MANUAL TÉCNICO DE PISCIFACTORÍA

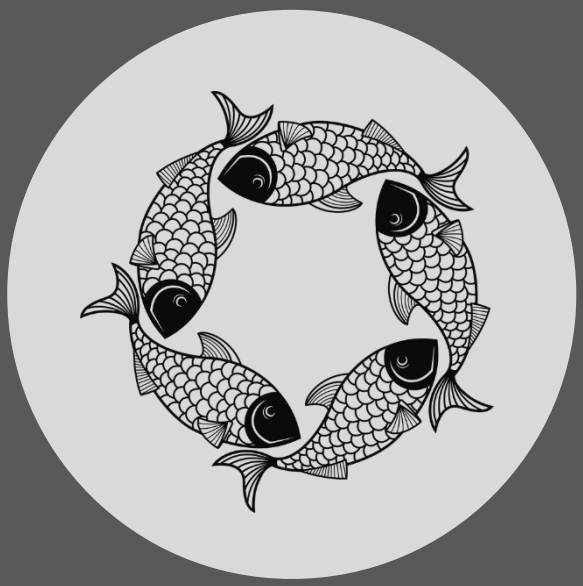
2ºDAM

IES ARMANDO COTARELO VALLEDOR

ACCESO

A

DATOS



Manuel Abalo Rietz

Adrián Ces López

Pablo Dopazo Suárez

GRUPO FANTASY

Contenido

Introducción

Este proyecto consiste en la creación de un simulador de piscifactoría en Java, abarcando todos los aspectos de la gestión y el ciclo de vida de los peces en entornos controlados de agua dulce y salada. El sistema está diseñado para emular procesos como la alimentación, reproducción, crecimiento y venta de peces, así como el manejo de recursos necesarios para la operación de la piscifactoría, como alimentos y espacios de almacenamiento. Además, integra un sistema de monedas que permite simular las transacciones económicas asociadas con la compra de insumos y la venta de peces. A medida que el proyecto evoluciona, se podrán implementar mejoras y correcciones que optimicen la funcionalidad general del simulador.

Package bd

Clase Conexión

Descripcion

La clase Conexion gestiona la conexión con la base de datos MySQL para la aplicación de piscifactoría.
Implementa el patrón Singleton, asegurando que solo haya una única conexión activa a la base de datos a la vez.
Permite establecer y cerrar la conexión de manera centralizada, evitando conexiones innecesarias.

Atributos

private static Properties connectionProps
Almacena las credenciales y configuraciones de la conexión.

private static Connection con
Gestionar la conexión con la base de datos.

private static final String USER
Nombre de usuario para la conexión a la base de datos.

private static final String PASSWORD
Contraseña del usuario para la conexión

private static final String SERVER
Dirección del servidor de la base de datos

private static final String PORTNUMBER
Puerto del servidor MySQL (3307).

privte static final String DATABASE
Nombre de la base de datos utilizada.

Contructor

- **private Conexion()**: Constructor privado para evitar que se creen instancias de la clase.

Métodos

public static Connection obtenerConexion()
Establece la conexión con la base de datos si no está creada y la retorna.

public static void cerrarConexion()
Cierra la conexión con la base de datos si está abierta.

Clase DAOPedidos

Descripcion

La clase DAOPedidos gestiona las operaciones de acceso a la base de datos para la tabla pedidos. Permite insertar, actualizar, consultar y eliminar pedidos en la base de datos.

public static void insertarPedido(PedidoDTO pedido)
Inserta un nuevo pedido en la base de datos.

Parámetro:
PedidoDTO pedido: Objeto que contiene los datos del pedido (cliente, pez, cantidad de peces, peces enviados).

public static void actualizarPedido(PedidoDTO pedido)
Actualiza la cantidad de peces enviados en un pedido existente.

Parámetros
pedido (PedidoDTO): Objeto que representa el pedido a actualizar.

public static List<String> listarPedidosPendientes()
Devuelve una lista de pedidos que aún no han sido completados

public static List<String> listarPedidosCompletados()
Devuelve una lista de pedidos completado.

public static void borrarPedidos()
Borra todos los pedidos de la tabla pedidos.

public static PedidoDTO obtenerPedidoPorId(int idPedido)
Obtiene un pedido específico de la base de datos.

Parámetros:
idPedido (int): ID del pedido que se desea obtener.

public static String obtenerNombrePez(int idPez)
Obtiene el nombre de un pez dado su id.

Parámetros:
idPez (int): ID del pez.

public static boolean existeCliente(int idCliente)
Verifica si un cliente con el ID especificado existe en la base de datos.

Parámetros:

idCliente (int): ID del cliente.

public static boolean existePez(int idPez)

Verifica si un pez con el ID especificado existe en la base de datos.

Parámetros:

idPez (int): ID del pez.

Clase GeneradorBD

La clase se encarga de la creación e inicialización de la base de datos del sistema de piscifactoría.

Métodos

public static void crearTablas()

Crea las tablas necesarias en la base de datos si todavía no existen.

public static void insertarPeces()

Inserta en la tabla peces los nombres y nombres científicos de las especies de peces.

public static void insertarClientes()

Inserta datos iniciales de clientes en la tabla clientes.

public static void generarBD()

Llama a los métodos de creación de tablas e inserción de datos iniciales.

Package dto

Clase Cliente dto

Descripcion

La clase ClienteDTO representa un objeto de transferencia de datos (DTO) para un cliente en el sistema de piscifactoría.
Se utiliza para encapsular la información de un cliente y facilitar su manipulación en la base de datos.

Atriubutos

private int id

Identificador único del cliente en la base de datos.

private String nombre

Nombre del cliente.

private String nif

Número de Identificación Fiscal (NIF) del cliente, utilizado para identificación única.

private String telefono

Número de teléfono del cliente.

Constructores

- **public ClienteDTO(int id, String nombre, String nif, String telefono):** Constructor que inicializa un objeto ClienteDTO con todos sus atributos.

Parámetros:

int id: Identificador único del cliente.

String nombre: Nombre del cliente.

String nif: Número de Identificación Fiscal del cliente.

String telefono: Número de teléfono del cliente.

ClasePedidoDTO

La clase PedidoDTO representa un objeto de transferencia de datos (DTO) para gestionar los pedidos de peces en la piscifactoría.
Se utiliza para encapsular la información de un pedido y facilitar su manipulación en la base de datos.

Atributos

private int id
Identificador único del pedido en la base de datos.

private int idCliente
Identificador del cliente que realizó el pedido.

private int idPez
Identificador del pez solicitado en el pedido.

private int cantidadPeces
Cantidad total de peces solicitados en el pedido.

private int pecesEnviados
Cantidad de peces enviados en el pedido.
Puede ser menor o igual a cantidadPeces.

Constructores

- **public PedidoDTO(int id, int idCliente, int idPez, int cantidadPeces, int pecesEnviados):** Constructor que inicializa un objeto PedidoDTO con todos sus atributos.

Parámetros:

int id: Identificador único del pedido.

int idCliente: ID del cliente que realizó el pedido.

int idPez: ID del pez solicitado en el pedido.

int cantidadPeces: Número total de peces solicitados.

int pecesEnviados: Número de peces que ya fueron enviados.

Clase PezDTO

Descripcion

La clase PezDTO representa un objeto de transferencia de datos (DTO) para gestionar los peces en la base de datos de la piscifactoría.
Se utiliza para encapsular la información de un pez y facilitar su manipulación en la capa de datos.

Atributos

private int id
Identificador único del pez en la base de datos.

private String nombre
Nombre común del pez.

private String nombreCientifico
Nombre científico del pez.

Constructores

- **public PezDTO(int id, String nombre, String nombreCientifico):** Constructor que inicializa un objeto PezDTO con todos sus atributos.

Package gestor_guardado

Clase Cargado

Descripción

La clase Cargado se encarga de gestionar la carga de partidas guardadas en el sistema de piscifactoría. Lee archivos de guardado en formato JSON, extrae la información de piscifactorías, tanques y peces, y la asigna a una instancia del simulador.

Métodos

public static void cargarPartida(Simulador simulador, String nombrePartida)
Carga una partida guardada desde un archivo JSON y restaura el estado del simulador.

Parámetros:
Simulador simulador: Instancia del simulador donde se restaurarán los datos.

String nombrePartida: Nombre del archivo de guardado a cargar.

private static Pez crearPez(String nombrePez, boolean sexo)
Crea una nueva instancia de pez basada en su tipo.

Parámetros:
String nombrePez: Nombre del pez a crear.

boolean sexo: Sexo del pez (true para macho, false para hembra).

Clase Guardado

Descripción

La clase Guardado permite almacenar el estado actual del simulador en un archivo JSON. Incluye la información de la empresa, el día actual, el saldo de monedas, las piscifactorías con sus respectivos tanques y peces, así como los almacenes de comida.

Atributos

private Simulador simulador

Referencia al simulador cuya partida se va a guardar.

Constructores

public Guardado(Simulador simulador)

Inicializa la clase Guardado con una instancia del simulador.

Parámetros:

Simulador simulador: Objeto del simulador cuyo estado se desea guardar.

Métodos

public void guardarPartida()

Guarda el estado del simulador en un archivo JSON dentro de la carpeta saves.

Package edificios

Package edificios.almacenes

Clase AlmacenCentral

Descripción

La clase AlmacenCentral gestiona el almacenamiento centralizado de comida y su distribución en una piscifactoría. Contiene dos almacenes específicos para diferentes tipos de comida: animal y vegetal, y cuenta con métodos para añadir comida y distribuirla equitativamente.

Atributos

private int maxCap

Capacidad máxima total del almacén central. Valor inicial de 400 unidades.

private int stock

Cantidad actual de comida almacenada en el almacén central. Inicia en 0.

private int price

Precio de construcción del almacén central, establecido en 2000 monedas.

private AlmacenComida warehouseA

Instancia del almacén específico para comida animal, con capacidad inicial de 200 unidades.

private AlmacenComida warehouseV

Instancia del almacén específico para comida vegetal, también con capacidad inicial de 200 unidades.

Constructores

- **public AlmacenCentral()**: Constructor que inicializa el almacén central con capacidad total de 400 unidades, y crea dos almacenes específicos (warehouseA y warehouseV) con 200 unidades cada uno.

Métodos

public void addVegtalFood(int ammount)

Método que añade comida vegetal al almacén central y actualiza el stock.

Parámetro:

int ammount: Cantidad de comida vegetal a añadir.

public void addAnimalFood(int ammount)

Añade comida animal al almacén central y actualiza el stock.

Parámetro:

int ammount: Cantidad de comida animal a añadir.

public void getOcuped()

Muestra en consola la ocupación actual del almacén central.

public void upgrade()

Aumenta la capacidad máxima del almacén en 50 unidades

public int getMaxCap()

Devuelve la capacidad máxima del almacén central.

public int getStock()

Devuelve la cantidad de comida almacenada en el almacén central.

public AlmacenComida getWarehouseA()

Devuelve la instancia del almacén de comida animal.

public AlmacenComida getWarehouseV()

Devuelve la instancia del almacén de comida vegetal.

@Override public String toString()

Sobrescribe el método toString para representar el estado del almacén central. Devuelve una cadena con la información del almacén central, incluyendo el precio, cantidad de comida animal y vegetal, stock total y capacidad máxima.

Clase AlmacenComida

Descripción

La clase AlmacenComida representa un almacén de comida en la piscifactoría. Gestiona la cantidad de comida almacenada y la capacidad máxima del almacén, permitiendo añadir comida, retirar comida, consultar el espacio disponible y mejorar su capacidad.

private int stock

Cantidad de comida actualmente almacenada en el almacén. Inicia en 0.

private int maxCap

Capacidad máxima del almacén. Su valor inicial se establece en el constructor.

Constructores

- **public AlmacenComida(int maxCap)**: Constructor que inicializa el almacén de comida con una capacidad máxima (maxCap) específica y un stock inicial de 0.

Métodos

public void addFood(int ammount)

Método para añadir una cantidad específica de comida al stock del almacén.

Parámetro:

int ammount: Cantidad de comida a añadir al almacén.

public void upgrade(int capacityAdded): Aumenta la capacidad máxima del almacén de comida.

Parámetro:

int capacityAdded: Número de espacios adicionales a añadir a la capacidad del almacén.

public int getMaxCap()

Devuelve la capacidad máxima del almacén de comida.

public int getStock()

Devuelve la cantidad de comida almacenada en el almacén.

public void setStock(int stock)

Establece una nueva cantidad para el stock de comida.

Parámetro:

int stock: Nueva cantidad de comida en el almacén.

public int restarComida(int cantidad)

Resta una cantidad específica de comida del almacén.

Parámetro:

int cantidad: Cantidad de comida a restar.

public int getSpace()

Calcula el espacio libre disponible en el almacén.

public String toString(): Sobrescribe el método toString para representar el estado del almacén de comida. Devuelve una representación en cadena del estado del almacén, mostrando la cantidad de comida almacenada y la capacidad máxima.

Package edificios.piscifactoria

Clase Piscifactoría

Descripción

La clase Piscifactoria representa una piscifactoría dentro del sistema de gestión de peces. Contiene información sobre los tanques disponibles, la cantidad de peces en cada uno y el almacenamiento de comida. También proporciona métodos para gestionar el crecimiento de los peces, la alimentación, la venta de peces y la mejora de la capacidad de almacenamiento.

Atributos

private String name

Nombre de la piscifactoría.

private int maxFood
Capacidad máxima de comida que puede almacenar la piscifactoría.

private int currentFood
Cantidad de comida disponible actualmente en la piscifactoría.

private int maxTank
Número máximo de tanques que puede albergar la piscifactoría.

private int tankID
Identificador único de cada tanque.

private ArrayList<Tanque> tanques
Lista de tanques que forman parte de la piscifactoría.

protected AlmacenComida comidaVegetal
Instancia del almacén de comida vegetal.

protected AlmacenComida comidaAnimal
Instancia del almacén de comida animal.

Constructores

- **protected Piscifactoria(String name)**
  Constructor que inicializa la piscifactoría con el nombre especificado.

Parámetros:

String name: Nombre de la piscifactoría.

Métodos

public void showStatus()
Muestra en consola las estadísticas generales de la piscifactoría, incluyendo la ocupación de los tanques, cantidad de peces vivos, peces alimentados, proporción de sexos y estado del almacén de comida.

public void showTankStatus()
Muestra el estado de cada tanque de la piscifactoría.

public void showFishStatus(Tanque tanque)
Muestra la información de los peces en un tanque específico.

Parámetros:

Tanque tanque: Tanque a examinar.

public void showCapacity(Tanque tanque)
Muestra la capacidad de un tanque en relación con la piscifactoría.

Parámetros:

Tanque tanque: Tanque a examinar.

public void showFood()
Muestra el estado del almacén de comida de la piscifactoría, indicando la cantidad actual y la capacidad máxima.

Public void nextDay(Estadisticas estadisticas)
Simula el paso de un día en la piscifactoría, haciendo crecer los peces en cada tanque.

Parámetro:
Estadisticas estadisticas: Datos estadísticos del sistema.

public void sellFish()
Vende todos los peces adultos y vivos de la piscifactoría.

public abstract void upgradeFood()
Método abstracto que mejora la capacidad del almacén de comida.

public boolean canAddTanque()
Comprueba si es posible añadir un tanque a la piscifactoría.

public abstract void compraTanque()
Método abstracto que gestiona la compra de un tanque.

public void cleanTank(Tanque tank)
Vacía un tanque específico de la piscifactoría.

Parámetro:
Tanque tank: Tanque a limpiar.

public void listTanks()
Muestra una lista de los tanques de la piscifactoría.

@Override public String toString()
Devuelve una cadena con la información general de la piscifactoría.

Clase PiscifactoriaMar

Descripción

La clase PiscifactoriaMar representa una piscifactoría de tipo marino dentro del sistema. Es una subclase de Piscifactoria y añade funcionalidad específica como la compra de tanques marinos y la mejora de capacidad de almacenamiento de comida.

Atriibutos

private final int PRICE
Precio de construcción de una piscifactoría de mar, establecido en 2000 monedas.

private int tankID
Identificador único de los tanques de la piscifactoría.

private int maxFood
Cantidad máxima de comida que la piscifactoría puede almacenar.

private int maxTankCapacity
Capacidad máxima de peces permitidos en cada tanque.

Constructores

- **public PiscifactoriaMar(String name)**
  Constructor que inicializa una piscifactoría de mar con un nombre específico.

Parámetro:
String name: Nombre de la piscifactoría.

- **public PiscifactoriaMar(String name, int stock)**
  Constructor que inicializa una piscifactoría de mar con un stock inicial de comid.

Parámetro:
String name: Nombre de la piscifactoría.

int stock: Cantidad inicial de comida en los almacenes.

Métodos
@Override public void upgradeFood()
Mejora la capacidad del almacén de comida de la piscifactoría, incrementando su capacidad en 100 unidades por 200 monedas.

@Override public void compraTanque()
Permite comprar un tanque marino si hay espacio disponible en la piscifactoría.

Clase PiscifactoriaRio

Descripción
La clase PiscifactoriaRio representa una piscifactoría de tipo fluvial dentro del sistema. Es una subclase de Piscifactoria y añade funcionalidades específicas como la compra de tanques de río y la mejora de la capacidad de almacenamiento de comida.

Atributos
private final int PRICE
Precio de construcción de una piscifactoría de río, establecido en 500 monedas.

private int tankID
Identificador único de los tanques de la piscifactoría.

private int maxFood
Capacidad máxima de comida que la piscifactoría puede almacenar.

private int maxTankCapacity
Capacidad máxima de peces permitidos en cada tanque de río.

Constructores

- **public PiscifactoriaRio(String name)**
  Constructor que inicializa una piscifactoría de río con un nombre específico.

Parámetros:

String name: Nombre de la piscifactoría.

- **public PiscifactoriaRio(String name, int stock)**
  Constructor que inicializa una piscifactoría de río con un stock inicial de comida.

Parámetros:

String name: Nombre de la piscifactoría.

int stock: Cantidad inicial de comida en los almacenes.

Métodos

@Override public void upgradeFood()
Mejora la capacidad del almacén de comida de la piscifactoría, incrementando su capacidad en 25 unidades por 50 monedas.

@Override public void compraTanque()
Permite comprar un tanque de río si hay espacio disponible en la piscifactoría.

Package edificios.tanque

Clase Tanque

Descripción

La clase Tanque representa un tanque de peces en el sistema de piscifactoría. Se encarga de gestionar los peces almacenados, su alimentación, crecimiento, reproducción y limpieza. También permite calcular la cantidad de comida requerida, verificar la ocupación y vender peces maduros.

Atributos

private ArrayList<Pez> fishes
Lista que almacena los peces presentes en el tanque.

int maxCapacity
Capacidad máxima de peces que el tanque puede albergar. Tiene visibilidad package-private.

private int tankNum
Número identificador del tanque.

private String fishType
Tipo de pez que puede contener el tanque, inicializado como null.

private boolean type
Indica el tipo de tanque: true para río y false para mar.

Constructores

- **public Tanque(int maxCapacity, int tankNum, boolean type)**
  Inicializa un tanque con una capacidad máxima y un número de identificación.

Parámetros:

int maxCapacity: Capacidad máxima del tanque.

int tankNum: Número identificador del tanque.

boolean type: Tipo de tanque (true para río, false para mar).

Métodos

public void showStatus()
Muestra en consola el estado actual del tanque, incluyendo la ocupación, peces vivos, peces alimentados, adultos, proporción de hembras y machos, y peces fértiles.

public void showfishestatus()
Muestra el estado de cada pez en el tanque llamando al método showStatus() de cada pez.

public void showCapacity(Piscifactoria piscifactoria)
Muestra en consola el porcentaje de ocupación del tanque con el nombre de la piscifactoría.

Parámetro:

Piscifactoria piscifactoria: Piscifactoría a la que pertenece el tanqu.

public void addFishes(Pez fish)
Añade un pez al tanque si no está lleno y si el tipo de pez es correcto.

Parámetro:

Pez fish: Objeto pez a añadir.

public boolean fishMatch()
Comprueba si hay al menos un macho y una hembra fértiles en el tanque.

public void nextDay(Piscifactoria p, Estadisticas estadisticas)
Simula el paso de un día en el tanque, hace crecer a los peces y vende aquellos que han alcanzado la edad óptima.

Parámetros:

Piscifactoria p: Piscifactoría a la que pertenece el tanque.

Estadisticas estadisticas: Registro de estadísticas de nacimientos y ventas.

public void sellFishes()
Vende todos los peces maduros y vivos, actualizando el sistema de monedas SISMonedas.

public int fishesAlive()
Devuelve el número de peces vivos en el tanque.

public void cleanTank()
Elimina todos los peces del tanque.

public void cleanDeadFishes()
Elimina los peces muertos del tanque.

public int alimentedFishes()
Devuelve el número de peces que han comido y están vivos.

public int foodAmount()
Calcula la cantidad de comida necesaria para todos los peces del tanque.

public int matureFishes()
Devuelve el número de peces maduros y vivos.

public int fishesF()
Devuelve el número de peces hembra vivos.

public int fishesM()
Devuelve el número de peces macho vivos.

public int fertiles()
Devuelve el número de peces fértiles y vivos.

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

public boolean isEmpty()
Comprueba si el tanque está vacío.

public boolean isFull()
Comprueba si el tanque está lleno.

public int getMaxCapacity()
Devuelve la capacidad máxima del tanque.

public int getTankNum()
Devuelve el número del tanque.

@Override public String toString()
Devuelve una representación en cadena del estado del tanque.

public int contarEnfermos()
Cuenta el número de peces enfermos en el tanque.

public void propagarEnfermedad()
Simula la propagación de enfermedades en el tanque, teniendo en cuenta la presencia de peces muertos o enfermos.

public void curarPeces()
Cura todos los peces enfermos del tanque.

Package Peces

Clase Pez

Descripción

La clase abstracta Pez representa la estructura y el comportamiento básico de un pez en el sistema de piscifactoría. Define atributos fundamentales como edad, sexo, fertilidad, y métodos para gestionar su crecimiento, alimentación, reproducción y estado de vida.

Atributos

protected PecesDatos fishStats

Objeto PecesDatos que contiene las estadísticas del pez, como edad de madurez, fertilidad, y otros datos específicos.

protected final String name

Nombre común del pez.

protected final String scientifcName

Nombre científico del pez.

protected int age

Edad actual del pez en días, que comienza en 0 y aumenta con el tiempo.

protected final boolean sex

Sexo del pez; true indica macho y false indica hembra.

protected boolean fertile

Indica si el pez es fértil y está listo para reproducirse.

protected boolean alive

Estado de vida del pez; true indica que está vivo, false indica que está muerto.

protected boolean eat

Estado de alimentación del pez; true si ha comido, false si tiene hambre.

protected boolean mature

Indica si el pez ha alcanzado la madurez.

protected boolean reproducible

Indica si el pez puede reproducirse (true si es fértil y maduro).

protected int reproductionCycle

Ciclo de reproducción del pez, que disminuye cada día hasta llegar a cero, momento en el cual el pez se vuelve fértil.

Constructores

- **public Pez(boolean sex, PecesDatos fishStats)**: Inicializa un pez con estadísticas y sexo.

Parámetros:

boolean sex: Sexo del pez.

pecesDatos fishStats: estadísticas iniciales del pez.

Méteodos

public void showStatus()

Muestra en consola el estado actual del pez (edad, sexo, estado de alimentación, madurez, y fertilidad).

@Override public abstract Pez reproduce(boolean sex)
Método abstracto que debe ser implementado en las subclases para permitir la reproducción del pez.

public abstract int eat()
Método abstracto que simula la acción de comer del pez.

public void grow(boolean comido)
Incrementa la edad del pez en un día.

Parámetros:
boolean comido: Indica si el pez ha comida.

public void reproduce(Tanque tank)
Permite al pez reproducirse en un tanque si es fértil, añadiendo una cantidad de huevos definida en fishStats.

Parámetro:
Tanque tank: El tanque donde se agregarán los nuevos peces.

public void reset()
Reinicia los atributos del pez a su estado inicia

public PecesDatos getFishStats()
Devuelve el objeto PecesDatos que contiene las estadísticas del pez.

public boolean isAlive()
Indica si el pez está vivo.

public boolean isEat()
Indica si el pez ha comido.

public boolean isMature()
Indica si el pez ha alcanzado la madurez.

public boolean isFertile()
Indica si el pez es fértil.

public boolean isMale()
Devuelve true si el pez es macho.

public void setSex(boolean sex)
Establece el sexo del pez.

Parámetro:
boolean sex: Sexo del pez.

public void setEat(boolean eat)
Establece si el pez ha comido o no.

Parámetro:
boolean eat: Estado de alimentación.

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

public boolean isReproducible()
Indica si el pez puede reproducirse

public void infectar()
Marca al pez como enfermo.

public void setEnfermo(boolean enfermo)
Establece el estado de enfermedad del pez.

public void curar()
Cura al pez, eliminando su estado de enfermedad.

public boolean isEnfermo()
Indica si el pez está enfermo o no.

public void grow(boolean comido)
Simula el crecimiento del pez y maneja las probabilidades de muerte por enfermedad.

Parámetro:
comido: true si el pez ha comido, false en caso contrario

Package peces.especies

Clase Besugo

Descripción

La clase Besugo representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que el Besugo tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public Besugo(boolean sex):** Constructor que inicializa una instancia de la clase Besugo con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas del Besugo desde AlmacenPropiedades.BESUGO.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Besugo.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase Caballa

Descripción

La clase Caballa representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que la Caballa tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public Caballa(boolean sex):** Constructor que inicializa una instancia de la clase Caballa con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas de la Caballa desde AlmacenPropiedades.CABALLA

Métodos

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Caballa.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase CarpaPlateada

Descripción

La clase CarpaPlateada representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionFiltrador, lo que implica que la CarpaPlateada tiene un comportamiento de alimentación Filtrador y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionFiltrador y Pez:

protected PecesDatos fishStats

protected int age

protected boolean sex

protected boolean fertile

protected boolean alive

protected boolean eat

protected boolean mature

protected int reproductionCycle

Constructores

- **public CarpaPlateada (boolean sex):** Constructor que inicializa una instancia de la clase CarpaPlateada con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionFiltrador con las propiedades específicas del CarpaPlateada desde AlmacenPropiedadesCARPA_PLATEADA.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Carpa Plateada.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase LenguadoEuropeo

Descripción

La clase LenguadoEuropeo representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que la LenguadoEuropeo tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public LenguadoEuropeo (boolean sex):** Constructor que inicializa una instancia de la clase LenguadoEuropeo con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas del LenguadoEuropeo desde AlmacenPropiedades.LENGUADO_EUROPEO.

Métodos

@Override public Pez reproduce(boolean sex)
Método que crea y devuelve una nueva instancia de Lenguado Europeo.

@Override public String toString()
Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase LubinaEuropea

Descripción

La clase LubinaEuropea representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que la LubinaEuropea tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public LubinaEuropea (boolean sex):** Constructor que inicializa una instancia de la clase LubinaEuropea con el sexo especificado.

Parámetro:
boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas de la LubinaEuropea desde AlmacenPropiedades.LUBINA_EUROPEA.

Métodos

@Override public Pez reproduce(boolean sex)
Método que crea y devuelve una nueva instancia de Lubina Europea.

@Override public String toString()
Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase LubinaRayada

Descripción

La clase LubinaRayada representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que la LubinaRayada tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public LubinaRayada (boolean sex):** Constructor que inicializa una instancia de la clase LubinaRayada con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas de la LubinaRayadadesde AlmacenPropiedades.LUBINA_RAYADA.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Lubina Rayada.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase LucioDelNorte

Descripción

La clase LucioDelNorte representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoroActivo, lo que implica que el LucioDelNorte tiene un comportamiento de alimentación carnívora acitva y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoroActivo y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public LucioDelNorte (boolean sex):** Constructor que inicializa una instancia de la clase LucioDelNorte con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoroActivo con las propiedades específicas del LucioDelNorte de AlmacenPropiedades.LUCIO_DEL_NORTE.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Lucio del Norte.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase Pejerrey

Descripción

La clase Pejerrey representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que el Pejerrey tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public Pejerrey (boolean sex):** Constructor que inicializa una instancia de la clase Pejerrey con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas del Pejerrey AlmacenPropiedades. Pejerrey.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Pejerrey.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase PercaEuropea

Descripción

La clase PercaEuropea representa un pez específico en el sistema de piscifactoría. Hereda de la

clase AlimentacionCarnivoroActivo, lo que implica que el PercaEuropea tiene un comportamiento de alimentación carnívora acitva y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoroActivo y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public PercaEuropea (boolean sex):** Constructor que inicializa una instancia de la clase PercaEuropea con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoroActivo con las propiedades específicas del PercaEuropea de AlmacenPropiedades.PERCA_EUROPEA.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Perca Europea.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase Robalo

Descripción

La clase Robalo representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que el Robalo tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public Robalo (boolean sex):** Constructor que inicializa una instancia de la clase Robalo con el sexo especificado.

Parámetro:
boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas del Robalo desde AlmacenPropiedades.ROBALO.

Métodos

@Override public Pez reproduce(boolean sex)
Método que crea y devuelve una nueva instancia de Robalo.

@Override public String toString()
Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase SalmonAtlantico

Descripción

La clase SalmonAtlantico representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que el SalmonAtlantico tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public SalmonAtlantico (boolean sex):** Constructor que inicializa una instancia de la clase SalmonAtlantico con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas del SalmonAtlantico desde AlmacenPropiedades.SALMON_ATLANTICO.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Salmón Atlántico.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase SalmonChinook

Descripción

La clase SalmonChinook representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionCarnivoro, lo que implica que el SalmonChinook tiene un comportamiento de alimentación carnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionCarnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **public SalmonChinook (boolean sex):** Constructor que inicializa una instancia de la clase SalmonAtlantico con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionCarnivoro con las propiedades específicas del SalmonChinook desde AlmacenPropiedades.SALMON_CHINOOK.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Salmón Chinook.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Clase Dorada

Descripción

La clase Dorada representa un pez específico en el sistema de piscifactoría. Hereda de la clase AlimentacionOmnivoro, lo que implica que el Dorada tiene un comportamiento de alimentación omnívora y las propiedades básicas de un pez.

Atributos

Hereda todos los atributos de la clase AlimentacionOmnivoro y Pez:

protected PecesDatos fishStats
protected int age
protected boolean sex
protected boolean fertile
protected boolean alive
protected boolean eat
protected boolean mature
protected int reproductionCycle

Constructores

- **publicDorada (boolean sex):** Constructor que inicializa una instancia de la clase SalmonAtlantico con el sexo especificado.

Parámetro:

boolean sex: Sexo del pez, donde true representa macho y false representa hembra.

Llamada a super: Llama al constructor de la clase AlimentacionOmnivoro con las propiedades específicas del Dorada desde AlmacenPropiedades.DORADA.

Métodos

@Override public Pez reproduce(boolean sex)

Método que crea y devuelve una nueva instancia de Dorada.

@Override public String toString()

Sobrescribe el método toString() para proporcionar una representación detallada del estado del pez.

Package recompensas

Clase GenerarRecompensas

Descripción

La clase GenerarRecompensa gestiona la creación y almacenamiento de recompensas en formato XML. Genera distintos tipos de recompensas como alimento, monedas, tanques y piscifactorías, con distintos niveles de rareza y cantidad.

Atributos

private static final String PATH_RECOMPENSAS

Ruta base donde se guardarán los archivos XML de las recompensas.

Métodos

public Document generarArchivoXML()

Crea y devuelve un documento XML con la estructura base de una recompensa.

public void algaReward(int lvl)

Genera una recompensa de alga según el nivel especificado.

Parámetros:

int lvl: Nivel de la recompensa (1-5).

public void piensoReward(int lvl)

Genera una recompensa de pienso de peces.

Parámetros:

int lvl: Nivel de la recompensa (1-5).

public void BolivaresReward(int lvl)

Genera una recompensa de monedas.

Parámetros:

int lvl: Nivel de la recompensa (1-5).

public void comidaReward(int lvl)

Genera una recompensa de comida general para peces.

Parámetros:

int lvl: Nivel de la recompensa (1-5).

public void tanqueReward(String tipo)

Genera una recompensa de tanque (río o mar).

Parámetros:

String tipo: r para rio m para mar.

public void pisciMarReward(String tipo)

Genera una recompensa de piscifactoría de mar.

Parámetros:

String tipo: Partes a y b de la recompensa.

public void pisciRioReward(String tipo)
Genera una recompensa de piscifactoría de río.

Parámetros:
String tipo: Partes a y b de la recompensa.

public void almacenReward(String tipo)
Genera una recompensa de almacén central.

Parámetros:
String tipo: Partes a b c y d de la recompensa.

private void aumentarCantidad(File archivo)
Incrementa la cantidad de una recompensa en su archivo XML.

Parámetros:
File archivo: Archivo XML donde se incrementa la cantidad.

public void guardarXML(Document doc, String rutaArchivo)
Guarda un documento XML en la ruta especificada.

Parámetros:
Document doc: Documento XML a guardar.

String rutaArchivo: Ruta donde se guarda el archivo.

Clase GestorRecompensas

Descripción

La clase GestorRecompensas se encarga de la gestión de recompensas en formato XML dentro del sistema de piscifactoría. Permite la lectura, reducción, canjeo y procesamiento de recompensas almacenadas en archivos XML.

Atributos

private static final String PATH_RECOMPENSAS
Ruta base donde se almacenan los archivos XML de recompensas.

Métodos

private void reducirCantidad(String nombreFichero)
Reduce en 1 la cantidad de una recompensa dentro de un archivo XML y elimina el archivo si el valor llega a 0.

Parámetros:
String nombreFichero: Nombre del archivo XML de la recompensa.

public List<String> obtenerMenuRewards()
Genera una lista con los nombres de las recompensas disponibles en la carpeta rewards.

public void mostrarYCanjearReward(Simulador sim)
Muestra un menú con las recompensas disponibles y permite canjear una.

Parámetros:
Simulador sim: Instancioa del simulador donde se aplicara la recompensa.

private void canjearReward(String seleccion, Simulador sim)
Ejecuta el canje de una recompensa, aplicando sus efectos en el simulador.

Parámetros:
String seleccion: Nombre de la recompensa seleccionada.

Simulador sim: Instancia del simulador donde se aplica la recompensa.

private String generarNombreFichero(String base, String parte)
Genera el nombre del archivo XML para una recompensa multipartida.

Parámetros:
String base: Nombre base de la recompensa

String parte: Parte de la recompensa.

private void procesarPremio(Document doc, Simulador sim)
Lee el archivo XML de la recompensa y aplica sus efectos en el simulador

Parámetros:
Document doc: Documento XML de la recompensa.

Simulador sim: Instancia del simulador donde se aplica la recompesna.

private void guardarXML(Document doc, File archivo)
Guarda un documento XML actualizado en un archivo.

Parámetros:
Document doc: Documento XML de la recompensa.

File archivo: Archivo donde se almacena el XML.

Package registros

Clase Log

Descripcion

La clase Log implementa un sistema de registro de eventos en un archivo de logs.
Utiliza el patrón **Singleton** para asegurar que solo exista una única instancia en la aplicación.
Registra eventos clave dentro del simulador, como compras, ventas, cambios de día y errores.

Atributos

private static Log instance
Instancia única de la clase Log (Singleton).

private static final String PATH_LOG
Directorio donde se almacenan los archivos de log.

private BufferedWriter bw
Objeto para escribir en el archivo de log.

private String rutaArchivo
Ruta del archivo de log correspondiente a la partida actual.

private static final String ERROR_LOG_FILE
Ruta del archivo de log exclusivo para errores.

Constructores

private Log(String nombrePartida):
Constructor privado que crea o abre el archivo de log asociado a la partida.

Parámetros:

String nombrePartida: Nombre de la partida para guardar en el Log.

public static Log getInstance(String nombrePartida)
Método estático para obtener la única instancia de Log.

Parámetros:

String nombrePartida: Nombre de la partida para guardar en el Log.

private String getFecha()
Obtiene la fecha y hora actual en formato "yyyy-MM-dd HH:mm:ss".

public void write(String linea)
Escribe una línea en el archivo de log, precedida por la fecha y hora.

Parámetros:

String línea: Mensaje a escribir en el Log

public void logStart(String nombrePartida, String pisciInicial)
Registra el inicio de una partida y su piscifactoría inicial.

Parámetros:

String nombrePartida: Nombre de la partida.

String pisciInicial: Nombre de la piscifactoría inicial.

public void logComprarComida(int cantidadComida, String tipoComida, boolean almacen, String piscifactoria)
Registra la compra de comida en el log.

Parámetros:

int cantidadComida: Cantidad de comida comprada.

String tipoComida: Tipo de comida ("vegetal", "animal", etc.).

boolean almacen: true si se almacena en el almacén central, false si se almacena en una piscifactoría.

String piscifactoria: Nombre de la piscifactoría (si aplica).

public void logComprarPeces(String nombrePez, boolean tipoSex, int tanque, String piscifactoria)

Registra la compra de un pez.

Parámetros:

String nombrePez: Nombre del pez comprado.

boolean tipoSex: true si el pez es macho, false si es hembra.

int tanque: Número del tanque donde se almacena.

String piscifactoria: Piscifactoría donde se compra el pez.

public void logVenderPeces(int peces, String piscifactoria)

Registra la venta de peces.

Parámetros:

int peces: Cantidad de peces vendidos.

String piscifactoria: Piscifactoría donde se realiza la venta.

public void logLimpiarTanque(int tanque, String piscifactoria)

Registra la limpieza de un tanque.

Parámetros:

int tanque: Número del tanque limpiado.

String piscifactoria: Piscifactoría donde está el tanque.

public void logVaciarTanque(int tanque, String piscifactoria)

Registra el vaciado de un tanque.

Parámetros:

int tanque: Número del tanque vaciado.

String piscifactoria: Piscifactoría donde está el tanque.

public void logCompraEdificio(String tipoEdificio, String piscifactoria)

Registra la compra de un edificio.

Parámetros:

String tipoEdificio: Tipo de edificio ("tanque", "almacen", "mar", "rio").

String piscifactoria: Piscifactoría donde se realiza la compra.

public void logMejorarEdificio(String piscifactoria)

Registra la mejora de una piscifactoría.

Parámetros:

String piscifactoria: Piscifactoría mejorada.

public void logNextDay(int day)
Registra el paso al siguiente día.

Parámetros:
int day: Número del día actual.

public void logOpsOcultas(int codOp, String piscifactoria)
Registra operaciones ocultas en el log.

Parámetros:
int codOp: Código de la operación (98 para añadir peces, 99 para añadir monedas).

String piscifactoria: Piscifactoría donde se realizó la operación (si aplica).

public void logSalir()
Registra el cierre de la partida.

public void logCreaRecompensa(String recompensa)
Registra la creación de una recompensa.

Parámetros:
String recompensa: Nombre de la recompensa creada.

public void logUsaRecompensa(String recompensa)
Registra el uso de una recompensa.

Parámetros:
String recompensa: Nombre de la recompensa usada.

public void logGuardadoPartida(String partida)
Registra el guardado de la partida.

Parámetros:
String partida: Nombre de la partida guardada.

public void logError(String mensaje)
Registra un error en el archivo de log de errores.

Parámetros:
String mensaje: Mensaje de error.

public void logClose()
Cierra el archivo de log y libera los recursos.

Clase Registros

Descripción

La clase Registros gestiona el sistema de registro de eventos en el simulador.
Facilita la transcripción de eventos clave mediante Transcripciones y Log.
Sigue el patrón Singleton, asegurando que solo haya una instancia por partida.

Atributos

private static Transcripciones transcripciones
Instancia utilizada para delegar las transcripciones de eventos.

private static Log log
Instancia utilizada para gestionar el registro de logs de eventos.

private static Registros instance
Instancia única de la clase Registros (Singleton).

Métodos

public static Registros getInstance()
Obtiene la instancia única de la clase Registros.

public void crearRegistro(String nombrePartida)
Crea un nuevo registro de transcripciones y logs para la partida.

Parámetros:
String nombrePartida: Nombre de la partida para generar los registros.

public static void registrarInicio(String nombrePartida, int monedas, String[] pecesImplementados, String piscInicial)
Registra el inicio de la partida.

Parámetros:
String nombrePartida: Nombre de la partida.

int monedas: Cantidad de monedas iniciales.

String[] pecesImplementados: Lista de peces implementados.

String piscInicial: Nombre de la piscifactoría inicial.

public static void registrarCompraComida(int cantidadComida, String tipoComida, int monedasGastadas, boolean almacen, String piscifactoria)
Registra la compra de comida.

Parámetros:
int cantidadComida: Cantidad de comida comprada.

String tipoComida: Tipo de comida comprada.

int monedasGastadas: Monedas gastadas en la compra.

boolean almacen: true si se almacena en el almacén central, false si en la piscifactoría.

String piscifactoria: Piscifactoría en la que se realiza la compra.

public static void registrarComprarPeces(String nombrePez, boolean tipoSex, int monedas, int tanque, String piscifactoria)
Registra la compra de peces.

Parámetros:
String nombrePez: Nombre del pez comprado.

boolean tipoSex: true si el pez es macho, false si es hembra.

int monedas: Costo de compra.

int tanque: Número del tanque donde se almacena.

String piscifactoria: Piscifactoría donde se compra el pez.

public static void registrarVenderPeces(int peces, String piscifactoria, int bolivares)
Registra la venta de peces.

Parámetros:
int peces: Cantidad de peces vendidos.

String piscifactoria: Piscifactoría donde se realiza la venta.

int bolivares: Monedas obtenidas por la venta.

public static void registrarLimpiarTanque(int tanque, String piscifactoria)
Registra la limpieza de un tanque.

Parámetros:
int tanque: Número del tanque limpiado.

String piscifactoria: Piscifactoría donde está el tanque.

public static void registrarVaciarTanque(int tanque, String piscifactoria)
Registra el vaciado de un tanque.

Parámetros:
int tanque: Número del tanque vaciado.

String piscifactoria: Piscifactoría donde está el tanque.

public static void registrarCompraEdificio(String tipoEdificio, int monedas, int tanque, String piscifactoria)
Registra la compra de un edificio.

Parámetros:
String tipoEdificio: Tipo de edificio ("tanque", "almacen", "mar", "rio").

int monedas: Monedas gastadas en la compra.

int tanque: Número de tanque (si aplica).

String piscifactoria: Piscifactoría donde se realiza la compra.

public static void registrarMejoraEdificio(String piscifactoria, int aumentoComida, int monedas)

Registra la mejora de una piscifactoría.

Parámetros:

String piscifactoria: Piscifactoría mejorada.

int aumentoComida: Cantidad de comida aumentada en la mejora.

int monedas: Monedas gastadas en la mejora.

---

public static void regitrarNextDay(int dia, int pecesRio, int pecesMar, int monedas, int pecesVendidos)

Registra el paso al siguiente día.

Parámetros:

int dia: Número del día actual.

int pecesRio: Cantidad de peces en piscifactorías de río.

int pecesMar: Cantidad de peces en piscifactorías de mar.

int monedas: Cantidad de monedas actuales.

int pecesVendidos: Cantidad de peces vendidos en el día.

public static void registraropsOcultasint(int codOp, String piscifactoria, int cantidadDolares)

Registra operaciones ocultas en el log.

Parámetros:

int codOp: Código de la operación (98 para añadir peces, 99 para añadir monedas).

String piscifactoria: Piscifactoría donde se realizó la operación (si aplica).

int cantidadDolares: Cantidad de monedas añadidas en la operación oculta

public static void registrarSalir()

Registra el cierre de la partida.

public static void registraCreaRecompensa(String recompensa)

Registra la creación de una recompensa.

Parámetros:

String recompensa: Nombre de la recompensa creada.

public static void registraUsoRecompensa(String recompensa)

Registra el uso de una recompensa.

Parámetros:

String recompensa: Nombre de la recompensa usada.

public static void registrarError(String error)

Registra un error en el archivo de log de errores.

Parámetros:

String error: Mensaje de error.

public static void registrarCuraPeces(int peces, String piscifactoria, int monedas)

Registra la curación de peces en la piscifactoría.

Parámetros:

int peces: Cantidad de peces curados.

String piscifactoria: Piscifactoría donde se realiza la curación.

int monedas: Costo en monedas de la curación.

public static void cerrarRegistros()

Cierra los registros de transcripciones y logs.

Clase Transcripciones

Descsripción

La clase Transcripciones gestiona la escritura de eventos en un archivo de transcripción. Sigue el patrón Singleton, asegurando que solo haya una instancia por partida.

Atributos

private static Transcripciones instance

Instancia única de la clase Transcripciones.

private static final String PATH_TRANS

Ruta del directorio donde se almacenan las transcripciones.

private BufferedWriter bw

Buffer de escritura para el archivo de transcripción.

private String rutaArchivo

Ruta del archivo de transcripción correspondiente a la partida.

Métodos

public static Transcripciones getInstance(String nombrePartida)

Parámetros:

String nombrePartida: Nombre de la partida para generar el archivo.

public void escribir(String linea)

Escribe una línea en el archivo de transcripciones.

Parámetros:

String linea: Contenido de la línea a insertar.

public void trStart(String nombrePartida, int monedas, String[] pecesImplementados, String pisciInicial)

Registra el inicio de la partida junto con la información principal del sistema.

Parámetros:

String nombrePartida: Nombre de la partida.

int monedas: Dinero inicial.

String[] pecesImplementados: Lista de peces implementados.

String pisciInicial: Piscifactoría inicial.

public void trComprarComdida(int cantidadComida, String tipoComida, int monedasGastadas, boolean almacen, String piscifactoria)

Registra la compra de comida.

Parámetros:

int cantidadComida: Cantidad de comida comprada.

String tipoComida: Tipo de comida comprada.

int monedasGastadas: Monedas gastadas en la compra.

boolean almacen: true si se almacena en el almacén central, false si en la piscifactoría.

String piscifactoria: Piscifactoría donde se almacena la comida.

public void trComprarPeces(String nombrePez, boolean tipoSex, int monedas, int tanque, String piscifactoria)

Registra la compra de peces.

Parámetros:

String nombrePez: Nombre del pez comprado.

boolean tipoSex: true si el pez es macho, false si es hembra.

int monedas: Monedas gastadas en la compra.

int tanque: Número del tanque donde se almacena el pez.

String piscifactoria: Piscifactoría donde se compra el pez.

public void trVenderPeces(int peces, String nombrePisci, int yenes)

Registra la venta de peces.

Parámetros:

int peces: Cantidad de peces vendidos.

String nombrePisci: Piscifactoría donde se realizó la venta.

int yenes: Monedas obtenidas por la venta.

public void trLimpiarTanque(int tanque, String piscifactoria)
Registra la limpieza de un tanque.

Parámetros:
int tanque: Número del tanque limpiado.

String piscifactoria: Piscifactoría donde está el tanque.

public void trVaciarTanque(int tanque, String piscifactoria)
Registra el vaciado de un tanque.

Parámetros:
int tanque: Número del tanque vaciado.

String piscifactoria: Piscifactoría donde está el tanque.

public void trCompraEdificio(String tipoEdificio, int monedas, int tanque, String piscifactoria)
Registra la compra de un edificio.

Parámetros:
String tipoEdificio: Tipo de edificio ("tanque", "almacen", "mar", "rio").

int monedas: Monedas gastadas en la compra.

int tanque: Número de tanque (si aplica).

String piscifactoria: Piscifactoría donde se realiza la compra.

public void trMejoraEdificio(String piscifactoria, int aumentoComida, int monedas)
Registra la mejora de una piscifactoría.

Parámetros:
String piscifactoria: Piscifactoría mejorada.

int aumentoComida: Cantidad de comida aumentada en la mejora.

int monedas: Monedas gastadas en la mejora.

public void trNextDay(int dia, int pecesRio, int pecesMar, int monedas, int pecesVendidos)
Registra el paso al siguiente día.

Parámetros:
int dia: Número del día actual.

int pecesRio: Cantidad de peces en piscifactorías de río.

int pecesMar: Cantidad de peces en piscifactorías de mar.

int monedas: Cantidad de monedas actuales.

int pecesVendidos: Cantidad de peces vendidos en el día.

public void trOpsOcultas(int codOp, String piscifactoria, int cantidadDolares)
Registra el uso de opciones ocultas en el simulador.

Parámetros:
int codOp: Código de la operación (98 para añadir peces, 99 para añadir monedas).

String piscifactoria: Piscifactoría donde se realizó la operación (si aplica).

int cantidadDolares: Cantidad de monedas añadidas en la operación oculta.

public void trCreaRecompensas(String recompensa)
Registra la creación de una recompensa.

Parámetros:
String recompensa: Nombre de la recompensa creada.

public void trUsaRecompensas(String recompensa)
Registra el uso de una recompensa.

Parámetros:
String recompensa: Nombre de la recompensa usada.

public void trCurapeces(int peces, String piscifactoria, int monedas)
Registra la curación de peces en la piscifactoría.

Parámetros:
int peces: Cantidad de peces curados.

String piscifactoria: Piscifactoría donde se realizó la curación.

int monedas: Costo en monedas de la curación.

public void trCerrar()
Cierra el archivo de transcripciones liberando los recursos.

Package peces.alimentación

Clase AlimentaciónCarnivora

Descripción

La clase AlimentacionCarnivoro representa el comportamiento de alimentación carnívora de un pez en el sistema de piscifactoría. Extiende la clase Pez.

Constructores

- **public AlimentacionCarnivoro(PecesDatos fishStats, boolean sex)**

Parámetros:
PecesDatos fishStats: estadísticas iniciales del pez.

boolean sex: Sexo del pez.

Métodos

public int eat()

Sobrescribe el método eat de la clase Pez para simular la acción de alimentación del pez carnívoro.

@Override public abstract Pez reproduce(boolean sex)

Método abstracto que debe ser implementado en las subclases para crear una nueva cría de pez carnívoro.

Parámetros:

boolean sex: Sexo del nuevo pez (true para macho, false para hembra).

Clase AlimentaciónCarnivoroActivo

Descripción

La clase AlimentacionCarnivoroActivo representa el comportamiento de alimentación carnívora activa de un pez en el sistema de piscifactoría. Extiende la clase Pez.

Constructores

- **public AlimentacionCarnivoroActivo(PecesDatos fishStats, boolean sex)**

Parámetros:

- **PecesDatos fishStats**: Objeto que contiene las estadísticas iniciales del pez, como edad, fertilidad, y otros datos relevantes.
- **boolean sex:** Sexo del pez, donde true indica macho y false indica hembra.

Métodos

public int eat()

Sobrescribe el método eat de la clase Pez para simular la acción de alimentación del pez carnívoro activo.

@Override public abstract Pez reproduce(boolean sex)

Método abstracto que debe ser implementado en las subclases para crear una nueva cría de pez carnívoro activo.

Parámetros:

boolean sex: Sexo del nuevo pez (true para macho, false para hembra).

Clase AlimentaciónFiltrador

Descripción

La clase AlimentacionFiltrador representa el comportamiento de alimentación filtradora de un pez en el sistema de piscifactoría. Extiende la clase Pez.

Constructores

- **public AlimentacionCarnivoroActivo(PecesDatos fishStats, boolean sex)**

Parámetros:

- **PecesDatos fishStats**: Objeto que contiene las estadísticas iniciales del pez, como edad, fertilidad, y otros datos relevantes.

- **boolean sex:** Sexo del pez, donde true indica macho y false indica hembra.

Métodos

public int eat()

Sobrescribe el método eat de la clase Pez para simular la acción de alimentación del pez filtrador.

@Override public abstract Pez reproduce(boolean sex)

Método que crea un nuevo pez. Este método no está implementado y lanzará una excepción.

Clase AlimentacionOmnivoro

Descripción

La clase AlimentacionOmnivoro representa el comportamiento de alimentación de un pez omnívoro dentro del sistema de piscifactoría.Extiende de la clase Pez.

Constructores

public AlimentacionOmnivoro(PecesDatos fishStats, boolean sex)

Constructor que inicializa un pez omnívoro con sus características específicas.

Parámetros:

PecesDatos fishStats: Objeto que almacena las estadísticas del pez (edad de madurez, fertilidad, cantidad de huevos, etc.).

boolean sex: Sexo del pez (true para macho, false para hembra)

Métodos

@Override public int eat()

Sobrescribe el método eat() de la clase Pez para simular la alimentación del pez omnívoro.

@Override public abstract Pez reproduce(boolean sex)

Método abstracto que debe ser implementado en las subclases para crear una nueva cría de pez omnívoro.

Parámetros:

boolean sex: Sexo del nuevo pez (true para macho, false para hembra)

Package Sistemas

Clase Helper

Descripción

La clase Helper proporciona utilidades para la interacción con el usuario en el sistema de piscifactoría.

Métodos

public int mostrarMenu(String titulo, String[] opciones, int[] extraOps)

Parámetros:

- **String titulo:** Título del menú que se mostrará en pantalla.
- **String[] opciones**: Array con las opciones disponibles en el menú.
- **int[] extraOps**: Array con opciones adicionales que no aparecen en el menú principal.

public int leerOpcion(int maxOpcion, int[] extraOps)

Parámetros:

- **int maxOpcion:** Valor máximo permitido para la opción seleccionada (incluye la opción de salir).
- **int[] extraOps:** Opciones adicionales permitidas que no están en el menú principal.

private boolean isExtraOp(int opcion, int[] extraOps)

Parámetros:

- **int opción:** Opción seleccionada por el usuario.
- **int[] extraOps:** Array con opciones adicionales.

Clase SISMonedas

Descripción

La clase SISMonedas representa el sistema de gestión de monedas en la piscifactoría.
Implementa el patrón Singleton para garantizar que solo exista una única instancia del sistema de monedas en toda la aplicación.

Atributos

public static SISMonedas saldo

nstancia única de la clase SISMonedas, utilizada para implementar el patrón Singleton.

protected int monedas

Número de monedas disponibles en el sistema. Se inicializa con un valor de 100.

Constructores

- **private SISMonedas()**
  Constructor privado que inicializa el saldo de monedas a 100.

Métodos

public static SISMonedas saldo

Instancia única de la clase SISMonedas, utilizada para implementar el patrón Singleton.

protected int monedas

Número de monedas disponibles en el sistema. Se inicializa con un valor de 100.

Métodos

public static SISMonedas getInstance()

Devuelve la única instancia de la clase SISMonedas. Si la instancia aún no ha sido creada, la cera.

public void pagar(int cantidad)

Permite realizar un pago restando la cantidad especificada de monedas del saldo actual.

Parámetro:

int cantidad: Número de monedas a pagar. **Funcionamiento**:

public int getMonedas()

Devuelve la cantidad de monedas disponibles.

public void setMonedas(int monedas)
Establece una nueva cantidad de monedas en el saldo.
Parámetro:

int monedas: La nueva cantidad de monedas a establecer.

public static SISMonedas getSaldo()
Devuelve la instancia actual de SISMonedas (el saldo).

public static void setSaldo(SISMonedas saldo)
Establece una nueva instancia de SISMonedas.
Parámetro:

SISMonedas saldo: La nueva instancia de SISMonedas a establecer.

Clase Simulador

Descripcion

La clase Simulador gestiona la simulación de una piscifactoría, permitiendo la administración de piscifactorías, tanques, peces y recursos. También maneja estadísticas, recompensas y eventos del sistema.

Atributos

private String name
Nombre de la empresa o partida.

private int days
Número de días transcurridos en la simulación.

private List<Piscifactoria> fishFarms
Lista de piscifactorías en el simulador.

private SISMonedas monedas
Sistema de monedas de la simulación.

private Helper helper
Instancia del Helper para gestionar menús y opciones.

private AlmacenCentral centralWarehouse
Instancia del almacén central.

private Estadisticas estadisticas
Instancia de la clase encargada de gestionar estadísticas del simulador.

private GenerarRecompensa generar
Instancia de la clase que genera recompensas.

Métodos

public void init()
Inicializa el sistema, creando carpetas necesarias y configurando la primera piscifactoría.

public void menu()
Muestra el menú principal del simulador y gestiona las opciones seleccionadas

public void menuPisc()
Muestra el menú con la lista de piscifactorías disponibles.

public void selectPisc()
Permite seleccionar una piscifactoría entre las disponibles.

public void selectTank()
Permite seleccionar un tanque de una piscifactoría.

public void showIctio()
Muestra información detallada sobre un pez seleccionado.

public void showGeneralStatus()
Muestra el estado general del simulador, incluyendo días, monedas y piscifactorías.

public void shoSpecificStatus()
Permite seleccionar una piscifactoría y mostrar su estado.

public void showTankStatus()
Permite seleccionar u tanque y mostrar la información de los peces.

public void netDay(**int days**)
Avanza la simulación en el número de días especificado.

Parámetro:

days Número de días a avanzar.

public void addFood()
Permite añadir comida a una piscifactoría o al almacén central.

public void showStats()
Muestra las estadísticas del simulador.

public void sellFish()
Permite vender peces maduros y vivos.

public void cleanTank()
Limpia un tanque eliminando peces muertos.

public void emptyTank()
Vacía un tanque eliminando todos los peces.

public void upgrade()
Gestiona la compra y mejora de edificios.

public void buyBuildings()
Permite comprar nuevos edificios, como piscifactorías o el almacén central.

public void upgradeBuildings()
Permite mejorar los edificios existentes.

public void upgradeFisFarm()
Permite mejorar una piscifactoría específica.

public void upgradeCentralWarehouse()
Permite mejorar el almacén central.

public void addish()
Permite añadir pces a un tanque.

public void procesaOpcion99()
Añade 1000 monedas al saldo del simulador.

public void addFishAmmount()
Añade cuatro peces aleatorios a un tanque seleccionado.

public int getSeaFishfarms()
Devuelve el número de piscifactorías de mar en el simulador.

public int getRiverFishfarms()
Devuelve el número de piscifactorías de río en el simulador.

public void crearCarpetas(String... nombresCarpetas)
Crea múltiples carpetas según los nombres especificados.

Parámetro:

nombresCarpetas Lista de nombres de carpetas a crear.

public void gestionarPedidos()
Gestiona los pedidos de peces pendientes.

public void pedidilloAutomata()
Genera pedidos automáticos de peces.

public AlmacenCentral getAlmacenCentral()
Devuelve el almacén central del simulador.

public void setAlmacenCentral(AlmacenCentral almacen)
Asigna un almacén central al simulador.

Parámetro:

almacen Instancia del almacén central.

public void addPisci(Piscifactoria p)
Añade una piscifactoría al simulador.

Parámetro:

p Piscifactoría a añadir.

public Piscifactoria selectPisciMar()
Permite seleccionar una piscifactoría de mar.

public Piscifactoria selectPisciRio()
Permite seleccionar una piscifactoría de río.

public void repartirComidaAnimal(**int cantidad**)
Reparte comida animal entre las piscifactorías.

Parámetro:

cantidad Cantidad de comida a repartir.

public void repartirComidaVegetal(**int cantidad**)
Reparte comida vegetal entre las piscifactorías.

Parámetro:

cantidad Cantidad de comida a repartir.

public int getDays()
Devuelve el número de días transcurridos en el simulador.

public List<Piscifactoria> getFishFarms()
Devuelve la lista de piscifactorías en el simulador.

public String getName()
Devuelve el nombre de la empresa o partida.

public SISMonedas getMonedas()
Devuelve el sistema de monedas del simulador.

public AlmacenCentral getCentralWarehouse()
Devuelve el almacén central del simulador.

public String[] getPecesDisponibles()
Devuelve un array con los peces disponibles.

public Helper getHelper()
Devuelve la instancia del Helper.

public String[] getFishesNames()
Devuelve un array con los nombres de los peces disponibles.

public Estadisticas getEstadisticas()
Devuelve el objeto que gestiona las estadísticas del simulador.

public GenerarRecompensa getGenerar()
Devuelve el objeto encargado de generar recompensas.

public void setName(String nombre)
Establece el nombre de la empresa o partida.

Parámetro:

nombre Nombre de la empresa.

public void setDays(int dia)
Establece el número de días del simulador.

Parámetro:

dia Número de días.

public void setMonedas(SISMonedas monedas)
Establece el sistema de monedas del simulador.

Parámetro:

monedas Instancia de SISMonedas.

public static void main(String[] args)
Punto de entrada principal del simulador.