CEU

Aplicación para gestionar un Animalario II

Angelo De Nadai

Análisis y descripción de la aplicación:

La aplicación se ha dividido en cuatro paquetes:

animalario: se ocupa de gestionar los archivos Java relativos a las entidades principales del proyecto, todos los atributos están con protección "privada" para que no sean directamente accesibles desde el exterior , pero solo a través de los métodos establecidos de las clases.

Las clases son:

-Raton: representa el objeto raton con atributos (codigo, fecha de nacimiento, peso, sexo, temperatura, descripcion, cromosoma 1 y cromosoma 2), la clase permite lectura desde el exterior para todos los atributos, y la modificación solo por el peso, la temperatura y las mutaciones de los cromosomas

Hay tres funciones en la clase:

esEsteril = verifica la esterilidad del ratón devolviendo un booleano

esPolygamous = comprobar la poligamia del ratón devolviendo un booleano

esMaduro = Comproba si el ratón es maduro(edad > 45 días) comprobando con la fecha ingresada devolviendo un booleano

-Población: representa el objeto de una Población de Ratones con atributos (nombre, administrador, número de días pasados en el animalario, matriz de ratones machos, matriz de ratones hembra, matriz de familias), la clase permite que todos los atributos se lean desde el exterior

En la clase hay las siguientes funciones:

anadirRaton= Añade un nuevo ratón a una población ya existente

anadirFamilia=Se agrega una familia alla matriz familias si no es igual a null

ordenaAlfabeticamente=Ordenar una lista de ratones alfabéticamente(codigos)

ordenaPorFecha=Ordenar una lista de ratones por fecha nacimiento

ordenaPorPeso=Ordenar una lista de ratones por Peso(descendiendo)

getCodigosRatones=Lista todos los codigos de los ratones de una población

eliminarRaton= Elimina un ratón de una población indicando su código de referencia

creaFamilias=genera familias basadas en los ratones presentes en la población

procrearFamilias=Genera familias aleatorias con los ratones de la población

simulacionMontecarlo=Simula la evolución a lo largo del tiempo de la población, generando familias procreandolas

-Familia(abstract): representa el objeto de una Población de Ratones con atributos (padre y matriz de hijos), la clase permite que todos los atributos se lean desde el exterior

En la clase hay las siguientes funciones:

procrear=Genera hijos causal(ratones) a partir del padre ratón y la madre ratón

anadirMadre=metodo abstracto para añadir una madre ratón a la familia

anadirHijo=Añade un hijo ratón a la familia

-FamiliaMachoEsteril: Este Objeto estende la clase Familia, hereda los atributos y métodos del objeto Familia y los sobrescribe.

-FamiliaNormal: Este Objeto estende la clase Familia, hereda los atributos y métodos del objeto Familia y los sobrescribe.

-FamiliaPoligamica: Este Objeto estende la clase Familia, hereda los atributos y métodos del objeto Familia y los sobrescribe.

-Cromosoma: representa el objeto de un Cromosoma de un Ratón con atributos (Tipo de Cromosoma y mutación), la clase permite leer desde el exterior para todos los atributos, y mientras que la escritura no es posible para el atributo Tipo de Cromosoma

-TipoCromosoma: representa el objeto TipoCromosoma Enum que puede asumir los valores X o Y

-Sexo: representa el objeto Enum Sexo el cual puede asumir los valores MASCULINO o FEMENINO

auxialiar: contiene la clase "FuncionesAuxiliares" donde se guardan para fines organizativos las funciones del proyecto que interactúan con el usuario y que no pertenecen directamente a un objeto, son:

abrirArchivo = Abre el archivo especificado como parámetro y devuelve la población de ratones guardada dentro

guardarArchivo = Guardar una población en un archivo ya abierto

guardarArchivoComo = Guardar una población en un nuevo archivo

editarRaton = Modificar los parámetros de un ratòn

creaPoblacion = Crea una población de ratones

creaRaton = Crear un raton

creaPoblacionVirtual=Crea una población basada en los datos que proporcionas

imprimirEstadisticas=Convierte una matriz de enteros de estadísticas en una Stringa

listarRatonesOrdenados=pregunta en qué orden ordenar los ratones de la población pasada como parámetro devuelve la lista ordenada de ratones

main: contiene solo la clase Main donde se muestra un menú práctico en ejecución para interactuar con la aplicación

excepciones:
El paquete contiene las excepciones creadas específicamente para el manejo de errores de la aplicación, son:
NumDiasException= error que se puede generar al intentar modificar el atributo "días en el animalario" de la clase Población, si se ingresa un valor inaceptable, es decir, un número de días negativo o mayor a 270
RatonNoEncontrado= error que se puede generar si se intenta eliminar un mouse con código inexistente en el método "eliminaRaton" de la clase Población
NegativeInputExeption=Se genera un error si se ingresa un número negativo en un campo donde se requiere un valor positivo

Decisiones de diseño:

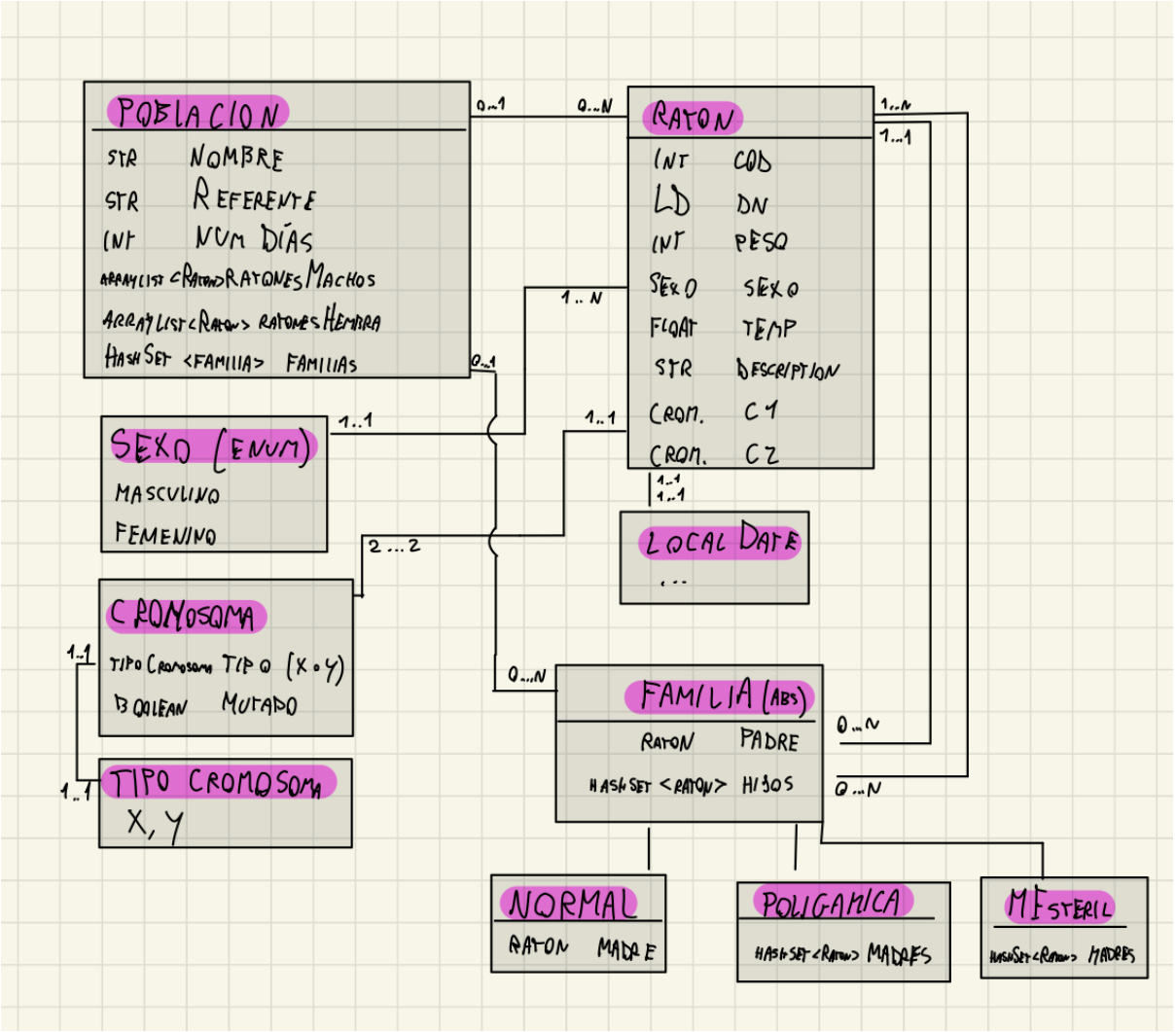
La aplicación se desarrolló a partir del diseño UML del proyecto, a partir del cual se decidió utilizar 9 entidades, una de las cuales ya ha sido implementada en java por la clase java.time.LocalDate. Todas las entidades han sido insertadas en el paquete animalario y están conectadas como se representa en el diagrama UML, en particular las clases Localdate, Sexo y Chromosome son atributos de la clase Raton; ChromosomeType es un atributo de Chromosome; mientras que Raton se usa para crear un matriz, que corresponde a un atributo de Población, los tipos de Familias son atribuitos del objeto Poblacio

Comprobaciones de integridad y excepciones:

Los principales controles presentes en la aplicación están sobre los datos de entrada del terminal, de hecho siempre hay un control try-catch para errores de entrada (InputMismatchException), además hay controles sobre la correcta compilación y modificación de los objetos, implementando ambos predefinidos java, que crea clases personalizadas para los tipos de errores que pueden ocurrir, en este caso me refiero a NumDiasException y RatonNoEncontrado, cuyas especificaciones están presentes en la sección Excepciones.
Otras exceptiones son:NullPointerException, NegativeInputExeption y ClassNotFoundException implementadas para gestionar funciones añadidas en el proyecto.

Técnicas de clasificación y búsqueda:

Para la ordenacion, utilicé el algoritmo quickSort para el que puede ordenar muchos objetos, por lo que logra ser el más eficiente.



Conclusiones:

El proyecto me absorbió mucho, trabajé todos los días de 5 a 10 horas, durante una semana, invirtiendo mucho tiempo en la planificación, en el desarrollo de algoritmos de ordenación y guardado de archivos y en el debugging, en general estoy satisfecho con el producto obtenido .