MANUAL TÉCNICO
PISCIFACTORÍA

FRANCISCO MISER JUNQUERA
MARCOS VIDAL GONZÁLEZ
DAVID TORRES RIAL

2º DAM
02 / 11 / 2024
ACCESO A DATOS

ÍNDICE

1. Clase AlmacenCentral....8
   1.1 Descripción....8
   1.2 Funcionalidades....8
   1.3 Constructor....9
   1.4 Atributos....9
   1.5 Métodos....10
      public boolean construir():....10
      public void aumentarCapacidad(int aumento):....10
      public boolean añadirComidaAnimal(int cantidad):....10
      public boolean añadirComidaVegetal(int cantidad):....10
      public int calcularCosto(int cantidad):....10
      public void mostrarEstado():....10
      public int getCapacidadAlmacen():....10
      public void setCapacidadAlmacen(int capacidadAlmacen):....11
      public int getCantidadComidaAnimal():....11
      public void setCantidadComidaAnimal(int cantidadComidaAnimal):....11
      public int getCantidadComidaVegetal():....11
      public void setCantidadComidaVegetal(int cantidadComidaVegetal):....11
      public boolean isConstruido():....11
2. Clase SistemaMonedas....12
   2.1 Descripción....12
   2.2 Funcionalidades....12
   2.3 Constructor....12
   2.4 Atributos....12
   2.5 Métodos....13
      public int getMonedas():....13
      public boolean ganarMonedas(int cantidad):....13
      public boolean gastarMonedas(int costo):....13
3. Clase InputHelper....14
   3.1 Descripción....14
   3.2 Funcionalidades....14
   3.3 Constructor....14
   3.4 Atributos....14
   3.5 Métodos....15
      public String readString(String prompt):....15
      public int readInt(String prompt):....15
      public double readDouble(String prompt):....15
      public void close():....15
4. Clase MenuHelper....16
   4.1 Descripción....16
   4.2 Funcionalidades....16
   4.3 Constructor....16
   4.4 Atributos....16
   4.5 Métodos....17
      public void addOption(int optionNumber, String description):....17
      public void showMenu():....17
      public void clearOptions():....17
      public void executeOption(int optionNumber):....17
5. Interfaz Activo....18
   5.1 Descripción....18

5.2 Funcionalidades....18
6. Interfaz Carnivoro....19
6.1 Descripción....19
6.2 Funcionalidades....19
7. Interfaz Filtrador....20
7.1 Descripción....20
7.2 Funcionalidades....20
8. Clase SalmonAtlantico....21
8.1 Descripción....21
8.2 Funcionalidades....21
8.3 Constructor....21
8.4 Atributos....21
8.5 Métodos....21
9. Clase TruchaArcoiris....22
9.1 Descripción....22
9.2 Funcionalidades....22
9.3 Constructor....22
9.4 Atributos....22
9.5 Métodos....22
10. Clase ArenqueDelAtlantico....23
10.1 Descripción....23
10.2 Funcionalidades....23
10.3 Constructor....23
10.4 Atributos....23
10.5 Métodos....23
11. Clase Besugo....24
11.1 Descripción....24
11.2 Funcionalidades....24
11.3 Constructor....24
11.4 Atributos....24
11.5 Métodos....24
12. Clase LenguadoEuropeo....25
12.1 Descripción....25
12.2 Funcionalidades....25
12.3 Constructor....25
12.4 Atributos....25
12.5 Métodos....25
13. Clase LubinaRayada....26
13.1 Descripción....26
13.2 Funcionalidades....26
13.3 Constructor....26
13.4 Atributos....26
13.5 Métodos....26
14. Clase Robalo....27
14.1 Descripción....27
14.2 Funcionalidades....27
14.3 Constructor....27
14.4 Atributos....27

14.5 Métodos....27
15. Clase CarpaPlateada....28
15.1 Descripción....28
15.2 Funcionalidades....28
15.3 Constructor....28
15.4 Atributos....28
15.5 Métodos....28
16. Clase Pejerrey....29
16.1 Descripción....29
16.2 Funcionalidades....29
16.3 Constructor....29
16.4 Atributos....29
16.5 Métodos....29
17. Clase PercaEuropea....30
17.1 Descripción....30
17.2 Funcionalidades....30
17.3 Constructor....30
17.4 Atributos....30
17.5 Métodos....30
18. Clase SalmonChinook....31
18.1 Descripción....31
18.2 Funcionalidades....31
18.3 Constructor....31
18.4 Atributos....31
18.5 Métodos....31
19. Clase TilapiaDelNilo....32
19.1 Descripción....32
19.2 Funcionalidades....32
19.3 Constructor....32
19.4 Atributos....32
19.5 Métodos....32
20. Clase Pez....33
20.1 Descripción....33
20.2 Funcionalidades....33
20.3 Constructor....33
20.4 Atributos....33
20.5 Métodos....34
public void showStatus()....34
public void grow()....34
public void reset()....34
public Pez clonar(boolean nuevoSexo)....34
public String getNombre()....34
public String getNombreCientifico()....34
public int getEdad()....34
public boolean isSexo()....35
public boolean isFertil()....35
public boolean isVivo()....35
public boolean isAlimentado()....35

public int getCiclo()....35
public PecesDatos getDatos()....35
public void setEdad(int edad)....35
public void setVivo(boolean vivo)....35
public void setAlimentado(boolean alimentado)....35
public void setCiclo(int ciclo)....35
public void setDatos(PecesDatos datos)....36
public void setFertil(boolean fertil)....36
21. Clase PiscifactoriaDeMar....37
21.1 Descripción....37
21.2 Funcionalidades....37
21.3 Constructor....37
21.4 Atributos....37
21.5 Métodos....38
public boolean añadirTanque()....38
public int getTanquesMar()....38
22. Clase PiscifactoriaDeRio....39
22.1 Descripción....39
22.2 Funcionalidades....39
22.3 Constructor....39
22.4 Atributos....39
22.5 Métodos....40
public boolean añadirTanque()....40
public int getTanquesRio()....40
23. Clase Piscifactoria....41
23.1 Descripción....41
23.2 Funcionalidades....41
23.3 Constructor....41
23.4 Atributos....42
23.5 Métodos....42
public void agregarTanque(Tanque tanque):....42
public List<Tanque> getTanques():....42
public void addPez(Pez pez):....42
public void showStatus():....43
public void showTankStatus():....43
public void showFishStatus(int numeroTanque):....43
public void showCapacity(int numeroTanque):....43
public void showFood():....43
public void nextDay():....43
private void alimentarPeces(Tanque tanque):....43
public void sellFish():....43
public void upgradeFood(int incremento):....43
public boolean verificarTanqueYPiscifactoria(Tanque tanqueBuscado):....43
public int getTotalPeces():....43
public int getTotalVivos():....44
public int getTotalAlimentados():....44
public int getTotalAdultos():....44
public int getTotalHembras()....44

public int getTotalMachos():....44
public int getTotalFertiles():....44
public int getCapacidadTotal():....44
public String getNombre():....44
public void setNombre(String nombre):....44
public void setNumeroTanquesDeRio(int numeroTanquesDeRio):....44
public void setNumeroTanquesDeMar(int numeroTanquesDeMar):....44
public int getNumeroTanquesDeRio():....45
public int getNumeroTanquesDeMar():....45
public int getComidaVegetalActual():....45
public void setComidaVegetalActual(int comidaVegetalActual):....45
public void setComidaAnimalActual(int comidaAnimalActual):....45
public int getComidaAnimalActual():....45
24. Clase Tanque....46
24.1 Descripción....46
24.2 Funcionalidades....46
24.3 Constructor....46
24.4 Atributos....46
24.5 Métodos....47
public void showStatus():....47
public void showFishStatus():....47
public void showCapacity():....47
public void nextDay():....47
public void reproduccion():....47
public boolean addPez(Pez pez):....47
public Class<?> getTipoPezActual():....47
public int getNumeroTanque():....47
public int getCapacidad():....47
public int getNumPeces():....47
public ArrayList<Pez> getPeces():....48
public int getHembras():....48
public int getMachos():....48
public int getFertiles():....48
public int getVivos():....48
public int getAlimentados():....48
public int getAdultos():....48
25. Clase Simulador....49
25.1 Descripción....49
25.2 Funcionalidades....49
25.3 Constructor....49
25.4 Atributos....49
25.5 Métodos....50
public void init():....50
public void menu():....50
public void showGeneralStatus():....50
public void showSpecificStatus():....50
public void showTankStatus():....50
public void showStats():....50

public void showIctio():.....50
public void nextDay():.....51
public void addFood():.....51
public void addFish():.....51
public void sell():.....51
public void cleanTank():.....51
public void emptyTank():.....51
public void upgrade():.....51
public void avanzarDias(int dias):.....51
public void agregarPecesAleatorios():.....51
public int getDias():.....51
public void setDias(int dias):.....51
public String getNombreEntidad():.....51
public void setNombreEntidad(String nombreEntidad):.....52
public List<Piscifactoria> getpiscifactorias():.....52
public void setpiscifactorias(ArrayList<Piscifactoria> piscifactorias):.....52
public SistemaMonedas getMonedas():.....52
public void setMonedas(SistemaMonedas monedas):.....52
public InputHelper getInputHelper():.....52
public void setInputHelper(InputHelper inputHelper):.....52

1. Clase AlmacenCentral

1.1 Descripción

La clase almacenCentral representa un almacén central diseñado para gestionar la alimentación de la piscifactoría, permitiendo almacenar tanto comida animal como vegetal. Esta clase proporciona métodos para construir el almacén, aumentar su capacidad, añadir diferentes tipos de comida y calcular costos, además de mostrar el estado actual del almacén.

1.2 Funcionalidades

1. **Inicialización:** Se inicializan los atributos con valores predeterminados al crear una instancia de AlmacenCentral, asegurando que el almacén esté listo para su uso.

2. **Construcción del Almacén**: Se permite construir el almacén una única vez mediante el método construir. Si se intenta construir nuevamente, se informa que el almacén ya está construido.

3. **Aumento de Capacidad**: Se permite aumentar la capacidad del almacén mediante el método aumentarCapacidad, validando que el incremento sea positivo para garantizar la integridad de los datos.

4. **Gestión de Comida**: Los métodos añadirComidaAnimal y añadirComidaVegetal permiten añadir cantidades específicas de comida, validando que la cantidad a añadir sea positiva y que no exceda la capacidad máxima del almacén.

5. **Cálculo de Costos**: Se permite calcular el costo total de añadir una cierta cantidad de comida mediante el método calcularCosto, aplicando descuentos especiales por cada 25 unidades añadidas.

6. **Visualización del Estado:** Se permite ver el estado actual del almacén mediante el método mostrarEstado, mostrando tanto la capacidad total como el porcentaje de espacio ocupado por cada tipo de comida.

1.3 Constructor

- **public AlmacenCentral()**: Se define el constructor de la clase AlmacenCentral. Se inicializan los siguientes valores predeterminados:

    - capacidadAlmacen se establece en 200.
    - cantidadComidaAnimal se inicializa en 0.
    - cantidadComidaVegetal se inicializa en 0.
    - construido se establece en false, indicando que el almacén aún no ha sido construido.

1.4 Atributos

- **capacidadAlmacen**: (Tipo: int)

    Se define la capacidad total máxima del almacén en unidades. Se establece inicialmente en 200. Esta capacidad puede ser aumentada mediante el método aumentarCapacidad.

- **cantidadComidaAnimal**: (Tipo: int)

    Se define la cantidad actual de comida animal almacenada en el almacén. Se inicializa en 0 y se actualiza al añadir comida animal.

- **cantidadComidaVegetal**: (Tipo: int)

    Se define la cantidad actual de comida vegetal almacenada en el almacén. Se inicializa en 0 y se actualiza al añadir comida vegetal.

- **construido**: (Tipo: boolean)

    - Se define si el almacén ha sido construido o no. Se inicializa en false y se establece en true una vez que se invoca el método construir con éxito.

1.5 Métodos

public boolean construir():

Se intenta construir el almacén. Si el almacén no ha sido construido previamente, se marca como construido y se retorna true. Si ya está construido, se informa que el almacén ya está construido y se retorna false.

public void aumentarCapacidad(int aumento):

Se aumenta la capacidad del almacén en la cantidad especificada. Se asegura que el incremento sea un valor positivo. Si se realiza con éxito, se imprime un mensaje indicando la nueva capacidad.

public boolean añadirComidaAnimal(int cantidad):

Se añade una cantidad de comida animal al almacén. Se verifica que la cantidad a añadir sea positiva y que no exceda la capacidad total del almacén. Si la cantidad excede, se ajusta la cantidad almacenada al límite de capacidad y se retorna true si se añadió alguna comida.

public boolean añadirComidaVegetal(int cantidad):

Se añade una cantidad de comida vegetal al almacén. Se asegura que la cantidad a añadir sea positiva y que no exceda la capacidad total del almacén. Si la cantidad excede, se ajusta la cantidad almacenada al límite de capacidad y se retorna true si se añadió alguna comida.

public int calcularCosto(int cantidad):

Se calcula el costo total asociado a la adición de una cierta cantidad de comida, aplicando un descuento por cada 25 unidades. Se retorna el costo final a pagar.

public void mostrarEstado():

Se imprime en consola el estado actual del almacén, mostrando la capacidad total y la cantidad actual de comida animal y vegetal, así como el porcentaje de espacio ocupado por cada tipo de comida.

public int getCapacidadAlmacen():

Se retorna la capacidad total del almacén, permitiendo acceder a esta información.

public void setCapacidadAlmacen(int capacidadAlmacen):

Se establece la capacidad del almacén. Se incluyen validaciones para asegurar que la nueva capacidad no supere la capacidad original.

public int getCantidadComidaAnimal():

Se retorna la cantidad de comida animal actualmente almacenada.

public void setCantidadComidaAnimal(int cantidadComidaAnimal):

Se establece la cantidad de comida animal en el almacén, asegurando que no exceda la capacidad total.

public int getCantidadComidaVegetal():

Se retorna la cantidad de comida vegetal almacenada.

public void setCantidadComidaVegetal(int cantidadComidaVegetal):

Se establece la cantidad de comida vegetal en el almacén, asegurando que no exceda la capacidad total.

public boolean isConstruido():

Se retorna un valor booleano que indica si el almacén ha sido construido.

2. Clase SistemaMonedas

2.1 Descripción

La clase SistemaMonedas representa un sistema de gestión de monedas para llevar el control de las monedas disponibles en la piscifactoría. Permite incrementar y gastar monedas, proporcionando un saldo actual y asegurando la validez de las operaciones.

2.2 Funcionalidades

1. Se permite incrementar la cantidad de monedas mediante el método ganarMonedas, validando que la cantidad a añadir sea mayor que cero.

2. Se permite gastar monedas mediante el método gastarMonedas, asegurando que el costo sea mayor que cero y menor o igual al saldo disponible.

3. Se permite obtener el saldo actual de monedas mediante el método getMonedas, proporcionando la cantidad de monedas disponibles en el sistema.

2.3 Constructor

- **public SistemaMonedas(int saldoInicial):** Se define el constructor que inicializa el saldo de monedas. Se asegura que el saldo inicial sea un valor no negativo. Si se proporciona un valor negativo, se lanza una excepción IllegalArgumentException.

2.4 Atributos

- **monedas: (Tipo: int)** Se define la cantidad de monedas disponibles en el sistema. Se inicializa con el saldo proporcionado al constructor.

2.5 Métodos

public int getMonedas():

Se retorna la cantidad de monedas actual disponible en el sistema.

public boolean ganarMonedas(int cantidad):

Se incrementa la cantidad de monedas en el sistema. Se verifica que la cantidad a ganar sea mayor que cero. Si se añade con éxito, se retorna true y se imprime el saldo actual. Si la cantidad es inválida, se retorna false.

public boolean gastarMonedas(int costo):

Se gasta una cantidad específica de monedas. Se asegura que el costo sea mayor que cero y menor o igual al saldo disponible. Si se gasta con éxito, se retorna true y se imprime el saldo restante. Si no hay suficientes monedas o el costo es inválido, se retorna false.

3. Clase InputHelper

3.1 Descripción

La clase InputHelper facilita la lectura validada de cadenas, números enteros y decimales desde la entrada del usuario. Se asegura de que la entrada no esté vacía y cumpla con el formato esperado, proporcionando mensajes de error claros en caso de entradas inválidas.

3.2 Funcionalidades

1. Se permite la lectura de cadenas, asegurando que no estén vacías y no contengan caracteres especiales.

2. Se permite la lectura de números enteros, validando que la entrada no esté vacía y sea un número válido.

3. Se permite la lectura de números decimales, verificando que la entrada no esté vacía y que sea un valor decimal válido.

4. Se proporciona un método para cerrar el objeto Scanner y liberar los recursos asociados.

3.3 Constructor

- **public InputHelper():** Se inicializa el objeto Scanner para la lectura de datos desde la entrada estándar.

3.4 Atributos

- **scanner: (Tipo: Scanner)** Se define el objeto Scanner utilizado para la lectura de entradas del usuario.

3.5 Métodos

public String readString(String prompt):

Se lee una cadena que no esté vacía y que no contenga caracteres especiales. Se muestra un mensaje de solicitud al usuario y se valida la entrada, retornando la cadena ingresada si es válida.

public int readInt(String prompt):

Se lee un número entero que no esté vacío y sea válido. Se muestra un mensaje de solicitud al usuario y se valida la entrada, retornando el número entero ingresado si es válido.

public double readDouble(String prompt):

Se lee un número decimal que no esté vacío y sea válido. Se muestra un mensaje de solicitud al usuario y se valida la entrada, retornando el número decimal ingresado si es válido.

public void close():

Se cierra el objeto Scanner y libera los recursos asociados, asegurando que no haya fugas de recursos.

4. Clase MenuHelper

4.1 Descripción

En la clase MenuHelper se gestiona un conjunto de opciones de menú, permitiendo agregar, mostrar y ejecutar opciones de manera eficiente. Se utiliza para presentar al usuario una interfaz de menú simple y manejable.

4.2 Funcionalidades

1. Se permite agregar opciones al menú con un número de opción y una descripción asociada.

2. Se permite mostrar todas las opciones del menú, excluyendo la opción de salida (opción 0) de la lista principal.

3. Se proporciona un método para limpiar todas las opciones del menú.

4. Se permite ejecutar una acción basada en la opción seleccionada por el usuario.

4.3 Constructor

- **public MenuHelper()**: Se inicializa el mapa menuOptions que contendrá las opciones del menú.

4.4 Atributos

- **menuOptions: (Tipo: Map<Integer, String>)** : Se define un mapa que almacena las opciones del menú, donde la clave es el número de opción y el valor es la descripción de la opción.

4.5 Métodos

public void addOption(int optionNumber, String description):

Se agrega una opción al menú, asociando un número de opción con su descripción correspondiente.

public void showMenu():

Se muestra el menú, listando todas las opciones disponibles excepto la opción 0, que se muestra al final.

public void clearOptions():

Se limpia el mapa de opciones del menú, eliminando todas las opciones existentes.

public void executeOption(int optionNumber):

Se ejecuta la acción asociada a la opción seleccionada, mostrando un mensaje que indica cuál opción está siendo ejecutada.

5. Interfaz Activo

5.1 Descripción

En la interfaz Activo se define un tipo de pez que es tanto un activo como carnívoro. Al extender de la interfaz Carnivoro, establece que cualquier implementación de Activo debe cumplir con las características y comportamientos de un pez carnívoro.

5.2 Funcionalidades

1. Se establece que cualquier clase que implemente esta interfaz debe ser un pez carnívoro y debe incluir las funcionalidades específicas asociadas a la clase Carnivoro.

6. Interfaz Carnivoro

6.1 Descripción

La interfaz Carnivoro indica que una clase que la implemente debe representar un pez que tiene una dieta carnívora. Esta interfaz establece un contrato para las clases que quieren ser reconocidas como peces carnívoros, aunque no define ningún método específico.

6.2 Funcionalidades

1. Se establece que cualquier clase que implemente esta interfaz debe cumplir con las características asociadas a los peces carnívoros.

7. Interfaz Filtrador

7.1 Descripción

La interfaz Filtrador indica que una clase que la implemente debe representar un pez que tiene una dieta de filtración. Esta interfaz establece un contrato para las clases que quieren ser reconocidas como peces filtradores, aunque no define ningún método específico.

7.2 Funcionalidades

1. Se establece que cualquier clase que implemente esta interfaz debe cumplir con las características asociadas a los peces filtradores.

8. Clase SalmonAtlantico

8.1 Descripción

La clase SalmonAtlantico representa un pez de la especie salmón atlántico, que es carnívoro. Esta clase extiende de la clase Pez y permite la creación de objetos que representan ejemplares de salmón atlántico con características específicas, como su sexo.

8.2 Funcionalidades

1. Se establece que un objeto de la clase SalmonAtlantico es un pez carnívoro.

2. Permite inicializar el pez con un sexo específico, ya sea macho o hembra.

8.3 Constructor

SalmonAtlantico(boolean sexo):

Se inicializa un objeto SalmonAtlantico con su sexo específico.
Se utiliza el constructor de la clase padre Pez, pasando el sexo y las propiedades del salmón atlántico.

8.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

8.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

9. Clase TruchaArcoiris

9.1 Descripción

La clase TruchaArcoiris representa un pez de la especie trucha arcoíris, que es carnívoro. Esta clase extiende de la clase Pez y permite la creación de objetos que representan ejemplares de trucha arcoíris con características específicas, como su sexo.

9.2 Funcionalidades

1. Se establece que un objeto de la clase TruchaArcoiris es un pez carnívoro.

2. Permite inicializar el pez con un sexo específico, ya sea macho o hembra.

9.3 Constructor

- **TruchaArcoiris(boolean sexo):**
  Se inicializa un objeto TruchaArcoiris con su sexo específico.
  Se utiliza el constructor de la clase padre Pez, pasando el sexo y las propiedades de la trucha arcoíris.

9.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

9.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

10. Clase ArenqueDelAtlantico

10.1 Descripción

La clase ArenqueDelAtlantico representa un pez de la especie arenque del Atlántico, que es un pez filtrador. Esta clase extiende de la clase Pez y permite la creación de objetos que representan ejemplares de arenque del Atlántico con características específicas, como su sexo.

10.2 Funcionalidades

1. Se establece que un objeto de la clase ArenqueDelAtlantico es un pez filtrador.

2. Permite inicializar el pez con un sexo específico, ya sea macho o hembra.

10.3 Constructor

- **ArenqueDelAtlantico(boolean sexo):**
  Se inicializa un objeto ArenqueDelAtlantico con su sexo específico.
  Se utiliza el constructor de la clase padre Pez, pasando el sexo y las propiedades del arenque del Atlántico.

10.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

10.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Filtrador.

11. Clase Besugo

11.1 Descripción

La clase Besugo representa un pez de la especie besugo, que es un pez carnívoro. Esta clase extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de besugo con características específicas, como su sexo.

11.2 Funcionalidades

1. Se establece que un objeto de la clase Besugo es un pez carnívoro, lo que implica que puede alimentarse de otros peces u organismos acuáticos.

11.3 Constructor

- **Besugo(boolean sexo):**
  Se inicializa un objeto Besugo con su sexo específico, el cual puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades del besugo, que se obtienen de AlmacenPropiedades.BESUGO.

11.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

11.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

12. Clase LenguadoEuropeo

12.1 Descripción

La clase LenguadoEuropeo representa un pez de la especie lenguado europeo, que es un pez carnívoro. Esta clase extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de lenguado con características específicas, como su sexo.

12.2 Funcionalidades

1. Se establece que un objeto de la clase LenguadoEuropeo es un pez carnívoro, lo que implica que puede alimentarse de otros peces u organismos acuáticos.

12.3 Constructor

- **LenguadoEuropeo(boolean sexo):**
  Se inicializa un objeto LenguadoEuropeo con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades del lenguado, que se obtienen de AlmacenPropiedades.LENGUADO_EUROPEO.

12.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

12.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

13. Clase LubinaRayada

13.1 Descripción

La clase LubinaRayada representa un pez de la especie lubina rayada, que es un pez carnívoro. Esta clase extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de lubina con características específicas, como su sexo.

13.2 Funcionalidades

1. Se establece que un objeto de la clase LubinaRayada es un pez carnívoro, lo que implica que puede alimentarse de otros peces u organismos acuáticos

13.3 Constructor

- **LubinaRayada(boolean sexo):**
  Se inicializa un objeto LubinaRayada con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades de la lubina, que se obtienen de AlmacenPropiedades.LUBINA_RAYADA.

13.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

13.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

14. Clase Robalo

14.1 Descripción

En la clase Robalo se representa un pez de la especie robalo, caracterizado por ser carnívoro. Esta clase extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de robalo con características específicas, como su sexo.

14.2 Funcionalidades

1. Se establece que un objeto de la clase Robalo es un pez carnívoro, lo que implica que puede alimentarse de otros peces u organismos acuáticos.

14.3 Constructor

- **Robalo(boolean sexo):**
  Se inicializa un objeto Robalo con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades del robalo, que se obtienen de AlmacenPropiedades.ROBALO.

14.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

14.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

15. Clase CarpaPlateada

15.1 Descripción

En la clase CarpaPlateada se representa un pez de la especie carpa plateada, caracterizado por ser un pez filtrador. Esta clase se extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de carpa plateada con características específicas, como su sexo.

15.2 Funcionalidades

1. Se establece que un objeto de la clase CarpaPlateada es un pez filtrador, lo que implica que se alimenta filtrando partículas del agua.

15.3 Constructor

- **CarpaPlateada(boolean sexo):**
  Se inicializa un objeto CarpaPlateada con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades de la carpa plateada, que se obtienen de AlmacenPropiedades.CARPA_PLATEADA.

15.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

15.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Filtrador.

16. Clase Pejerrey

16.1 Descripción

En la clase Pejerrey se representa un pez de la especie pejerrey, se caracteriza por ser un pez carnívoro. Esta clase se extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de pejerrey con características específicas, como su sexo.

16.2 Funcionalidades

1. Se establece que un objeto de la clase Pejerrey es un pez carnívoro, lo que implica que se alimenta de otros peces o animales pequeños.

16.3 Constructor

- **Pejerrey(boolean sexo):**
  Se inicializa un objeto Pejerrey con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades del pejerrey, que se obtienen de AlmacenPropiedades.PEJERREY.

16.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez

16.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

17. Clase PercaEuropea

17.1 Descripción

La clase PercaEuropea representa un pez de la especie perca europea, que se caracteriza por ser un pez activo. Esta clase extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de perca europea con características específicas, como su sexo.

17.2 Funcionalidades

1. Se establece que un objeto de la clase PercaEuropea es un pez activo, lo que implica que presenta un comportamiento dinámico en su entorno acuático.

17.3 Constructor

- **PercaEuropea(boolean sexo):**
  Se inicializa un objeto PercaEuropea con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades de la perca europea, que se obtienen de AlmacenPropiedades.PERCA_EUROPEA.

17.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez.

17.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Activo.

18. Clase SalmonChinook

18.1 Descripción

La clase SalmonChinook representa un pez de la especie salmón chinook, conocido por ser carnívoro. Esta clase extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de salmón chinook con características específicas, como su sexo.

18.2 Funcionalidades

1. Se establece que un objeto de la clase SalmonChinook es carnívoro, lo que implica que se alimenta principalmente de otros peces o animales acuáticos.

18.3 Constructor

- **SalmonChinook(boolean sexo):**
  Se inicializa un objeto SalmonChinook con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades del salmón chinook, que se obtienen de AlmacenPropiedades.SALMON_CHINOOK.

18.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez.

18.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Carnivoro.

19. Clase TilapiaDelNilo

19.1 Descripción

La clase TilapiaDelNilo representa un pez de la especie tilapia del Nilo, conocido por ser un filtrador. Esta clase extiende de la clase Pez, permitiendo la creación de objetos que representan ejemplares de tilapia del Nilo con características específicas, como su sexo.

19.2 Funcionalidades

1. Se establece que un objeto de la clase TilapiaDelNilo es un filtrador, lo que implica que se alimenta filtrando partículas de alimento del agua.

19.3 Constructor

- **TilapiaDelNilo(boolean sexo):**
  Se inicializa un objeto TilapiaDelNilo con su sexo específico, que puede ser macho (true) o hembra (false).
  Se invoca el constructor de la clase padre Pez, pasando el sexo y las propiedades de la tilapia del Nilo, que se obtienen de AlmacenPropiedades.TILAPIA_NILO.

19.4 Atributos

En esta clase no se definen atributos ya que estos se heredan de la clase Pez.

19.5 Métodos

En esta clase no se definen métodos ya que se heredan de la clase Pez y de la interfaz Filtrador.

20. Clase Pez

20.1 Descripción

La clase Pez es la clase base que representa a un pez en el sistema. Esta clase incluye atributos que describen las características fundamentales de un pez, así como métodos para gestionar su estado, crecimiento y reproducción.

20.2 Funcionalidades

1. **Inicialización:** Se inicializan los atributos con valores predeterminados al crear una instancia de AlmacenCentral, asegurando que el almacén esté listo para su uso.

2. **Construcción del Almacén**: Se permite construir el almacén una única vez

20.3 Constructor

- **Pez(boolean sexo, PecesDatos datos)**: Inicializa un nuevo pez con su sexo y datos asociados.

20.4 Atributos

- **nombre:** Nombre común del pez (final).
- **nombreCientifico:** Nombre científico del pez (final).
- **edad**: Edad del pez en días (inicializada a 0).
- **sexo:** Sexo del pez (true para macho, false para hembra) (final).
- **fertil**: Estado de fertilidad del pez (true si es fértil, false si no lo es).

- **vivo:** Estado de vida del pez (true si está vivo, false si está muerto).
- **alimentado**: Estado de alimentación del pez (true si ha sido alimentado, false si no lo ha sido).
- **ciclo**: Ciclo reproductivo del pez, que decrece hasta que el pez se vuelve fértil.
- **datos**: Objeto PecesDatos que contiene información específica sobre el pez.

20.5 Métodos

public void showStatus()

Se muestra el estado actual del pez, incluyendo edad, sexo, estado de vida, estado de alimentación, madurez y fertilidad.

public void grow()

Se simula el crecimiento del pez a lo largo de un día, afectando su edad, fertilidad y probabilidad de muerte.

public void reset()

Se reinicia el estado del pez a su condición inicial.

public Pez clonar(boolean nuevoSexo)

Se crea una copia del pez con un nuevo sexo, true para macho, false para hembra.

public String getNombre()

Se obtiene el nombre común del pez.

public String getNombreCientifico()

Se obtiene el nombre científico del pez.

public int getEdad()

Se obtiene la edad actual del pez en días.

public boolean isSexo()

Se determina si el pez es macho o hembra.

public boolean isFertil()

Se verifica si el pez es fértil.

public boolean isVivo()

Se comprueba si el pez está vivo o muerto.

public boolean isAlimentado()

Se verifica si el pez ha sido alimentado.

public int getCiclo()

Se obtiene el ciclo reproductivo actual del pez.

public PecesDatos getDatos()

Se obtiene el objeto PecesDatos asociado con este pez.

public void setEdad(int edad)

Se establece la nueva edad del pez.

public void setVivo(boolean vivo)

Se establece el estado de vida del pez, indicando si está vivo o muerto.

public void setAlimentado(boolean alimentado)

Se establece si el pez ha sido alimentado.

public void setCiclo(int ciclo)

Se establece el nuevo ciclo reproductivo del pez.

public void setDatos(PecesDatos datos)

Se establece el objeto PecesDatos asociado con el pez.

public void setFertil(boolean fertil)

Se establece el estado de fertilidad del pez.

21. Clase PiscifactoriaDeMar

21.1 Descripción

La clase PiscifactoriaDeMar extiende la clase Piscifactoria y representa una piscifactoría que se especializa en el cultivo de peces de mar. Esta clase gestiona los tanques de peces, limitando el número de tanques y proporcionando funcionalidad para añadir tanques de capacidad fija.

21.2 Funcionalidades

1. Permite la creación de una piscifactoría marina con un nombre y un sistema de monedas.

2. Incluye métodos para añadir tanques de capacidad fija de 100 unidades y para obtener el número actual de tanques en la piscifactoría.

21.3 Constructor

- **PiscifactoriaDeMar(String nombre, SistemaMonedas monedas):**
  Inicializa la piscifactoría marina con el nombre proporcionado y un sistema de monedas.
  Configura la capacidad máxima de comida de la piscifactoría en 100.
  Inicializa los contadores de comida vegetal y animal en 0.
  Añade un tanque de capacidad 100 y establece el contador de tanques en 1.

21.4 Atributos

- **CAPACIDAD_MAXIMA_TANQUES (Tipo: int):** Define la capacidad máxima del array de tanques, establecido en 10.

- **contadorTanquesMar (Tipo: int)**: Contador que rastrea el número actual de tanques en la piscifactoría.

21.5 Métodos

public boolean añadirTanque()

Añade un nuevo tanque con capacidad fija de 100 unidades a la piscifactoría, si hay espacio disponible.
Retorna true si el tanque se añadió correctamente y false si no hay espacio.

public int getTanquesMar()

Retorna el número actual de tanques en la piscifactoría.

22. Clase PiscifactoriaDeRio

22.1 Descripción

La clase PiscifactoriaDeRio extiende la clase Piscifactoria y representa una piscifactoría especializada en el cultivo de peces de río. Esta clase gestiona los tanques de peces, limitando el número de tanques y proporcionando funcionalidad para añadir tanques de capacidad fija.

22.2 Funcionalidades

1. Permite la creación de una piscifactoría de río con un nombre y un sistema de monedas.

2. Incluye métodos para añadir tanques de capacidad fija de 25 unidades y para obtener el número actual de tanques en la piscifactoría.

22.3 Constructor

PiscifactoriaDeRio(String nombre, SistemaMonedas monedas):

- Inicializa la piscifactoría de río con el nombre proporcionado y un sistema de monedas.
- Configura la capacidad máxima de comida de la piscifactoría en 25.
- Inicializa los contadores de comida vegetal y animal en 0.
- Añade un tanque de capacidad 25 y establece el contador de tanques en 1.

22.4 Atributos

- **CAPACIDAD_MAXIMA_TANQUES (Tipo: int)**: Define la capacidad máxima del array de tanques, establecido en 10.

- **contadorTanquesRio (Tipo: int)**: Contador que rastrea el número actual de tanques en la piscifactoría.

22.5 Métodos

public boolean añadirTanque()

Añade un nuevo tanque con capacidad fija de 25 unidades a la piscifactoría, si hay espacio disponible.
Retorna true si el tanque se añadió correctamente y false si no hay espacio.

public int getTanquesRio()

Retorna el número actual de tanques en la piscifactoría.

23. Clase Piscifactoria

23.1 Descripción

La clase Piscifactoria se define como una clase abstracta encargada de gestionar y coordinar el funcionamiento de una piscifactoría, la cual contiene múltiples tanques de peces y sistemas de alimentación y monedas.

23.2 Funcionalidades

1. **Administración de tanques y peces:** Se permite agregar tanques y gestionar los peces de acuerdo con el tipo de piscifactoría (de río o de mar). Los peces son distribuidos en tanques específicos dependiendo de sus características y necesidades ambientales.

2. **Monitoreo del estado de la piscifactoría**: Se muestra información detallada sobre la ocupación de la piscifactoría, el estado de los peces (vivos, adultos, alimentados, fértiles, y distribución por género) y el estado de cada tanque en particular.

3. **Gestión de alimentos:** Se administran dos tipos de comida (vegetal y animal) para alimentar a los peces. Los peces pueden ser filtradores, carnívoros o activos, y se alimentan según su tipo y estado. Además, se permite mejorar la capacidad del almacén de alimentos de la piscifactoría.

4. **Ciclo de vida de los peces**: Se permite avanzar en el ciclo de vida diario de los peces, incluyendo su alimentación y estado de salud, así como la venta de peces maduros.

5. **Gestión del sistema de monedas:** Se acumulan monedas en función de la venta de peces, permitiendo gestionar el crecimiento y éxito económico de la piscifactoría.

23.3 Constructor

Piscifactoria(String nombre, SistemaMonedas monedas): Se inicializa una nueva piscifactoría con el nombre especificado y el sistema de monedas asignado para la gestión de ingresos.

23.4 Atributos

- **nombre (String)**: Nombre de la piscifactoría.
- **tanques (List<Tanque>)**: Lista de tanques que contiene los peces de la piscifactoría.
- **monedas (SistemaMonedas)**: Sistema de monedas para gestionar las ganancias obtenidas.
- **capacidadMaximaComidaPiscifactoria (int)**: Capacidad máxima compartida para ambos tipos de alimentos.
- **comidaVegetalActual (int)**: Cantidad actual de comida vegetal disponible.
- **comidaAnimalActual (int)**: Cantidad actual de comida animal disponible.
- **numeroTanquesDeRio (int)**: Número de tanques de tipo río en la piscifactoría.
- **numeroTanquesDeMar (int)**: Número de tanques de tipo mar en la piscifactoría.
- **almacenCentral (AlmacenCentral)**: Almacén central que gestiona la reserva de comida.

23.5 Métodos

public void agregarTanque(Tanque tanque):

Se agrega un tanque a la piscifactoría.

public List<Tanque> getTanques():

Se devuelve la lista de tanques de la piscifactoría.

public void addPez(Pez pez):

Se agrega un pez a la piscifactoría, verificando que sea del tipo adecuado.

public void showStatus():

Se muestra toda la información de la piscifactoría, incluyendo estado de los tanques y estadísticas de los peces.

public void showTankStatus():

Se muestra el estado de cada tanque en la piscifactoría.

public void showFishStatus(int numeroTanque):

Se muestra la información de los peces de un tanque determinado.

public void showCapacity(int numeroTanque):

Se muestra la ocupación de un tanque determinado.

public void showFood():

Se muestra el estado del almacén de comida de la piscifactoría.

public void nextDay():

Hace que se avance el ciclo de vida en la piscifactoría, alimentando a los peces y actualizando sus estados.

private void alimentarPeces(Tanque tanque):

Se alimenta a los peces en un tanque específico.

public void sellFish():

Se vende los peces maduros y actualiza el sistema de monedas.

public void upgradeFood(int incremento):

Se mejora el almacén de comida aumentando su capacidad máxima.

public boolean verificarTanqueYPiscifactoria(Tanque tanqueBuscado):

Se verifica si el tanque pertenece a la piscifactoría y determina si esta es de tipo río o de mar.

public int getTotalPeces():

Se devuelve el total de peces en la piscifactoría.

public int getTotalVivos():

Se devuelve el total de peces vivos en la piscifactoría.

public int getTotalAlimentados():

Se devuelve el total de peces alimentados en la piscifactoría.

public int getTotalAdultos():

Se devuelve el total de peces adultos en la piscifactoría.

public int getTotalHembras()

Se devuelve el total de hembras en la piscifactoría.

public int getTotalMachos():

Se devuelve el total de machos en la piscifactoría.

public int getTotalFertiles():

Se devuelve el total de peces fértiles en la piscifactoría.

public int getCapacidadTotal():

Se devuelve la capacidad total de todos los tanques de la piscifactoría.

public String getNombre():

Se devuelve el nombre de la piscifactoría.

public void setNombre(String nombre):

Se establece el nombre de la piscifactoría.

public void setNumeroTanquesDeRio(int numeroTanquesDeRio):

Establece el número de tanques de río en la piscifactoría.

public void setNumeroTanquesDeMar(int numeroTanquesDeMar):

Se establece el número de tanques de mar en la piscifactoría.

public int getNumeroTanquesDeRio():

Se obtiene el número de tanques de agua dulce (río) que se encuentran en la piscifactoría.

public int getNumeroTanquesDeMar():

Se obtiene el número de tanques de agua salada (mar) disponibles en la piscifactoría.

public int getComidaVegetalActual():

Se obtiene la cantidad actual de comida vegetal almacenada en la piscifactoría.

public void setComidaVegetalActual(int comidaVegetalActual):

Se establece una nueva cantidad de comida vegetal en la piscifactoría, según el valor proporcionado.

public void setComidaAnimalActual(int comidaAnimalActual):

Se establece una nueva cantidad de comida animal en la piscifactoría, según el valor proporcionado.

public int getComidaAnimalActual():

Se obtiene la cantidad actual de comida animal en la piscifactoría.

24. Clase Tanque

24.1 Descripción

En la clase Tanque se representa un tanque de peces en el cual se pueden almacenar y gestionar ejemplares de diferentes especies. Esta clase proporciona métodos para añadir peces, mostrar el estado del tanque y realizar operaciones de crecimiento y reproducción de los peces.

24.2 Funcionalidades

1. **Gestión de Peces**: Se permite agregar peces al tanque, mostrando información relevante sobre su estado.

2. **Reproducción**: Se implementa la lógica para la reproducción de los peces presentes en el tanque.

3. **Crecimiento**: Se realiza el crecimiento de los peces con el paso del tiempo

24.3 Constructor

Tanque(int capacidadMaxima):

- Se inicializa un nuevo tanque con la capacidad máxima especificada y una lista vacía de peces.

24.4 Atributos

- **peces: ArrayList<Pez>** Lista que almacena los peces presentes en el tanque.

- **tipoPezActual: Class<?>** Tipo de pez actualmente permitido en el tanque.
- **capacidadMaxima: int** Capacidad máxima del tanque.
- **numeroTanque: int** Número identificador del tanque.

24.5 Métodos

public void showStatus():

Se muestra el estado actual del tanque, incluyendo ocupación, peces vivos, alimentados, adultos, distribución de sexos y fértiles.

public void showFishStatus():

Se muestra el estado de todos los peces en el tanque.

public void showCapacity():

Se muestra la capacidad actual del tanque.

public void nextDay():

Se avanza un día en el tanque, haciendo crecer los peces y ejecutando la reproducción.

void reproduccion():

Se maneja la lógica de reproducción de los peces en el tanque.

public boolean addPez(Pez pez):

Se agrega un pez al tanque, validando la capacidad y el tipo de pez.

public Class<?> getTipoPezActual():

Se devuelve el tipo de pez actual permitido en el tanque.

public int getNumeroTanque():

Se devuelve el número del tanque.

public int getCapacidad():

Se devuelve la capacidad máxima del tanque.

public int getNumPeces():

Se devuelve el número de peces en el tanque.

public ArrayList<Pez> getPeces():

Se devuelve la lista de peces en el tanque.

public int getHembras():

Se cuenta y devuelve el número de hembras en el tanque.

public int getMachos():

Se cuenta y devuelve el número de machos en el tanque.

public int getFertiles():

Se cuenta y devuelve el número de peces fértiles en el tanque.

public int getVivos():

Se cuenta y devuelve el número de peces vivos en el tanque.

public int getAlimentados():

Se cuenta y devuelve el número de peces alimentados en el tanque.

public int getAdultos():

Se cuenta y devuelve el número de peces adultos en el tanque.

25. Clase Simulador

25.1 Descripción

En la clase Tanque se representa un tanque de peces en el cual se pueden almacenar y gestionar ejemplares de diferentes especies. Esta clase proporciona métodos para añadir peces, mostrar el estado del tanque y realizar operaciones de crecimiento y reproducción de los peces.

25.2 Funcionalidades

1. Se inicializa la simulación y se configuran los elementos necesarios.
2. Se muestra el estado general de la piscifactoría y de los tanques de peces.
3. Se permite añadir comida y peces a los tanques.
4. Se realizan ventas de peces y se gestionan las monedas.
5. Se permite avanzar en el tiempo de la simulación, así como agregar peces aleatorios de manera gratuita.
6. Se proporciona una interfaz para limpiar y vaciar los tanques.
7. Se realizan mejoras en la piscifactoría.

25.3 Constructor

public Simulador()

- Se crea una instancia de la clase Simulador. Se inicializan los atributos necesarios para el funcionamiento de la simulación.

25.4 Atributos

- **private int dias:** Se almacena la cantidad de días transcurridos en la simulación.
- **private String nombreEntidad**: Se guarda el nombre de la entidad que representa la piscifactoría.
- **private List<Piscifactoria> piscifactorias**: Se gestiona la lista de piscifactorías disponibles en la simulación.
- **private SistemaMonedas monedas**: Se gestiona el sistema de monedas para realizar transacciones en la simulación.
- **private InputHelper inputHelper:** Se utiliza para facilitar la entrada de datos por parte del usuario.

25.5 Métodos

public void init():

Se inicializan los componentes necesarios para la simulación.

public void menu():

Se muestra el menú principal de opciones disponibles para el usuario.

public void showGeneralStatus():

Se presenta el estado general de la simulación.

public void showSpecificStatus():

Se muestra el estado específico de las piscifactorías.

public void showTankStatus():

Se proporciona información sobre el estado de los tanques.

public void showStats():

Se presentan estadísticas relevantes de la simulación.

public void showIctio():

Se despliega la ictiopedia, con información sobre los peces.

public void nextDay():

Se avanza un día en la simulación, actualizando el estado de los elementos.

public void addFood():

Se añade comida a la piscifactoría seleccionada.

public void addFish():

Se permite añadir peces a la piscifactoría seleccionada.

public void sell():

Se realiza la venta de peces y se gestionan las monedas.

public void cleanTank():

Se limpia el tanque seleccionado.

public void emptyTank():

Se vacía el tanque seleccionado.

public void upgrade():

Se mejoran las capacidades de la piscifactoría.

public void avanzarDias(int dias):

Se avanza la simulación una cantidad específica de días.

public void agregarPecesAleatorios():

Se añaden cuatro peces seleccionados al azar a una piscifactoría elegida.

public int getDias():

Se obtiene la cantidad de días transcurridos en la simulación.

public void setDias(int dias):

Se establece la cantidad de días transcurridos en la simulación.

public String getNombreEntidad():

Se obtiene el nombre de la entidad de la piscifactoría.

public void setNombreEntidad(String nombreEntidad):

Se establece el nombre de la entidad de la piscifactoría.

public List<Piscifactoria> getpiscifactorias():

Se obtiene la lista de piscifactorías disponibles.

public void setpiscifactorias(ArrayList<Piscifactoria> piscifactorias):

Se establece la lista de piscifactorías disponibles.

public SistemaMonedas getMonedas():

Se obtiene el sistema de monedas de la simulación.

public void setMonedas(SistemaMonedas monedas):

Se establece el sistema de monedas de la simulación.

public InputHelper getInputHelper():

Se obtiene el helper para la entrada de datos.

public void setInputHelper(InputHelper inputHelper):

Se establece el helper para la entrada de datos.