETIVOS

El alumno debe dominar los siguientes conceptos utilizados en la práctica:

- Utilización correcta de clase abstracta.
- Adecuado uso de herencia de clases, reutilización de código (super, toString…), modularidad y encapsulamiento de la información.
- Utilización adecuada de atributos *static* en las clases para un uso eficiente de la memoria.
- Uso de memoria dinámica.
- Modo gráfico.
- Serialización.

Objetivo secundario:

- Generación de números aleatorios en Java.

La práctica está dividida en dos partes. Finaliza la primera parte y una vez funcione correctamente, amplía el programa para incluir la funcionalidad de la segunda parte.

EVALUACIÓN

Además de los puntos anteriores, en esta práctica se tendrá en especial consideración para su evaluación, el seguimiento de los aspectos expuestos en el documento "NormasDePrácticas":

- Modularidad
- Robustez
- Integridad de los datos
- Eficiencia y eficacia
- Pruebas del software

INTRODUCCIÓN

Vamos a simular un entorno en el que los humanos son invadidos por zombies y vampiros.
En nuestro simulador tendremos:

- Humanos: Los humanos nacen, se reproducen y mueren.
  De los humanos nos interesa conocer el día de su nacimiento.
  Los humanos se caracterizan por su velocidad al correr.

- Humanos cazavampiros: debido a la proliferación de vampiros, los cazavampiros (humanos que han aprendido ciertas técnicas de caza) son esenciales en esta sociedad, ya que sin ellos, los vampiros podrían acabar con la raza humana.

- Vampiros: los vampiros no mueren por muerte natural. Únicamente pueden morir al ser cazados por cazavampiros, o bien si no pueden obtener la sangre que requieran. Al morder a un humano, en ocasiones, convierten al humano en vampiro.

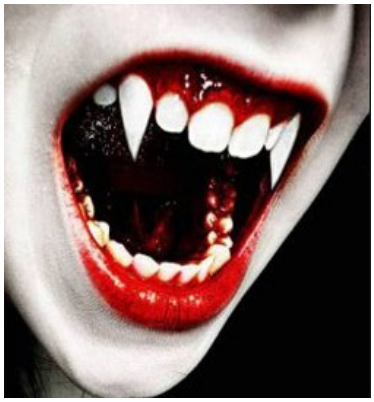
- Zombies: los zombies no se alimentan, y por esa razón mueren tras unos días de vida. Si durante su vida consiguen alcanzar (tocar) a un humano, lo convierten en zombie.

CREACIÓN DE NUESTRO ENTORNO

Nuestro simulador comenzará el día 1 y a partir de ahí irán pasando (incrementándose) los días. Para simplificar, trataremos las fechas por el número de día (no tendremos en cuenta los meses, años, etc.).

Cada vez que arranquemos el programa creará un entorno distinto con una temperatura inicial de 20 grados. Tendremos inicialmente un número concreto de seres de cada grupo. Para determinar el número de seres de cada grupo usaremos números aleatorios. Cada entorno al inicio, tendrá:

- Número de humanos (no cazavampiros) en el rango: 4.000 – 6.000
- Número de humanos cazavampiros en el rango: 10 - 15
- Número de vampiros en el rango: 15 - 20
- Un número de zombies en el rango: 20 - 30



VIDA DIARIA EN NUESTRO ENTORNO

Primeramente tenemos que tener en cuenta la temperatura de nuestro entorno. Dicha temperatura afecta a la reproducción de los humanos (a mayor temperatura, mayor reproducción).

La temperatura del entorno de simulación varía cada día. Concretamente tiene una probabilidad de subir o bajar medio grado según la tabla siguiente, en función de su temperatura actual:

Temperatura actual	**Cambio de temperatura cada día**
Mayor o igual a 22 grados	Probabilidad de subir medio grado: 45 % Probabilidad de bajar medio grado: 55 %
Menor a 22 y mayor a 18 grados	Probabilidad de subir medio grado: 65 % Probabilidad de bajar medio grado: 30 % Probabilidad de no variar su temperatura: 5 %
Menor o igual a 18 grados	Probabilidad de subir medio grado: 55 % Probabilidad de bajar medio grado: 45 %

Por tanto, para cada día que pase, deberás calcular la nueva temperatura en función de la temperatura actual y la probabilidad de la tabla anterior.

El comportamiento de cada ser será simulado también con probabilidades.

Vemos a continuación las probabilidades de cada grupo:

- **Humanos (no cazavampiros)**:
    - Se reproducen con una probabilidad diaria que varía en función de la temperatura (a mayor temperatura, mayor probabilidad de reproducción):

Temperatura actual	**Probabilidad diaria de reproducción**
Mayor o igual a 22 grados	Probabilidad de reproducción: 1/15
Menor a 22 y mayor a 18 grados	Probabilidad de reproducción: 1/30
Menor o igual a 18 grados	Probabilidad de reproducción: 1/50

    - Si un día se reproduce un humano, ese día podrá tener un número de hijos en el rango 1-3 (con la misma probabilidad para cada opción). Para simplificar el simulador, los humanos no tienen sexo, y cada individuo puede reproducirse utilizando las probabilidades anteriores.
    - Un humano muere diariamente por muerte natural con una probabilidad de 1/500.
    - Aparte, un humano puede morir diariamente por otras causas (accidentes, inanición, enfermedades, etc.) con una probabilidad diaria de 1/300.
    - De cada humano nos interesa almacenar y conocer en el programa el día de nacimiento y la velocidad al correr (que tiene una medida de 60 a 100).
    - La velocidad será asignada a cada humano en el rango 60-100 al crearse el entorno. Después, las velocidad se hereda, es decir, cuando un humano tiene descendencia, sus hijos tienen su misma velocidad.

- **Humanos cazavampiros**:
    - Puesto que son humanos, las probabilidades de reproducción y muerte son las mismas a las anteriores. Sucede lo mismo con la velocidad, cuyo comportamiento es igual.
    - Cuando un cazavampiros tiene hijos, todos ellos son también cazavampiros, ya que el padre les enseña las técnicas de caza.
    - Un cazavampiros tiene una probabilidad diaria de cazar, y por tanto, matar a un vampiro (sólo a uno) de 1/3. Si le corresponde cazar, pero no existen vampiros por haberse extinguido éstos, no pasa nada, simplemente ese día no cazará.
    - De los humanos cazavampiros nos interesa registrar su fecha de nacimiento (el día).
    - También es importante conocer su velocidad al correr.
    - Por último, necesitamos saber el número de vampiros que ha matado durante su vida.

- **Vampiros**:
    - Cada vampiro se alimenta (muerde) a un hombre no cazavampiros (y sólo uno) cada día con una probabilidad de un 50%. Si dicho día le corresponde comer (morder a un humano), entonces puede suceder dos cosas: o bien mata al humano (porque le chupa demasiada sangre) o bien convierte al humano en vampiro (al chuparle poca sangre). Ambas opciones tienen la misma probabilidad, es decir, un 50%.
    - Los vampiros no pueden morder a humanos cazavampiros, porque éstos saben protegerse.
    - Los vampiros únicamente tienen dos formas de morir (son inmortales salvo):

1) si lo mata un cazavampiros
2) muere de hambre (por inanición) si el día que le corresponde comer (morder a un humano) no puede hacerlo por no existir humanos (pueden haberse extinguido).

- De los vampiros nos interesa conocer el día de nacimiento como vampiro y el número de vampiros que ha "generado" (es decir, el número de humanos que tras morder ha convertido en vampiro).

- **Zombies**:
  - Los zombies pueden ir convirtiendo hombres en zombies sólo con tocarlos. Para ello, debes tener en cuenta que diariamente cada zombie tiene una probabilidad de convertir a un hombre (y sólo uno) en zombie de 1/10. En caso de convertir a un hombre en zombie, siempre será el de menor velocidad. Aquí no hay distinción de cazavampiros o no cazavampiros, ya que todos ellos pueden ser alcanzados y convertidos en zombies.
  - Los zombies no comen, por eso mueren siempre de hambre a los 8 días de ser zombies.
  - De los zombies nos interesa conocer el día de nacimiento como zombie (o día en el que se ha convertido el humano en zombie) y el número de zombies que ha "generado" (es decir, el número de humanos a los que ha alcanzado y convertido en zombie).

ACONTECIMIENTO: CALENTAMIENTO GLOBAL

Como es sabido, la temperatura del entorno afecta a la reproducción de los humanos.

Introduce en el programa una opción que permita de forma inmediata, y sólo para ese día, incrementar la temperatura en 10 grados. Los siguientes días evolucionarán a partir de dicha temperatura. Podremos así evaluar los efectos provocados cuando apliquemos este incremento de temperatura durante la simulación.

ACONTECIMIENTO: ENFRIAMIENTO GLOBAL

Introduce en el programa una opción que permita de forma inmediata (y sólo ese día) decrementar la temperatura en 10 grados. Los siguientes días evolucionarán a partir de dicha temperatura.

ACONTECIMIENTO: INVASIÓN DE ZOMBIES

Los zombies se han vuelto más veloces. Crea una opción que permita incrementar la probabilidad de que un zombie alcance, y por tanto convierta a un humano en zombie, a 1/3.

PARTE 1

Implementa la simulación de nuestro entorno expuesto hasta ahora, creando las clases que creas conveniente y cumpliendo con los siguientes **requerimientos**:

- Utiliza una clase abstracta como base para los seres y herencia.

- Ten en cuenta la reutilización de código y el encapsulamiento de la información.

- Crea un método para crear nuestro entorno con todos los seres, usando una distribución aleatoria según los requerimientos arriba expuestos.

- Utiliza memoria dinámica (un único ArrayList o HashSet de Java) para almacenar todos seres. Al utilizar memoria dinámica, el número de seres será ilimitado. Recuerda sus métodos más útiles: add(), size(), get(i), remove(i).

- Puedes usar (si lo necesitas) el siguiente método para diferenciar la clase de un objeto:

  If (obj1 instanceof Clase1)

  Método que devuelve true o false según obj1 sea o no de la clase Clase1.

- Crea diferentes métodos para los seres, que permitan realizar las acciones diarias básicas: comer, reproducirse y morir. Dentro de estos métodos, se realizarán las llamadas a la generación de los números aleatorios. Crea más métodos si lo consideras necesario.

  Vemos un ejemplo: *Un cazavampiros tiene una probabilidad diaria de cazar, y por tanto, matar a un vampiro (sólo uno) de 1/3.*

  Debes generar dentro del método *comer* del cazavampiro un número aleatorio para cada cazavampiro entre 1 y 3. Si se genera el número 1, entonces a ese cazavampiro le corresponde cazar a un vampiro. En ese caso deberás "matar" (eliminar) a un vampiro del vector.

  Vamos a ver otro ejemplo: *en el caso de los humanos, se reproducen (con 20 grados de temperatura) en una probabilidad diaria de 1/30*.

  En este caso generaremos un número aleatorio para cada humano de 1 a 30, y si el número generado es 1, significa que se reproducirá. Sólo en ese caso, deberás generar después un número aleatorio de 1 a 3 para saber cuántos hijos va a tener. Dicho número podría devolver el método de comer del humano. Recuerda que sus hijos tendrán (heredan) su misma velocidad.

- La ejecución del programa debe ser en **modo gráfico** (lo que implica crear por código clases que extiendan de JFrame, objetos gráficos en dichas clases, etc.).

  Deben incluirse los siguientes componentes en la pantalla principal:

- **Botones** que permitan realizar las siguientes acciones:

  - Botón para crear nuevo entorno: elimina los datos del anterior entorno (en caso de existir) y realiza todas las acciones del apartado: "Creación de nuestro entorno".

  - Botón para transcurrir un día: realizará todas las acciones del apartado: "Vida diaria en nuestro entorno".

    En caso de que llegue un día en el cuál el entorno se extinga, es decir, que no queden seres, debes mostrar un mensaje en pantalla avisando de la extinción total.

  - Botón para mostrar detalles: debe mostrarse una nueva pantalla sobre la principal con un resumen de nuestro entorno actual, con la siguiente información:
    - Fecha (número de días transcurridos desde su creación).
    - Desglose de los seres ordenados por grupos, y para cada uno de ellos el tipo de ser, su fecha de nacimiento (en número de día, para cualquier ser), su velodidad (en caso de los humanos), el número de vampiros matados (para los cazavampiros), número de humanos convertidos en vampiros (para los vampiros) y número de zombies que ha convertido (para los zombies).
    - Número de seres de cada grupo (separando humanos no cazavampiros, de los humanos cazavampiros).
    - Temperatura actual del entorno.
    - En caso de producirse un acontecimiento, debe mostrarse que está teniendo lugar. Por ejemplo: "Enfriamiento global".

  - Botón para "Avanzar 10 días": botón que permite avanzar 10 días en el simulador. En caso de que, en el transcurso de esos 10 días se extingan todos los seres, debes parar la simulación en ese día y mostrar el día (en número) en el que se produjo la extinción.

  - Un botón por cada acontecimientos, para producirlo al ser pulsado.
  - Botón para salir del programa

- Añade un caja de texto (textarea) en la pantalla principal, en la que, tras transcurrir cada día, muestre la información básica del entorno, es decir día actual, la temperatura y número de seres de cada grupo.

AYUDA

- Opcionalmente, pueden resultarte útiles los siguientes métodos que permiten mostrar ventanas de diálogo de forma gráfica:

  showMessageDialog: muestra un mensaje en pantalla
  showInputDialog: permite introducir un texto por teclado

  Podrás utilizar más métodos similares a los anteriores si lo crees conveniente.
  Para poder utilizarlos, tendrás que añadir en el programa la siguiente sentencia:

import javax.swing.JOptionPane;

- Implementa un método toString para mostrar los detalles de cada ser, así como otro toString que muestre el listado de todos los seres.

- Una vez funcione esta primera parte, podrás comprobar la evolución diaria de nuestro simulador. Lo que esperamos es que éste pueda perdurar en el tiempo, es decir, que todos los grupos de seres puedan perdurar a lo largo del paso del tiempo. Realiza la simulación con los parámetros exactos explicados en el enunciado.

PARTE 2

Resulta interesante que, al salir del programa, se almacene toda la información existente en memoria (todos los datos del entorno), y posteriormente pueda recuperarse esa información al arrancar de nuevo el programa para continuar con la simulación en el día en el que se quedó (como si no hubiéramos cerrado el programa). Podremos así cerrar el programa y volver a abrirlo para continuar con el estado del entorno previo al cierre del programa.

Una opción para realizar esto sería introducir toda la información en ficheros de texto. Sin embargo, esta operación sería muy costosa, ya que, por una parte, debería seguirse un formato muy estricto para almacenar la información, y por otra parte, el proceso de lectura, interpretación de los datos y carga en memoria que se ejecutaría al arrancar de nuevo el programa sería tedioso en cuanto a instrucciones necesarias.

Para conseguir este propósito de forma rápida y mucho más sencilla vamos a utilizar la *Serialización* que permite Java.

¿Qué permite la serialización?

La serialización permite convertir un objeto de Java en un fichero binario de bytes (con extensión .bin). Después se puede realizar la operación inversa, es decir, recuperar el objeto a partir del fichero de bytes. Por tanto, el proceso es cómodo y prácticamente automático, además de ser útil y muy apropiado para los requerimientos de esta parte 2 de la práctica.

Para poder usar la serialización, tanto la clase que contiene el ArrayList, como la clase de los objetos contenidos en el ArrayList, debe implementar el interfaz *serializable*.

Busca en Internet el resto de información necesaria para poder realizar la serialización en Java. Verás que es sencillo y muy útil. Con pocas líneas podremos conseguir nuestro propósito.

AYUDA: Generación de números aleatorios en Java

La clase *Random* proporciona un generador de números aleatorios.
Para obtener un número aleatorio debes seguir estos pasos:

1. Importar la clase*: import java.util.Random;*
2. Crear un objeto de la clase *Random*
3. Utilizar uno de los métodos de la clase para obtener el número

Podemos usar un constructor sin parámetros o bien pasarle una semilla. Si instanciamos varias veces la clase con la misma semilla, tendremos siempre la misma secuencia de números aleatorios (ya que los números que genera dependen de la semilla).

```
Random r1 = new Random();
Random r2 = new Random(4234);
Random r3 = new Random(4234);
// r2 y r3 darán la misma secuencia de números aleatorios
```

Lo más fácil es usar el constructor sin parámetros, que normalmente dará secuencias distintas de números aleatorios en cada instancia.

Una manera de obtener una semilla que sea distinta cada vez que ejecutemos nuestro programa, puede ser usar como semilla el tiempo actual en milisegundos del sistema, que puede obtenerse con *System.currentTimeMillis().* Esto permitirá obtener siempre números aleatorios distintos, porque la semilla nunca coincide.

Una vez creado el objeto, mostramos ahora algunos métodos de la clase *Random*:

nextInt() devuelve un número entero positivo o negativo dentro del rango de enteros.
nextInt(int n) devuelve un número entero >=0 y menor que n.
nextDouble() Devuelve un número positivo de tipo *double* mayor o igual que 0.0 y menor que 1.0

Por ejemplo, para generar un entero al azar:

```
System.out.println(r1.nextInt());
```

Para generar un entero entre 0 y 6:

```
System.out.println(r1.nextInt(7));
```

Para generar un entero entre 1 y 6:

```
System.out.println(r1.nextInt(6)+1);
```

En general, para generar enteros entre dos límites DESDE, HASTA, ambos incluidos:

```
r1.nextInt(HASTA-DESDE+1)+DESDE
```

Por ejemplo, para generar un número aleatorio entre 10 y 20:

```
System.out.println(r1.nextInt(20-10+1)+10);
```

OPCIONAL

Una vez finalizada la práctica, puedes añadir más detalles sobre la misma de forma opcional y voluntaria. Se sugiere:

- mejorar los parámetros probabilísticos para aumentar la estabilidad de todos los grupos de seres que componen nuestro entorno y aumentar el número de días de existencia,
  (si cambias los parámetros, debes permitir ejecutar el programa con los parámetros iniciales del enunciado, es decir, que puedes crear variantes pero mantener siempre la opción de lejecutar el entorno con los parámetros iniciales).
- mejorar la interfaz gráfica con imágenes para cada botón, fondo de la pantalla, audios...
- ampliar el programa con nuevos acontecimientos que inventes.
- realizar otras mejoras que consideres oportunas.