CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

ACTIVIDAD EVALUABLE

Programació
CFGS DAW

Autores:

Joan V. Cassany – jv.cassanycoscolla@edu.gva.es

Guillermo Garrido – g.garridoportes@edu.gva.es

2022/2023

Licencia

ACTIVIDAD EVALUABLE

'LA VIDA DE LOS MICROORGANISMOS'

1. INTRODUCCIÓN

El objetivo de este ejercicio consiste principalmente en hacerle utilizar el conjunto de conceptos aprendidos, tanto en la programación orientada a objetos como en la gestión de excepciones. Es por esto que en algunos casos se puede pensar que algunos elementos de los que se piden no son estrictamente necesarios o que se pueden aplicar soluciones más simples, pero con ello no conseguiríamos el objetivo y competencias que con este ejercicio se persiguen.

Por tanto, en este ejercicio se dan especificaciones muy concretas (sin indicar la solución) para que se utilicen las instrucciones y estructuras Java adecuadas. Es muy importante que las siga ya que la rubrica contemplará si se hace uso de éstas o no.

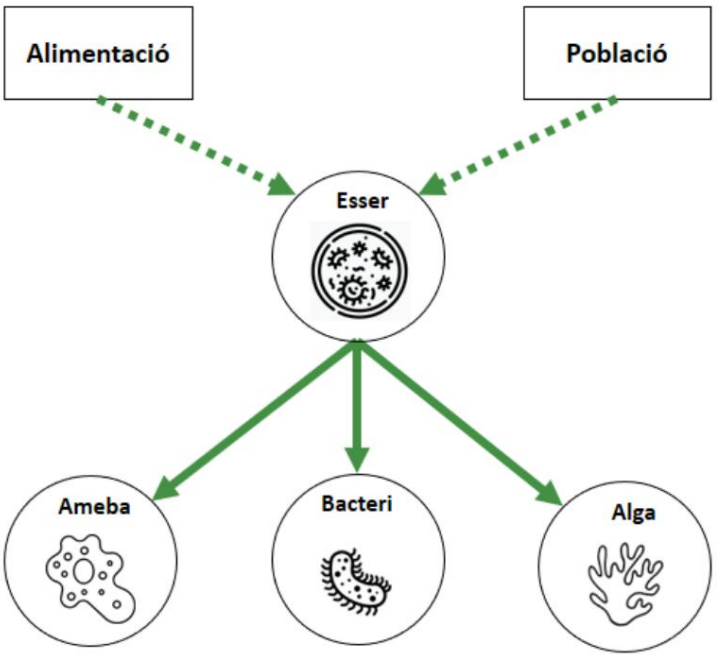
Igualmente, dada la complejidad de las dependencias, es importante que se ajuste a la nomenclatura específica que se proporciona a lo largo del ejercicio.

2. DESCRIPCIÓN DEL JUEGO: LA VIDA DE LOS MICROORGANISMOS

En este proyecto vamos a construir un ecosistema de seres vivos o microorganismos que interactuarán entre sí para alimentarse y para reproducirse. Todos los microorganismos pertenecen a un grupo común de seres que establece un conjunto de elementos comunes, que pueden venir definidos o dejar en manos de los distintos grupos de eses su definición.

Así pues, nuestro ecosistema de microorganismos o Essers estará formado por Amebes, Bacterias y Algas. Además existirán unos nutrientes de los que se alimentará la base de la cadena para ir engordando a nuestro ecosistema.

Toda la gestión de este ecosistema se realizará desde un programa principal con el que el usuario interactuará para consultar o provocar cambios en el mismo.



3. ESTRUCTURA BASE

Definiremos dos esquemas desde donde se establecerán determinados atributos y métodos que el resto de componentes tendrán que implementar. De esta forma podremos variar el comportamiento del ecosistema de forma fácil simplemente cambiando algunos valores.

3.1.ALIMENTO

Limitará a que el proyecto sólo pueda utilizar los siguientes alimentos: Todo, Ameba, Bacteria, Alga, Nutriente.

3.2.ALIMENTACIÓN

Se definirán las constantes de los pesos que se utilizarán en el proceso de creación de los distintos tipos de seres:

- Peso de la ameba=20
- Peso de la bacteria=10
- Peso del alga=3
- Peso de los nutrientes=5

También, indicará los métodos que todos los seres tendrán que implementar cuando se programen:

- comida(...): donde cada tipo de ser especificará cómo lleva a cabo la alimentación (a quién se come y cómo).
Por tanto, será necesario tener información de entrada al método sobre la lista completa de seres.

3.3.POBLACIÓN

Se definirá una constante multiplicadora para el peso de reproducción y otra para el máximo de seres de cada tipo que existirá.

- Peso reproducción=3
- Número máximo=20

También, indicará los métodos que todos los seres tendrán que implementar cuando se programen:

- reducirPoblacio(), que tendrá como finalidad que cada tipo de ser cuando sea comido reduzca su propio número total de individuos y el general.

- reproducir(...), que se utilizará para que cada tipo de organismos defina su método de reproducción y pueda incluir un nuevo ser de su tipo en la lista de seres, por lo que ésta formará parte de las datos de entrada al método.

4. LA SUPERCLASE: ESER

Será la superclase a partir de la cual se definirá el resto. Adoptará los esquemas que se hayan definido en el proyecto, tanto para su alimentación como para su reproducción.

Esta superclase podrá heredarse pero no instanciarse. En ella se establecerá la base de las definiciones del comportamiento que tendrán los eses. En algunos casos lo definirá completamente, en otros obligará a que las subclases implementen esos comportamientos.

4.1.ATRIBUTOS DE CLASE

Se definirán estos atributos.

- totalEssers, que se incrementará cuando se cree algún nuevo microorganismo y se reducirá cuando desaparezca. Hay que tener en cuenta que la desaparición de un ser deberá reducir tanto su contador (que se verá en su momento) como ese total de seres.

- consecutivo, que se irá incrementando a cada nuevo microorganismo que se cree y que se utilizará para que forme parte del texto que se asignará al nombre (atributo que se cita después). Ejemplo: AMEBA1, ALGA2, ALGA3, BACTERIO4, etc.

Estos valores sólo se pueden modificar o acceder desde la propia clase.

4.2.ATRIBUTOS DE INSTANCIA

Se definirán los siguientes atributos de instancia.

- nombre (String): texto de escritor que no se podrá modificar una vez asignado. - peso (entero): que podrá variar en cualquier momento (a la comida o reproducirse).

Estos atributos sólo podrán ser accedidos o modificados desde esta clase Essers.

4.3.CONSTRUCTOR

- El nombre se formará por la concatenación de un texto que vendrá desde las subclases (según el tipo de ser que corresponda) más el consecutivo de seres (por ejemplo AMEBA1).

- El peso deberá incrementarse en la cantidad que venga indicada según el tipo de ser que se haya creado.

No deberá olvidarse en este punto actualizar los atributos de clase que corresponda.

4.4.MÉTODOS

4.4.1.MÉTODOS QUE NO PODRAN SER SOBREESCRITOS POR LAS SUBCLAS

- String dirNom(): nos dará el nombre.

- int dirPes(): nos dará el peso.

- cambiaPes(): incrementará (o decrementará) el peso del objeto según el valor que le llegue.

4.4.2. MÉTODOS QUE NO PODRAN SER SOBREESCRITOS POR LAS SUBCLASES Y QUE SÓLO SE PUEDEN UTILIZAR A TRAVÉS DE LA CLASE

- int GeneraAleatori (inicial, cantidad): generará un número aleatorio entero. Este método podrá utilizarse desde cualquier otra clase (ya sea subclase o programa principal). Se utilizará para generar valores aleatorios dependiendo de la funcionalidad que se esté implementando. El proceso que llame a este método indicará el valor inicial y la cantidad de valores que necesita para obtener un valor aleatorio.

Nota: Aunque esta clase no sea el mejor sitio para ubicar este método, lo haremos así por cuestiones didácticas a la hora de realizar las llamadas.

4.4.3. MÉTODOS QUE SE IMPLEMENTARAN EN TODAS LAS CLASES Y QUE MOSTRARÁN INFORMACIÓN DE LA CLASE NO DE LA INSTANCIA

Por tanto, estos métodos no deben permitir la sobreescritura al declararse.

- int dirPoblacio(): nos indicará el total de individuos de esta clase.

4.4.4. MÉTODOS QUE SE OBLIGARÁ A LAS SUBCLASES A IMPLEMENTAR

Por tanto, no se requiere una definición detallada de su comportamiento en este punto.

- String mostrarEstado(): mostrará información resumida del objeto.

- String mostrarDetall(): mostrará información detallada del objeto.

4.4.5. MÉTODOS QUE TODA CLASE DEBE CONTENER

Aquí se incluirán todos los métodos que nos hallamos obligados a crear por definición.

Sólo se especificará el comportamiento de aquellos métodos que tenga sentido en este nivel de clases.

5. LA CLASE: AMEBA

Heredará de la superclase. Podrá crear instancias pero no permitirá la creación de subclases.

5.1.ATRIBUTOS DE CLASE

Se definirán estos atributos.

- totalAmebas que se incrementará cuando se cree una nueva y se reducirá cuando desaparezca. Debe tenerse en cuenta que la desaparición de una ameba deberá reducir tanto su acumulador como el general.

Estos valores sólo se pueden modificar o acceder desde la propia clase.

5.2.ATRIBUTOS DE INSTANCIA

Además de los atributos heredados, tendrá otros dos atributos.

- alimento (Aliment): indicará cuál es el tipo de alimentación que tiene. -
reproducción (boolea): que nos indicará si puede reproducirse.

Estos atributos sólo podrán ser accedidos o modificados desde esta clase Ameba.

5.3.CONSTRUCTOR

- el nombre de las amebas empezará con la palabra «AMEBA».

- el peso de las amebas será el determinado como constante.

- alimento, las amebas se lo comen todo.

- reproducción, las amebas se reproducen.

No deberá olvidarse en este punto actualizar los atributos de clase que corresponda.

5.4.MÉTODOS

5.4.1. MÉTODOS QUE SE IMPLEMENTARAN EN TODAS LAS CLASES Y QUE MOSTRARÁN INFORMACIÓN DE LA CLASE NO DE LA INSTANCIA

Por tanto, estos métodos no deben permitir la sobreescritura al declararse.

- int dirPoblacio(): nos indicará el total de individuos de esta clase.

5.4.2. MÉTODOS HEREDADES QUE SE DEBEN DEFINIR

- String mostrarEstado(): para las amebas se mostrará un mensaje con el siguiente texto

"@ " + nombre + " => PESO: " + peso

- String mostrarDetall(): se mostrará un mensaje con el siguiente texto

"@ " + nombre + " => PESO " + peso + " - ALIMENTACIÓN: " + alimento + " - REPRODUCCIÓN: " + "SI" o"NO"

- comida(...)

• Elegirá aleatoriamente un ser cualquiera de la lista que no sea ella misma. • Incrementará su peso con el peso del ser comido. • Informará de quién es la víctima y quién se ha comido con el texto "ALIMENTACIÓN **** " + nombre + ": me he comido a • Reducirá los totales de población que corresponda. • Eliminará de la lista a " + nombre víctima + ". Ahora pese " + peso la víctima .

- reproducir(...)

• Si su peso es mayor o igual al peso base de la ameba multiplicado por el peso de reproducción llevará a cabo la reproducción.
• Creará una nueva ameba.
• Reducirá su peso con lo que se ha quedado la nueva ameba. • Nos informará con el texto "REPRODUCCIÓN ****" + nombre + " me he reproducido y he creado en " + nombre nueva + ". Ahora pese " + peso
• Si no tiene el peso adecuado, simplemente nos dirá que no puede reproducirse todavía con el texto no me puedo "REPRODUCCIÓN ****" + nombre + " con un peso de " + peso + " reproducir") (1)

(1) NOTA: Este último caso puede resolverse por mensaje o por excepción. La puntuación será diferente según se opte por una solución u otra. Revise la rúbrica.

6. LA CLASE: BACTERIO

Heredará de la superclase. Podrá crear instancias pero no permitirá la creación de subclases.

6.1.ATRIBUTOS DE CLASE

Se definirán estos atributos.

- totalBacterias que se incrementará cuando se cree un nuevo y se reducirá cuando desaparezca. Debe tenerse en cuenta que la desaparición de una bacteria deberá reducir tanto su acumulador como el general.

Estos valores sólo se pueden modificar o acceder desde la propia clase.

6.2.ATRIBUTOS DE INSTANCIA

Además de los atributos heredados, tendrá otros dos atributos.

- alimento (Aliment): indicará cuál es el tipo de alimentación que tiene -
reproducción (boolea): que nos indicará si puede reproducirse

Estos atributos sólo podrán ser accedidos o modificados desde esta clase Bacteria.

6.3.CONSTRUCTOR

- el nombre de las amebas empezará con la palabra «BACTERIO».

- el peso de las bacterias será el determinado como constante.

- alimento, las bacterias comen algas.

- reproduccion, las bacterias se reproducen.

No deberá olvidarse en este punto actualizar los atributos de clase que corresponda.

6.4.MÉTODOS

6.4.1. MÉTODOS QUE SE IMPLEMENTARAN EN TODAS LAS CLASES Y QUE MOSTRARÁN INFORMACIÓN DE LA CLASE NO DE LA INSTANCIA

Por tanto, estos métodos no deben permitir la sobreescritura al declararse.

- int dirPoblacio(): nos indicará el total de individuos de esta clase.

6.4.2. MÉTODOS HEREDADES QUE SE DEBEN DEFINIR

- String mostrarEstado(): para las bacterias se mostrará un mensaje con el siguiente texto
"/ " + nombre + " => PESO: " + peso

- String mostrarDetall(): se mostrará un mensaje con el siguiente texto
"/ " + nombre + " => PESO " + peso + " - ALIMENTACIÓN: " + alimento + " - REPRODUCCIÓN: " + "SI" o"NO"

- comida(...)

• Las bacterias sólo comen algas. Si no quedan dará un mensaje de ancianos. (2) Si quedan elegirá un alga aleatoriamente de la lista. • Incrementará su peso con el peso del ser comido. • Informará de quién es la víctima y quién se ha comido con el texto "ALIMENTACIÓN **** " + nombre + ": me he comido a • Reducirá los totales de población que corresponda. • Eliminará de la lista a la víctima . " + nombre víctima + ". Ahora pese " + peso

(2) NOTA: Este último caso puede resolverse por mensaje o por excepción. La puntuación será diferente según se opte por una solución u otra. Revise la rúbrica.

- reproducir(...)

• Las bacterias se dividen en dos hijos de igual tamaño. • Si su peso es mayor o igual al peso de reproducción multiplicado por el doble del peso base de la bacteria se llevará a cabo la reproducción. • Creará una nueva bacteria con la mitad de su peso. • Reducirá su peso con lo que se ha quedado la nueva bacteria. • Nos informará con el texto "REPRODUCCIÓN ****"

+ nombre + " me he reproducido y he creado en " + nombre nuevo + ". Ahora pese " + peso

• Si no tiene el peso adecuado, simplemente nos dirá que no puede reproducirse todavía con el texto no me puedo "REPRODUCCIÓN ****" + nombre + " con un peso de " + peso + " reproducir") (3)

(3) NOTA: Este último caso se puede resolver por mensaje o por excepción. La puntuación será diferente según se opte por una solución u otra. Revise la rúbrica.

7. LA CLASE: ALGA

Heredará de la superclase. Podrá crear instancias pero no permitirá la creación de subclases.

7.1.ATRIBUTOS DE CLASE

Se definirán estos atributos.

- totalAlgas que se incrementará cuando se cree una nueva y se reducirá cuando desaparezca. Debe tenerse en cuenta que la desaparición de un alga deberá reducir tanto su acumulador como el general.

Estos valores sólo se pueden modificar o acceder desde la propia clase.

7.2.ATRIBUTOS DE INSTANCIA

Además de los atributos heredados, tendrá otros dos atributos.

- alimento (Aliment): indicará cuál es el tipo de alimentación que tiene -
reproducción (boolea): que nos indicará si puede reproducirse

Estos atributos sólo podrán ser accedidos o modificados desde esta clase Alga.

7.3.CONSTRUCTOR

- el nombre de las amebas empezará con la palabra «BACTERIO».

- el peso de las algas será el determinado como constante.

- alimento, las algas comen nutrientes.

- reproduccion, las algas se reproducen.

No deberá olvidarse en este punto actualizar los atributos de clase que corresponda.

7.4.MÉTODOS

7.4.1. MÉTODOS QUE SE IMPLEMENTARAN EN TODAS LAS CLASES Y QUE MOSTRARÁN INFORMACIÓN DE LA CLASE NO DE LA INSTANCIA

Por tanto, estos métodos no deben permitir la sobreescritura al declararse.

- int dirPoblacio(): nos indicará el total de individuos de esta clase.

7.4.2. MÉTODOS HEREDADES QUE SE DEBEN DEFINIR

- String mostrarEstado(): para las algas se mostrará un mensaje con el siguiente texto

"#" + nombre + " => PESO: " + peso

- String mostrarDetall(): se mostrará un mensaje con el siguiente texto

"# " " + nombre + => PESO " + peso + " - ALIMENTACIÓN: " + alimento + " - REPRODUCCIÓN: " + "SI" o"NO"

- comida(...)

• Las algas sólo comen nutrientes. Básicamente lo que se hace es incorporar el peso de un nutriente al peso del alga.
• Incrementará su peso con el peso constante del nutriente que se ha comido. • Informará de quién es la víctima ya quién se ha comido con el texto "ALIMENTACIÓN ****" + nombre + ": me he comido en " + nombre víctima + ". Ahora pese • " + peso
Si el peso actual del alga es mayor que el doble del peso base del alga por la constante peso de reproducción se reproducirá.

- reproducir(...)

Las algas pueden reproducirse más de una vez en función del peso que tengan. Por tanto, generará algas (hará todas las acciones del proceso de reproducción) mientras su peso sea mayor o igual al peso base del alga por la constante del peso de reproducción.

• Creará una nueva alga. •
Reducirá su peso con peso base de la nueva alga creada. • Nos informará con el texto "REPRODUCCIÓN ****" + nombre + " me he reproducido y he creado en " + nombre nuevo + ". Ahora pese " + peso
• Si no tiene el peso adecuado, simplemente nos dirá que no puede reproducirse todavía con el texto no me puedo "REPRODUCCIÓN ****" + nombre + " con un peso de " + peso + " reproducir") (4)

(4) NOTA: Este último caso se puede resolver por mensaje o por excepción. La puntuación será diferente según se opte por una solución u otra. Revise la rúbrica.

8. PROGRAMA PRINCIPAL: LAVIDADELSMICROORGANISMOS

8.1.CREACIÓN DEL ECOSISTEMA

Se creará una lista de seres. Tenga en cuenta el tipo que debe tener la lista para poder incorporar objetos de todas las clases.

Se pedirá al usuario el número de seres que desea tener de cada tipo:

- Introduce el número de amebas (entre 1 y (constante número máximo de seres)):
- Introduce el mínimo de bacterias (entre 1 y (constante número máximo de seres)):
- Introduce el mínimo de algas (entre 1 y (constante número máximo de seres)):

8.2.MÉTODO leerNumero()

Para leer cada uno de estos valores se creará la función leerNumero(...) que valide mediante un contexto de excepciones que se introduzca un dato numérico y que esté entre los límites indicados. La función sólo finalizará cuando el dato sea correcto.

Si el número no es correcto dará el mensaje:

"ERROR: Debe introducir un número entre " + min + " , " + max + "."

Si no se ha introducido un valor numérico el mensaje será:

"ERROR: No se ha introducido un valor numérico."

8.3.MÉTODO crearEssers()

Una vez tengamos las cantidades de cada ser a crear, invocaremos procedimiento crear Seres (...) que, a partir de la lista vacía de seres y de los valores indicados por el usuario creará de forma aleatoria los seres de cada tipo que corresponda.

Se plantean dos alternativas para crear la lista.

Alternativa 1: Los seres se introducirán en la lista de forma contigua (primero todos los de un tipo, después otro, etc.).

Alternativa 2: Los seres se introducirán en la lista de forma aleatoria usando la función generaAleatorio() utilizando 3 valores para elegir en cada momento qué tipo de ser se inserta.

Nota: Las dos opciones serán válidas, pero la alternativa 2 valdrá 1 PUNTO EXTRA por la dificultad que comporta.

8.4.MÉTODO procesaMenu()

Una vez ya tenemos creada la lista inicial de seres daremos paso a la interacción con el usuario mediante un menú que gestionaremos en el método processaMenu(...).

Se mostrará al usuario un mensaje con las opciones que puede elegir.

«OPCIONES==> 1.-Una Interacción, 2.-Diez Interacciones, 3.-Listado, 4-Detalle 0.-Salir: «

Se validará la información introducida utilizando la función ya vista leerNumero(...).

Opción 1: se llamará al método produceInteraccio (..) que veremos a continuación para provocar una interacción entre los seres que tenemos.

Opción 2: se llamará al método produceInteraccio (..) que veremos a continuación 10 veces para provocar una evolución rápida de los microorganismos.

Opción 3: se mostrará un listado del estado en el que se encuentra actualmente el ecosistema haciendo uso del método muestraListaEssers (...) que veremos a continuación.

Opción 4: se mostrará el detalle del ser que elija el usuario haciendo uso del método mostrarEsser(...) que veremos a continuación.

Opción 0: se finalizará el bucle e iremos al final del programa para mostrar la lista de seres que queda haciendo uso del método muestraListaEssers (...) y dar un mensaje de despido.

8.5.MÉTODO muestraListaEssers()

Recorrerá la lista de seres mostrándonos información del estado de cada uno de ellos.

Por último, dará un mensaje resumen del estado actual del conjunto de microorganismos que tenemos:

"POBLACIÓN: TOTAL SERES=>" + total de eses + ", AMEBES=>" + total de amebas + ", BACTERÍAS=>" + total de bacterias + ", ALGAS=>" + total de algas

8.6.MÉTODO muestraEser()

Se proponen dos alternativas entre las que puede elegir.

Alternativa 1: se pedirá un valor numérico entero para acceder por índice a una posición de la lista y mostrar el detalle de este microorganismo. Se validará que el número esté en los límites correctos para acceder a la lista haciendo uso de un método ya conocido.

Alternativa 2: El usuario introducirá un nombre por teclado y se buscará en la lista para proporcionar la información detallada de este microorganismo. Si no existe ese nombre en la lista se dará un error.

«ERROR: el microorganismo» + nombre + « no aparece en la lista.»

Nota: Ambas opciones serán válidas, pero la alternativa 2 valdrá 0,5 PUNTO EXTRA por la dificultad que comporta.

8.7.MÉTODO produceInteracción()

Se elegirá un ser de la lista al azar utilizando la función genera Aleatorio(...).

Si el elemento elegido es una instancia de una ameba o de una bacteria podrá comer o reproducirse, por tanto generaremos otro número aleatorio para que se opte por una u otra opción.

Si el elemento elegido es una instancia de un alga, sólo puede comer. Recuerde que las algas se reproducen, si es necesario, cuando comen.

9. EJEMPLO DE EJECUCIÓN

```
Introduce el número de amebas (entre 1 y 20): 4

Introduce el número de bacterias (entre 1 y 20): 3

Introduce el número de algas (entre 1 y 20): 5


 OPCIONES==> 1.-Una Interacción, 2.-Diez Interacciones, 3.-Listado, 4-Detalle 0.-Salir: 3 /
BACTERIO1 => PESO: 10
# ALGA2 => PESO: 2
@ AMEBA3 => PESO: 20
/ BACTERIO4 => PESO: 10
@ AMEBA5 => PESO: 20
@ AMEBA6 => PESO: 20
@ AMEBA7 => PESO: 20
/ BACTERIO8 => PESO: 10
# ALGA9 => PESO: 2
# ALGA10 => PESO: 2
# ALGA11 => PESO: 2
# ALGA12 => PESO: 2
 POBLACIÓN: TOTAL SERES=>12, AMEBES=>4, BACTERIOS=>3, ALGAS=>5

 OPCIONES==> 1.-Una Interacción, 2.-Diez Interacciones, 3.-Listado, 4-Detalle 0.-Salir: 2
REPRODUCCIÓN **** AMEBA5 con un peso de 20 no me puedo reproducir
ALIMENTACIÓN **** ALGA10: me he comido un nutriente que pesa 3. Ahora peso 5
REPRODUCCIÓN **** BACTERI8 con un peso de 10 no me puedo reproducir
ALIMENTACIÓN **** AMEBA5: me he comido en ALGA12. Ahora peso 22
ALIMENTACIÓN **** BACTERIO4: me he comido en ALGA10. Ahora peso 15
ALIMENTACIÓN **** AMEBA7: me he comido en AMEBA5. Ahora peso 42
ALIMENTACIÓN **** BACTERIO1: me he comido en ALGA11. Ahora peso 12
ALIMENTACIÓN **** BACTERIO1: me he comido en ALGA9. Ahora peso 14
ALIMENTACIÓN **** AMEBA3: me he comido en AMEBA6. Ahora peso 40
ALIMENTACIÓN **** AMEBA3: me he comido en ALGA2. Ahora peso 42

 OPCIONES==> 1.-Una Interacción, 2.-Diez Interacciones, 3.-Listado, 4-Detalle 0.-Salir: 3 /
BACTERIO1 => PESO: 14
@ AMEBA3 => PESO: 42 /
BACTERIO4 => PESO: 15
@ AMEBA7 => PESO: 42 /
BACTERIO8 => PESO: 10
 POBLACIÓN: TOTAL SERES=>5, AMEBES=>2, BACTERIOS=>3, ALGAS=>0

 OPCIONES==> 1.-Una Interacción, 2.-Diez Interacciones, 3.-Listado, 4-Detalle 0.-Salir:
```

10.RUBRICA

Clase Ser puntuará 1 PUNTO.

Clase Ameba puntuará 1,5 PUNTOS.

Clase Bacteria puntuará 1,5 PUNTOS.

Clase Alga puntuará 1,5 PUNTOS.

Programa Principal puntuará 1 PUNTO.

En cada clase se asignará 0,5 puntos de los 1,5 puntos indicados en la correcta declaración de las clases y los métodos (herencia, polimorfismo, privacidad, etc.)

En las subclases se asignará 0,5 puntos de los 1,5 puntos indicados en el correcto funcionamiento de los métodos comer y reproducir.

Con las consideraciones anteriores la nota máxima computada a quitar será de 6,5.

Si se incluye la gestión de excepciones se contará con 0,5 PUNTOS EXTRA por cada excepción (hay 4 puntos en los que se ha propuesto la opción de incluir una excepción).

Si se opta por la alternativa 2 en la creación de la lista se contará con 1 PUNTO EXTRA.

Si se opta por la alternativa 2 al mostrar el detalle de un ser se contará con 0,5 PUNTOS EXTRA.

Se recomienda iniciar el ejercicio con las alternativas más sencillas y cuando se tenga el proyecto ya funcionando, si se dispone de tiempo, se hagan las modificaciones necesarias para incorporar las alternativas complejas y mejorar la nota.

Se evaluará también la adecuada inclusión de comentarios, ordenación del código, seguimiento de la nomenclatura propuesta y la programación con un código eficiente.