Los Becarios
2ºDAM

Manual técnico piscifactoría

Breogán Fernández Tacón
Nicolás Iglesias Solla
Cristian Juvino Soto

Simulador:

Simulador:

Clase principal del programa, que ejecuta toda la lógica.

Atributos

dias(int):
Días avanzados en el sistema.

nombreEmpresa(String):
Nombre con el que se inicia el sistema.

almacenCentral(AlmacenCentral):
Objeto AlmacenCentral, objeto implementado como singleton, que representa el almacén compartido de comida y se utiliza para el manejo de inventario de alimentos en la piscifactoría.

piscifactorias(Arraylist<Piscifactoria>):
Lista de las piscifactorías del sistema.

monedero (Monedero):
Objeto Monedero, también singleton, que se encarga de gestionar las monedas generadas por la piscifactoría.

stats(estadisticas.Estadisticas):
Objeto Estadisticas, que se encarga de gestionar el número de peces que se venden y nacen.

Constructor

Simulador()

Crea un objeto de la clase Simulador para poder utilizarlo en el main del programa.

Métodos

init:
public void init()
Inicializa el sistema, asignando el nombre, creando una piscifactoría y añadiendo 100 monedas.

menu:
public void menu()
Muestra un menú con las opciones a realizar en el sistema.

menuPisc:
public void menuPisc()
Muestra la lista de piscifactorías actuales en forma de menú, más una opción 0 para salir.

selectPisc:
public int selectPisc()
Muestra el menú de piscifactorías y permite seleccionar una de ellas.

selectTank:
public Tanque selectTank()
Muestra el menú de tanques y permite seleccionar un tanque, mostrando un menú de los disponibles.

showGeneralStatus:
public void showGeneralStatus()
Muestra el día actual, las monedas del sistema, el estado de todas las piscifactorías del sistema y en caso de tener la mejora del almacén central, muestra el estado de este también.

showSpecificStatus:
public void showSpecificStatus()
Permite seleccionar una piscifactoría y muestra su estado.

showTankStatus:
public void showTankStatus()
Permite seleccionar un tanque y muestra su estado.

showStats:
public void showStats()
Muestra el estado de todos los peces del sistema el número de ellos nacidos, el número de vendidos y las monedas obtenidas con esto,y por último un mensaje del total de estos datos en toda la piscifactoría.

showIctio:
public Pez showIctio()
Muestra una lista de los peces disponibles en el sistema y permite elegir uno.

nextDay:
public void nextDay()
Avanza un día en el sistema, realiza el crecimiento de los peces, la reproducción y la venta de peces óptimos y por último muestra un mensaje de los peces vendidos en todo el sistema y las monedas obtenidas con ello.

addFood:
public void addFood()
Permite seleccionar una piscifactoría, seleccionar el tipo de comida que quieres añadir, seleccionar la cantidad de comida, y la añade.

addFish:
public void addFish()
Muestra las piscifactorías del sistema permite elegir una, después muestra los tanques del sistema y permite elegir uno y después muestra los posibles peces posibles a añadir en ese tranque, permite elegir uno y lo añade.

sell:
public void sell()
Permite seleccionar una piscifactoría y vende todos los peces adultos de esta, mostrando al final un mensaje de los peces vendidos y las monedas obtenidas con ello.

cleanTank:
public void cleanTank()
Elimina los peces muertos de todos los tanques.

emptyTank:
public void emptyTank()
Permite seleccionar un tanque, y elimina todos los peces del mismo independientemente de su estado.

upgrade:
public void upgrade()
Muestra un menú de mejoras disponibles para el sistema.

comidaAnimalVacia:
public boolean comidaAnimalVacia()
Verifica si la comida animal se ha agotado. Retorna true si la cantidad de comida es cero, de lo contrario, false.

infoLib:
public void infoLib(String tipoPez)
Devuelve la información de la librería de un tipo de pez específico, pasado por parámetro.

pecesVivosEnSist:
public int pecesVivosEnSist()
Devuelve el número de peces vivos en el sistema.

espacioEnPisci:
public int espacioEnPisci(Piscifactoria pisci)

Parámetros:
- Piscifactoría pisci: Objeto Piscifactoría.

Devuelve el total de espacios de una piscifactoría.

pecesTotalesEnSist:
public int pecesTotalesEnSist()
Devuelve el número de peces en el sistema.

pecesEnPiscifactoria:
public int pecesEnPiscifactoria()
Calcula y retorna el número total de peces en la piscifactoría.

espacioTotalSist:
public int espacioTotalSist()
Devuelve el número de espacios disponibles en el sistema.

elegirComida:
public int[] elegirComida(Piscifactoria pisc)

Parámetros:
- Piscifactoría pisci: Objeto Piscifactoría.

Permite elegir el tipo de comida y la cantidad que quieres añadir.

añadirComidaPisc:
public void añadirComidaPisc(Piscifactoria pisc)

Parámetros:
- Piscifactoría pisci: Objeto Piscifactoría.

Añade a una piscifactoría la cantidad del tipo de comida pasado por parámetro

AñadirComidaAlm:
public void AñadirComidaAlm()
Añade al almacén central la cantidad del tipo de comida pasado por parámetro.

selectRecompensa:
public void selectRecompensa()
Permite seleccionar una recompensa disponible, primero las lista y el usuario elige la que quiere.

truco98:
public void truco98()
Añade cuatro peces aleatorios a una piscifactoría seleccionada.

truco97:
public void truco97(String nombreArchivo, int nivel)
Metodo de prueba que permite crear una recompensa.

Parámetros:

- nombreArchivo (String): nombre del archivo.
- nivel (int): nivel de la recompensa.

public static void main(String[] args):
Ejecuta toda la lógica del programa.

Piscifactoría:

Piscifactoría:

Clase que representa una piscifactoría, contiene uno o varios tanques de peces, donde se gestionan todas las operaciones relacionadas con ellos.

Atributos

nombre (String):
Nombre de la piscifactoría.

tipo (CriaTipo):
Define el tipo de piscifactoría, que puede ser RIO o MAR.

tanques (ArrayList<Tanque>):
Lista de tanques que componen la piscifactoría. Cada tanque puede contener peces y gestionar sus propias operaciones de crianza.

comidaVegetal (int):
Cantidad de comida vegetal actualmente disponible en la piscifactoría.

comidaAnimal (int):
Cantidad de comida animal actualmente disponible en la piscifactoría.

maxComidaAnimal (int):
Capacidad máxima de almacenamiento de comida animal en la piscifactoría.

maxComidaVegetal (int):
Capacidad máxima de almacenamiento de comida vegetal en la piscifactoría.

monedero (Monedero):
Objeto Monedero, implementado como singleton, que se encarga de gestionar las monedas generadas por la piscifactoría.

Constructor

Piscifactoria(String nombre, boolean primera)
Crea la primera piscifactoría de tipo RIO, con alimentos iniciales y estableciendo el máximo de las capacidades.

Piscifactoria(String nombre, CriaTipo tipo)
Crea una piscifactoría de tipo RIO o MAR sin cantidades iniciales de comida y estableciendo el máximo de las capacidades.

Métodos

getNombre:
public String getNombre()
Devuelve el nombre de la piscifactoría.

getTipo:
public CriaTipo getTipo()
Devuelve el tipo de piscifactoría (RÍO o MAR).

getTanques:
public ArrayList<Tanque> getTanques()
Retorna la lista de tanques en la piscifactoría.

getComidaVegetal:
public int getComidaVegetal()
Devuelve la cantidad de comida vegetal disponible.

getComidaAnimal:
public int getComidaAnimal()
Devuelve la cantidad de comida animal disponible.

getMaxComidaAnimal:
public int getMaxComidaAnimal()
Devuelve la capacidad máxima de comida animal.

getMaxComidaVegetal:
public int getMaxComidaVegetal()
Devuelve la capacidad máxima de comida vegetal.

getMonedero:
public Monedero getMonedero():
Retorna el monedero asociado a la piscifactoría.

showStatus:
public void showStatus():
Muestra el estado general de la piscifactoría, incluyendo el número total de tanques, la ocupación de peces, el porcentaje de peces alimentados, adultos y fértiles, y los niveles de comida vegetal y animal.

selectTank:
public int selectTank():
Permite al usuario seleccionar un tanque y devuelve el índice del tanque elegido.

showTankStatus:
public void showTankStatus():

Muestra el estado detallado de cada tanque, incluyendo la cantidad de peces y su estado de salud.

showFishStatus:
public void showFishStatus():
Muestra el estado de los peces dentro de cada tanque.

showCapacity:
public void showCapacity()
Muestra la capacidad del tanque seleccionado por el usuario, proporcionando detalles de la ocupación actual y la capacidad máxima.

addFood:
public void addFood()
Añade una cantidad específica de comida de tipo Animal o Vegetal a la piscifactoría.

restFood:
public void restFood()
Reduce la cantidad de comida de tipo Animal o Vegetal en la piscifactoría.

comidaAnimalLlena:
public boolean comidaAnimalLlena()
Verifica si la capacidad de almacenamiento de comida animal está llena. Retorna true si está en capacidad máxima, de lo contrario, false.

comidaVegetalLlena:
public boolean comidaAnimalLlena()
Verifica si la capacidad de almacenamiento de comida vegetal está llena. Retorna true si está en capacidad máxima, de lo contrario, false.

comidaAnimalVacia:
public boolean comidaAnimalVacia()
Verifica si la comida animal se ha agotado. Retorna true si la cantidad de comida es cero, de lo contrario, false.

comidaVegetalVacia:
public boolean comidaVegetalVacia()
Verifica si la comida vegetal se ha agotado. Retorna true si la cantidad de comida es cero, de lo contrario, false.

nextDay:
public int[] nextDay()
Simula el paso de un día en la piscifactoría, actualizando los tanques y el estado de los peces. Determina el tipo de alimento necesario y ajusta la cantidad disponible.

sellFish:
public void sellFish()

Vende los peces adultos y vivos de la piscifactoría, actualizando el monedero con las monedas obtenidas.

pecesEnPiscifactoria:
public int pecesEnPiscifactoria()
Calcula y retorna el número total de peces en la piscifactoría.

pecesMaxPiscifactoria:
public int pecesMaxPiscifactoria()
Calcula el número máximo de peces que pueden habitar en la piscifactoría según la capacidad de los tanques.

pecesVivosPiscifactoria:
public int pecesVivosPiscifactoria()
Calcula el número de peces vivos en la piscifactoría.

pecesAlimentadosPiscifactoria:
public int pecesAlimentadosPiscifactoria()
Calcula el número de peces alimentados en la piscifactoría.

pecesAdultosPiscifactoria:
public int pecesAdultosPiscifactoria()
Calcula el número de peces adultos en la piscifactoría.

pecesMachoPiscifactoria:
public int pecesMachoPiscifactoria()
Calcula el número de peces machos en la piscifactoría.

pecesHembraPiscifactoria:
public int pecesHembraPiscifactoria()
Calcula el número de peces hembras en la piscifactoría.

pecesFertilesPiscifactoria:
public int pecesFertilesPiscifactoria()
Calcula el número de peces fértiles en la piscifactoría.

upgradeFood:
public void upgradeFood()
Mejora la capacidad de almacenamiento de comida en la piscifactoría, aumentando el límite máximo según el tipo (RÍO o MAR).

venta:
public int[] venta()
Vende los peces óptimos en cada tanque y devuelve un array con la cantidad de monedas obtenidas y el número total de peces vendidos.

Tanque:

Tanque:

Clase que representa a los tanques que albergan a los peces de una piscifactoría.

Atributos

peces(Arraylist<Pez>):
Nombre con el que se inicia el sistema.

maxPeces(int):
Número máximo de peces que puede haber en el tanque.

tipoPez(PecesDatos):
Tipo de pez existente en el tanque.

monedero(Monedero):
Objeto Monedero, objeto implementado como singleton, que se encarga de gestionar las monedas generadas por la piscifactoría.

tipoT(CriaTipo):
Tipo de tanque, RIO o MAR.

almacenCentral(AlmacenCentral):
Objeto AlmacenCentral, también singleton, que representa el almacén compartido de comida y se utiliza para el manejo de inventario de alimentos en la piscifactoría.

Constructor

Tanque(int maxPeces, CriaTipo tipoT)

Constructor que crea el tanque y asigna el máximo de peces y el tipo del tanque.

Métodos

showStatus:
public void showStatus(int numTanque)

Parámetros:
- numTanque: Número del tanque.

Muestra las estadísticas del tanque.

showFishStatus:

public void showFishStatus()
Muestra las estadísticas de todos los peces del tanque.

showCapacity:
public void showCapacity(int numTanque)
Muestra información de la capacidad del tanque.

Parámetros:
- numTanque: Número del tanque.

nextDay:
public void nextDay(Piscifactoria pisci)

Parámetros:
- Piscifactoria pisci: Objeto Piscifactoría.

Avanza un día, alimenta los peces del tanque, reproduce los peces y vende los peces que estén en su estado óptimo.

pecesEnTanque():
public int pecesEnTanque()
Devuelve la cantidad de peces que hay en el tanque.

pecesVivos:
public int pecesVivos()
Devuelve la cantidad de peces vivos en el tanque.

pecesAlimentados:
int pecesAlimentados()
Devuelve la cantidad de peces alimentados en el tanque.

pecesAdultos:
public int pecesAdultos()
Devuelve la cantidad de peces adultos en el tanque.

pecesHembra:
public int pecesHembra()
Devuelve la cantidad de peces hembra en el tanque.

pecesMacho:
public int pecesMacho()
Devuelve la cantidad de peces macho en el tanque.

pecesFertiles:
public int pecesFertiles()
Devuelve la cantidad de peces fértiles en el tanque.

ventaPecesOptimos:
public int[] ventaPecesOptimos()
Vende todos los peces óptimos del tanque.

showCompatible:
public Pez showCompatible()
Permite elegir entre los peces compatibles con el tanque.

getNumTanque:
public int getNumTanque()
Devuelve el número del tanque.

getPeces:
public ArrayList<Pez> getPeces()
Devuelve la lista de peces en el tanque.

getMaxPeces:
public int getMaxPeces()
Devuelve la capacidad máxima de peces permitidos en el tanque.

getTipoPez:
public PecesDatos getTipoPez()
Devuelve el tipo de pez del tanque.

Peces:

Pez:

Clase abstracta padre de los peces, que define gran parte de atributos y métodos.

Atributos:

Nombre (String):
Nombre común del pez.

Cientifico (String):
Nombre científico del pez.

Ciclo (int):
Numero de días tras los que puede volver a reproducirse una vez ya reproducido.

Coste (int):
Precio de compra del pez.

Huevos (int):
Cantidad de huevos que pone.

Madurez (int):
Tiempo que tarda en llegar a edad adulta.

Monedas (int):
Cantidad de monedas que da al venderlo.

Optimo (int):
Número de días que tarda en venderse.

Piscifactoria (CriaTipo):
Tipo de piscifactoría en la que puede criarse.

Tipo (PecesTipo):
Tipo de pez.

Edad (int):
Edad del pez.

Sexo (boolean):
Sexo del pez.

Comido (boolean):
Si el pez ha comido.

Fertil (boolean):
Si el pez puede reproducirse.

Vivo (boolean):
Si el pez está vivo.

UltimaPuesta (int):
Último ciclo de puesta.

Constructor

Pide los datos de un pez e introduce estos datos en los atributos, pero no se construye porque Pez es una clase abstracta.

Métodos:

getName:
public String getName()
Devuelve el nombre común del pez.

getCientifico:
public String getCientifico()
Devuelve el nombre científico del pez.

getAge:
public int getAge()
Devuelve la edad del pez.

isFemale:
public boolean isFemale()
Devuelve si el pez es hembra.

isAlive:
public boolean isAlive()
Devuelve si el pez está vivo.

isFertile:
public boolean isFertile()
Devuelve si el pez es fértil.

isAdulto:
public boolean isAdulto()
Devuelve si el pez es adulto.

getHuevos:

public int getHuevos()
Devuelve la cantidad de huevos que pone el pez

getCoste:
public int getCoste()
Devuelve el coste de compra del pez.

getMonedas:
public int getMonedas()
Devuelve la cantidad de monedas que da el pez al ser vendido.

getPiscifactoria:
public CriaTipo getPiscifactoria()
Devuelve el tipo de piscifactoría a la que pertenece.

isAlimentado:
public boolean isAlimentado()
Devuelve si el pez ha comido.

getOptimo:
public int getOptimo()
Devuelve la edad óptima para vender el pez.

notFertil:
public void notFertil()
Hace que el pez deje de ser fértil.

resetPuesta:
public void resetPuesta()
Pone el tiempo de última puesta en 0.

showStatus:
public void showStatus()
Muestra el estado del pez.

grow():
public int grow()
Hace crecer el pez en un día, comprobando todos los factores y haciéndolo comer.
Pide la cantidad de comida disponible y devuelve la cantidad de comida consumida.

reproducirse:
public Pez reproducirse()
Pide una comprobación de si la cría es macho o hembra, y devuelve una cría del mismo pez con el sexo que se le pase.

comer:
public int comer()

Hace comer al pez si hay comida suficiente, comprobando la cantidad que se le pasa por parámetro, y devuelve la cantidad de comida que el pez ha consumido.

reset:
public void reset()
Resetea el pez, manteniendo todos los atributos pero poniendo los de estado a 0.

Propiedades:

Carnívoro:

Clase que representa a un pez carnívoro, hereda de Pez.

Filtrador:

Clase que representa a un pez filtrador, hereda de Pez.

Omnívoro:

Clase que representa a un pez omnívoro, hereda de Pez.

Double:

Dorada:

Clase que representa al pez Dorada, hereda de Omnivoro.

TruchaArcoiris:

Clase que representa al pez Trucha Arcoíris, hereda de Carnívoro.

Mar:

Abadejo:

Clase que representa al pez Abadejo, hereda de Carnívoro.

Besugo:

Clase que representa al pez Besugo, hereda de Carnívoro.

Caballa:

Clase que representa al pez Caballa, hereda de Carnívoro.

Rodaballo:

Clase que representa al pez Rodaballo, hereda de Carnívoro.

Sargo:

Clase que representa al pez Sargo, hereda de Omnívoro.

Rio:

Carpa:

Clase que representa al pez Carpa, hereda de Omnívoro.

CarpaPlateada:

Clase que representa al pez Carpa Plateada, hereda de Filtrado.

LucioDelNorte:

Clase que representa al pez Lucio del Norte, hereda de Carnívoro.

Pejerrey:

Clase que representa al pez Pejerrey, hereda de Carnívoro.

TilapiaDelNilo:

Clase que representa al pez Tilapia del Nilo, hereda de Filtrador.

Común :

AlmacenCentral:

Clase singleton que representa a un almacén central para el manejo de comida animal y vegetal en una piscifactoría. Se encarga de almacenar, distribuir y gestionar la capacidad de ambos tipos de alimentos entre múltiples instancias de piscifactorías.

Atributos

static AlmacenCentral getInstance():
Método estático que proporciona la instancia única de AlmacenCentral. Si la instancia no existe, la crea.

comidaAnimal (int)
Comida animal disponible en el almacén

comidaVegetal (int)
Comida vegetal disponible en el almacén

capacidadComidaAnimal (int)
Capacidad máxima de la comida animal

capacidadComidaVegetal (int)
Capacidad máxima de la comida vegetal

Constructor

private AlmacenCentral():
Constructor privado para implementar el patrón Singleton y asegurar que solo exista una instancia de AlmacenCentral.

Métodos

getInstance:
public static AlmacenCentral getInstance()
Método estático que proporciona la instancia única de AlmacenCentral. Si la instancia no existe, la crea.

getComidaAnimal:
public int getComidaAnimal()
Devuelve la cantidad actual de comida animal disponible en el almacén.

getComidaVegetal:
public int getComidaVegetal()
Devuelve la cantidad actual de comida vegetal disponible en el almacén.

getCapacidadComidaAnimal:
public int getCapacidadComidaAnimal()
Devuelve la capacidad máxima de almacenamiento de comida animal.

getCapacidadComidaVegetal:
public int getCapacidadComidaVegetal()
Devuelve la capacidad máxima de almacenamiento de comida vegetal.

cogerComidaAnimal:
public int cogerComidaAnimal(int cantidad)

Parámetros:
- Cantidad: Cantidad de comida animal a coger.

Reduce la cantidad de comida animal en el almacén en la cantidad especificada, siempre que haya suficiente comida disponible. Devuelve la cantidad retirada o 0 si no hay suficiente.

cogerComidaVegetal:
public int cogerComidaVegetal(int cantidad)

Parámetros:
- Cantidad: Cantidad de comida vegetal a coger.

Reduce la cantidad de comida vegetal en el almacén en la cantidad especificada, siempre que haya suficiente comida disponible. Devuelve la cantidad retirada o 0 si no hay suficiente.

addFood:
public void addFood(int cantidad, String tipo)

Parámetros:
- cantidad: Cantidad de comida a añadir.
- tipo: Tipo de comida a añadir.

Añade una cantidad específica de alimento de tipo "Animal" o "Vegetal" al almacén.

repartir:
public void repartir(ArrayList<Piscifactoria> piscifactorias)

Parámetros:
- piscifactorías: Arraylist de piscifactorías.

Distribuye proporcionalmente la comida animal y vegetal disponible en el almacén entre una lista de piscifactorías. La distribución se realiza solo si las piscifactorías no están llenas y si existe suficiente comida en el almacén para repartir.

aumentarCapacidad:
public void aumentarCapacidad(int capacidad)

Parámetros:
- Cantidad: Cantidad a aumentar.

Aumenta la capacidad máxima de almacenamiento de comida animal y vegetal en el almacén en la cantidad especificada, permitiendo una mayor capacidad de almacenamiento para ambos tipos de alimento.

Monedero:

Clase singleton que representa el monedero del sistema. Se encarga de almacenar las monedas obtenidas gracias a ir avanzando en el uso del sistema.

Atributos

instance(Monedero):
Instancia del objeto Monedero.

monedas(int):
Cantidad de monedas.

Constructor

Monedero()
Constructor vacío de la clase Monedero.

Métodos

getInstance:
public static Monedero getInstance()
Permite crear el objeto en caso de que no exista, y en caso de que exista solo devuelve la instancia.

getMonedas:
public int getMonedas()
Obtiene la cantidad de monedas.

setMonedas:
public void setMonedas(int monedas)

Permite cambiar la cantidad de monedas.

Recompensas:

Recompensas

La clase Recompensas permite gestionar recompensas del sistema almacenándolas en archivos XML. Proporciona métodos para crear recompensas, modificarlas, listarlas y reclamarlas, permitiendo su distribución según el nivel o tipo de recompensa.

Atributos

ruta (String):
Ruta donde se almacenan los archivos XML de recompensas. Por defecto, es "rewards".

doc (Document):
Objeto Document de la biblioteca DOM4J que representa el documento XML en uso durante las operaciones.

Métodos

hacerCarpeta:
public static void hacerCarpeta()
Crea la carpeta base donde se almacenarán los archivos XML de recompensas si no existe.

algaXml
public static void algaXml(int nivel)
Genera un archivo XML con información sobre recompensas de tipo "alg**a".**
Parámetros:

- nivel (int): Nivel de la recompensa (1 a 5), que define la cantidad de algas.

almacenXml
public static void almacenXml(int nivel)
Genera un archivo XML con información sobre recompensas relacionadas con la construcción de un almacén central.
Parámetros:

- nivel (int): Nivel de la recompensa, que corresponde a las partes (A, B, C, D) de un almacén central.

comidaXml
public static void comidaXml(int nivel)
Genera un archivo XML con información sobre recompensas de comida general.
Parámetros:

- nivel (int): Nivel de la recompensa (1 a 5), que define los valores específicos.

monedasXml
public static void monedasXml(int nivel)
Genera un archivo XML con información sobre recompensas de monedas.
Parámetros:

- nivel (int): Nivel de la recompensa (1 a 5), que define los valores específicos.

piensoXml
public static void piensoXml(int nivel)
Genera un archivo XML con información sobre recompensas de pienso animal.
Parámetros:

- nivel (int): Nivel de la recompensa (1 a 5), que define los valores específicos.

pisciMarXml
public static void pisciMarXml(int parte)
Genera un archivo XML con información sobre recompensas para construir piscifactorías de mar.
Parámetros:

- parte (int): Parte de la recompensa (A o B).

pisciRioXml
public static void pisciRioXml(int parte)
Genera un archivo XML con información sobre recompensas para construir piscifactorías de río.
Parámetros:

- parte (int): Parte de la recompensa (A o B).

tanqueXml
public static void tanqueXml(int tipo)
Genera un archivo XML con información sobre recompensas para construir tanques.
Parámetros:

- tipo (int):Tipo del tanque: 1 para río, 2 para mar.

save
public static void save(String nombreArchivo)
Guarda el archivo XML actual en la ruta especificada.
Parámetros:

- nombreArchivo (String): Nombre del archivo donde se guardará el documento.

addQuantity
public static void addQuantity(String nombreArchivo)
Incrementa la cantidad en el archivo XML especificado en 1.
Parámetros:

- nombreArchivo (String): Nombre del archivo a modificar.

restQuantity

public static void restQuantity(String nombreArchivo)
Reduce la cantidad en el archivo XML especificado en 1. Si llega a 0 el archivo se elimina.
Parámetros:

- nombreArchivo (String): Nombre del archivo a modificar.

listRecompensas
public static void listRecompensas()
Lista todas las recompensas disponibles leyendo los archivos en la carpeta de recompensas.

reclamar
public static void reclamar(Registros registros, File file, ArrayList<Piscifactoria> piscifactorias)
Reclama una recompensa, aplicándola a un conjunto de piscifactorías y realiza las acciones necesarias según el tipo de recompensa.
Parámetros:

- registros (Registros): Objeto Registros para gestionar operaciones relacionadas
- file (File): Archivo XML de la recompensa a reclamar.
- piscifactorías (ArrayList<Piscifactoria>): Lista de piscifactorías para la gestión de las recompensas.

processFoodReward
public static void processFoodReward(Element foodElement, ArrayList<Piscifactoria> piscifactorias)
Procesa las recompensas de tipo "food" del archivo XML y las distribuye entre las piscifactorías según el tipo de comida.
Parámetros:

- foodElement (Element): Elemento XML que representa una recompensa de comida
- piscifactorías (ArrayList<Piscifactoria>): Lista de piscifactorías donde se distribuirá la recompensa.

processBuildingsReward
public static void processBuildingsReward(Element buildingElement, ArrayList<Piscifactoria> piscifactorias)
Procesa las recompensas de tipo "buildings" del archivo XML y realiza las acciones correspondientes, como añadir nuevas piscifactorías o tanques.

Parámetros:

- foodElement (Element): Elemento XML que representa una recompensa de construcción.
- piscifactorías (ArrayList<Piscifactoria>):Lista de piscifactorías donde se aplicarán las recompensas.

processCoinsReward

public static void processCoinsReward(Element coinsElement)

Procesa las recompensas de tipo "coins" del archivo XML y añade la cantidad correspondiente al monedero del jugador.

Parámetros:

- coinsElement (Element): Elemento XML que representa una recompensa de monedas.

Registros:

Registros

La clase Registros es una clase intermedia que se utiliza para llamará a los métodos de las clases Logs y Transcripciones.

Atributos

transcripciones (Transcripciones):
Objeto de la clase Transcripciones.

logs (Logs):
Objeto de la clase Logs.

Constructor

Registros(String nombrePartida)
Constructor que inicializa las instancias de las clases logs y transcripciones.

Métodos

inicio:
public void inicio(String primeraPisc,String[] extras,String[] pecesImplementasdos,int monedas,String nombrePartida)

Metodo que llama a los métodos de inicio de la clase transcripciones y logs.

comprarComida:
public void comprarComida(int cantComida, String tipoCom, int monedas, String lugarGuardado ,Piscifactoria pisc)

Metodo que llama a los metodos comprarComida de la clase transcripciones y logs.

comprarPeces:
public void comprarPeces(Pez pez, char sexo, int tanque, Piscifactoria pisc, int monedas)

Método que llama a los metodos comprarPeces de la clase transcripciones y logs.

venderPeces:
public void venderPeces(int cantPeces, int monedas, Piscifactoria pisc)

Metodo que llama a los métodos venderPeces de la clase transcripciones y logs.

limpiarTanque:
public void limpiarTanque(int tanque, Piscifactoria pisc)

Metodo que llama a los metodos limpiarTanque de la clase transcripciones y logs.

vaciarTanque:
public void vaciarTanque(int tanque, Piscifactoria pisc)

Metodo que llama a los metodos vaciarTanque de la clase transcripciones y logs.

comprarEdificio:
public void comprarEdificio(Piscifactoria pisc, int monedas)

Metodo que llama a los metodos comprarEdificio de la clase transcripciones y logs.

mejorarEdificio:
public void mejorarEdificio(Piscifactoria pisc, int monedas, int tanque, AlmacenCentral almacenCentral)

Metodo que llama a los metodos mejorarEdificio de la clase transcripciones y logs.

pasarDia:
public void pasarDia(int numDia, int pecesRio, int pecesMar, int monedasObtenidas, int monedasTotales)

Metodo que llama a los metodos pasarDia de la clase transcripciones y logs.

ocultas:
public void ocultas(Piscifactoria pisc, int monedasAñad, int monedas)

Metodo que llama a los metodos ocultas de la clase transcripciones y logs.

recompensaCreada:
public void recompensaCreada(String nombreRec)

Método que llama a los metodos recompensaCreada de la clase transcripciones y logs.

recompensaUsada:
public void recompensaUsada(String nombreRec)

Método que llama a los metodos recompensaUsada de la clase transcripciones y logs.

registrarGuardado:
public void registrarGuardado()

Método que llama al metodos registrarGuardado de la clase logs.

registrarCarga:
public void registrarCarga()

Método que llama al metodo registrarCarga de la clase logs.

salir:
public void salir()

Metodo que llama al metodo salir de logs y al método close de logs y transcripciones.

Transcripciones:

Transcripciones

Clase singleton encargada de hacer las transcripciones del sistema.

Atributos

instance(Transcripciones):
Instancia del objeto de Transacciones.

ruta(String):
Nombre de la carpeta de guardado.

bw(BufferedWriter):
Objeto BuffererWriter para poder escribir en el archivo.

Constructor

Transcripciones()
Constructor vacío de la clase Transacciones.

Métodos

getInstance:
public static Transcripciones getInstance(String nombrePartida)

Método que permite crear el objeto en caso de que no exista, y en caso de que exista solo devuelve la instancia.

escribirArchivo:
private void escribirArchivo(String texto)

Metodo que escribe un linea de texto pasada por parametro en el archivo.

inicio:
public void inicio(String primeraPisc,String[] extras,String[] pecesImplementasdos,int monedas,String nombrePartida)

Metodo que escribe en el archivo todos los parametros del inicio del programa.

comprarComida:
public void comprarComida(int cantComida, String tipoCom, int monedas, String lugarGuardado ,Piscifactoria pisc)

Metodo que escribe en el archivo la infromacion de las compras de comida.

comprarPeces:
public void comprarPeces(Pez pez, char sexo, int tanque, Piscifactoria pisc, int monedas)

Metodo que escribe en el archivo la informacion de las compras de los peces.

venderPeces:
public void venderPeces(int cantPeces, int monedas, Piscifactoria pisc)

Metodo que escribe en el archivo la informacion de las ventas manuales de los peces.

limpiarTanque:
public void limpiarTanque(int tanque, Piscifactoria pisc)

Metodo que escribe en el archivo la informacion de la limpieza de un tanque.

vaciarTanque:
public void vaciarTanque(int tanque, Piscifactoria pisc)

Metodo que escribe en el archivo la informacion del vaciado de los peces del tanque.

comprarEdificio:
public void comprarEdificio(Piscifactoria pisc, int monedas)

Metodo que escribe en el archivo la informacion de la compra de algun edificio.

mejorarEdificio:
public void mejorarEdificio(Piscifactoria pisc, int monedas, int tanque, AlmacenCentral almacenCentral)

Metodo que escribe en el archivo la informacion de las mejoras de los edificios.

pasarDia:
public void pasarDia(int numDia, int pecesRio, int pecesMar, int monedasObtenidas, int monedasTotales)

Metodo que escribe en el archivo la informacion de los peces totales al final del dia y las monedas obtenidas con la venta automatica y las monedas totales.

ocultas:
public void ocultas(Piscifactoria pisc, int monedasAñad, int monedas)

Metodo que escribe en el archivo la informacion del uso de las opciones ocultas del programa.

recompensaCreada:
public void recompensaCreada(String nombreRec)
Método que escribe en el archivo la informacion de la creacion de recompensas del programa.

recompensaUsada:
public void recompensaUsada(String nombreRec)

Método que escribe en el archivo la informacion del uso de recompensas del programa.

close:
public void close()

Método que cierra el buffererWriter.

Logs:

Logs

Clase singleton encargada de hacer los logs del sistema.

Atributos

instance(Logs):
Instancia del objeto de Transacciones.

ruta(String):
Nombre de la carpeta de guardado.

bw(BufferedWriter):
Objeto BuffererWriter para poder escribir en el archivo.

Constructor

Logs()
Constructor vacío de la clase Logs.

Métodos

getInstance:
public static Transcripciones getInstance(String nombrePartida)

Método que permite crear el objeto en caso de que no exista, y en caso de que exista solo devuelve la instancia.

escribirArchivo:
private void escribirArchivo(String texto)

Metodo que escribe un linea de texto pasada por parametro en el archivo.

inicio:
public void inicio(String primeraPisc,String[] extras,String[] pecesImplementasdos,int monedas,String nombrePartida)

Método que registra en el archivo la hora del inicio del programa.

comprarComida:
public void comprarComida(int cantComida, String tipoCom, int monedas, String lugarGuardado ,Piscifactoria pisc)

Método que registra en el archivo la hora de las compras de comida.

comprarPeces:
public void comprarPeces(Pez pez, char sexo, int tanque, Piscifactoria pisc, int monedas)

Método que registra en el archivo la hora de las compras de los peces.

venderPeces:
public void venderPeces(int cantPeces, int monedas, Piscifactoria pisc)

Método que registra en el archivo la hora de las ventas manuales de los peces.

limpiarTanque:
public void limpiarTanque(int tanque, Piscifactoria pisc)

Método que registra en el archivo la hora de la limpieza de un tanque.

vaciarTanque:
public void vaciarTanque(int tanque, Piscifactoria pisc)

Método que registra en el archivo la hora del vaciado de los peces del tanque.

comprarEdificio:
public void comprarEdificio(Piscifactoria pisc, int monedas)

Método que registra en el archivo la hora de la compra de algun edificio.

mejorarEdificio:
public void mejorarEdificio(Piscifactoria pisc, int monedas, int tanque, AlmacenCentral almacenCentral)

Método que registra en el archivo la hora de las mejoras de los edificios.

pasarDia:
public void pasarDia(int numDia)

Método que registra en el archivo la hora de en la que se paso un día.

ocultas:
public void ocultas(Piscifactoria pisc, int monedasAñad, int monedas)

Método que registra en el archivo la hora del uso de las opciones ocultas del programa.

recompensaCreada:
public void recompensaCreada(String nombreRec)

Método que registra en el archivo la hora de la creacion de recompensas del programa.

recompensaUsada:
public void recompensaUsada(String nombreRec)

Método que registra en el archivo la hora del uso de recompensas del programa.

registrarGuardado:
public void registrarGuardado()

Método que registra en el archivo la hora de guardado del programa.

registrarCarga:
public void registrarCarga()

Método que registra en el archivo la hora de cargado del programa.

salir:
public void salir()

Método que registra en el archivo la hora de finalización de la partida

close:
public void close()

Método que cierra el buffererWriter.

fechaActual:
private String fechaActual()

Método que obtiene la fecha actual.

Saves:

DTOAlmacen

Clase que guarda la información del almacén central.

Atributos

disponible(boolean)
Disponibilidad del almacén.

capacidad(int)
Capacidad de comida del almacén.

comida(Map<String, Integer)
Cantidad de comida actual del almacén.

Constructor

DTOAlmacen(AlmacenCentral alm)
Constructor que recibe el almacen central y guarda los datos.

Métodos

getCapacidad
public int getCapacidad()
Getter de capacidad del almacén.

getComida
public Map<String, Integer> getComida()
Getter de la cantidad de comida actual del almacén.

isDisponible
public boolean isDisponible()
Getter de la disponibilidad del almacén central.

DTOPez

Clase que guarda la información de los peces

Atributos

edad(int)
Edad del pez en días.

sexo(boolean)
Sexo del pez.

vivo(boolean)
Estado del pez.

maduro(boolean)
Madurez del pez.

fertil(boolean)
Fertilidad del pez.

ciclo(int)
Dias de ciclo del pez.

alimentado(boolean)
Alimentación del pez.

Constructor

DTOPez(Pez pez)
Constructor que recibe un pez y guarda sus datos.

Métodos

isAlimentado
public boolean isAlimentado()
Getter del estado de alimentación del pez.

isFertil
public boolean isFertil()
Getter del estado de fertilidad del pez.

isMaduro
public boolean isMaduro()
Getter del estado de madurez del pez.

isSexo
public boolean isSexo()
Getter del sexo del pez.

isVivo
public boolean isVivo()
Getter del estado de vitalidad del pez.

getCiclo
public int getCiclo()
Getter de días de ciclo de reproducción del pez.

getEdad
public int getEdad()
Getter de la edad del pez

DTOPiscifactoria

Clase que guarda la información de las piscifactorías.

Atributos

nombre(String)
Nombre de la piscifactoría.

tipo(int)
Tipo de piscifactoría

capacidad(int)
Capacidad de los peces de la piscifactoría

comida(Map<String, Integer>)
Cantidades de comida en clave-valor

tanques(List<DTOTanque>)
Lista de tanques de la piscifactoría

Constructor

DTOPiscifactoría(Piscifactoria pisci)
Constructor que recibe una piscifactoría y guarda los datos.

Metodos

getCapacidad
public int getCapacidad()
Getter de capacidad de los peces.

getComida
public Map<String, Integer> getComida()
Getter de la comida de la piscifactoría.

getNombre
public String getNombre()
Getter del nombre de la piscifactoría.

getTanques
public List<DTOTanque> getTanques()
Getter de los tanques de la piscifactoría

getTipo
public int getTipo()
Getter del tipo de piscifactoría

DTOSimulador

Clase que guarda la información del simulador

Atributos

implementados(List<String>)
Lista de peces implementados.

nombres(String[])
Lista de nombres en array, usado para las estadisticas

stats(Estadisticas)
Instancia de las estadísticas

empresa(String)
Nombre de la empresa

dia(int)
Dia del simulador

monedas(int)
Monedas disponibles

orca(String)
Estadísticas de venta, nacimientos…

edificios(Map<String, Object>)
Edificios disponibles

piscifactorias(List<DTOPiscifactoria)
Piscifactorías del simulador

Constructor

DTOSimulador(Simulador sim)
Constructor que recibe el simulador y guarda los datos

Metodos

getDia
public int getDia()
Getter del día.

getEdificios
public Map<String, Object> getEdificios()
Getter de los edificios.

getMonedas
public int getMonedas()
Getter de las monedas.

getEmpresa
public String getEmpresa()
Getter del nombre de la empresa.

getPiscifactorias
public List<DTOPiscifactoría> getPiscifactorias()
Getter de las piscifactorías

getOrca
public String getOrca()
Getter de las estadisticas de los peces

getImplementados
public List<String> getImplementados()
Getter de los peces implementados.

DTOTanque

Clase que guarda la información a guardar de los tanques.

Atributos

pez(String)
Nombre del pez que hay en el tanque

num(int)
Número de peces en el tanque.

datos(Map<String, Integer>)
Datos de los peces del tanque.

peces(List<DTOPez)
Lista de peces en el tanque.

Constructor

DTOTanque(Tanque tanque)
Constructor que recibe un tanque y guarda los datos.

Métodos

getPez
public String getPez()
Getter del nombre del pez que hay en el tanque.

getNum
public int getNum()
Getter del numero de peces en el tanque

getDatos
public Map<String, Integer> getDatos()
Getter de los datos.

getPeces
public List<DTOPez> getPeces()
Getter de la lista de peces en el tanque.

SaveLoad

Clase que gestiona la lógica de guardar y cargar partida en formato JSON

Atributos

saveDir(File)
Directorio de los archivos de guardado

Métodos

save
public void save(DTOSimulador sim, File archivo, Registros reg)
Guarda la partida, dejando constancia de ello en el registro.

saveDirCreate
public static void saveDirCreate()
Crea la carpeta de guardado previa comprobación de su existencia.

cargar
public void cargar(File json)
Carga la partida a partir de un fichero JSON

Helpers:

MenuHelper:

Clase que se encarga de mostrar menús por pantallal

Métodos

mostrarMenu:
public static void mostrarMenu(String[] opciones, boolean salir)

Parámetros:
- opciones: Array de string con las opciones del menú.
- salir: Muestra la opción de salir si el booleano es true.

Crea un menú numerado con los parámetros introducidos en el array de Strings, con una opción salir o no, que muestra salir en el número 0.

MonederoHelper:

Clase que se encarga de proporcionar métodos para ayudar al correcto uso de las monedas en el sistema.

Métodos

monedasSuficientes:
public static boolean monedasSuficientes(int cant)

Parámetros:
- cant: Cantidad de monedas a comprobar

Verifica si hay monedas suficientes en el monedero

calcularDescuento:
public static int calcularDescuento(int precio)

Parámetros:
- Precio: Precio base para calcular el descuento.

Calcula el descuento aplicable para la compra de comida.

InputHelper:

Clase que se encarga de controlar la entrada de datos.

Atributos

BufferedReader br:
BufferedReader para leer datos pasados por pantalla

Métodos

GetIntWithBuffRead:
public static int GetIntWithBuffRead()
Devuelve un entero después de comprobar que lo pasado por pantalla es un número entero.

GetFloatWithBuffRead:
public static Float GetFloatWithBuffRead()
Devuelve un número de coma flotante después de comprobar que lo pasado por pantalla es correcto.

ReadStringWithBuffRead:
public static String ReadStringWithBuffRead()
Devuelve un String pasado por pantalla.

CloseBuffReader:
public static void CloseBuffReader()
Cierra el flujo de datos