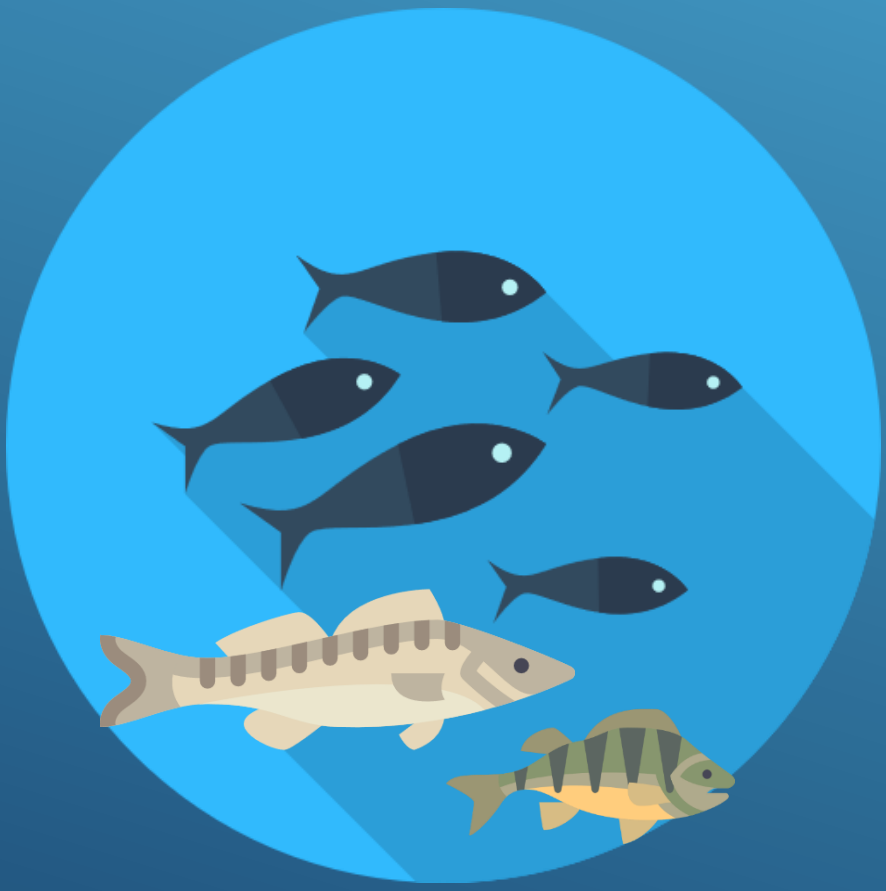
Adrián Fernández González



TRABAJO PISCIFACTORÍA

Entrega II. Acceso a ficheros

Índice

Trabajo II. Acceso a ficheros

Introducción

Esta es la primera ampliación del trabajo, añadiendo funcionalidades de acceso a ficheros básicos, XML y JSON.

El proyecto ha de guardarse a buen recaudo, pues se usará la entrega anterior para crear la siguiente.

Es importante **corregir** los fallos detectados en cada entrega para que las siguientes funcionen correctamente.

Todas las entregas se realizarán mediante un **zip o rar** con los archivos del sistema y los documentos a modo de manual.

Debe **respetarse el nombre** de todo lo estipulado, tanto clases como paquetes.

Lee detenidamente este documento antes de empezar. Crear un diagrama en papel ayuda a visualizar las cosas.

Revisa los **archivos adjuntos**, analízalos y utilízalos como guía de lo estipulado en el documento.

Algunos datos se describen superficialmente, ya que su implementación puede realizarse de diversas formas y queda a tu disposición escoger la forma más oportuna. Ten en cuenta que, durante la corrección, has de justificar tu elección de forma razonable.

Ten en cuenta que, aunque se describa un proceso en una sección determinada, su implementación puede ser realizada en otro lugar.

Código identificador personal

Para el uso de ciertas utilidades de esta entrega y posteriores, cada uno recibirá un código alfanumérico que deberá almacenar en su sistema. Este es **invariable**, **permanente** y **común** para todas las ejecuciones.

Se utilizará para cuestiones de **transferencia** de información entre sistemas y para ciertas tareas de **evaluación**, por lo que **no debes usar el de otro**.

Modificaciones

Primero de todo, para equilibrar el sistema, se van a realizar una serie de modificaciones en ciertas funcionalidades.

Probabilidad de muerte

Ahora, todos los peces tienen un 5% de morir cada **día par** antes de su madurez (incluido).

Vender peces

La venta manual de peces ahora se realizará de una única piscifactoría seleccionada por el usuario, no de todas. Además, darán la mitad del dinero de lo normal. Los longevos darán la mitad de lo que iban a dar. El redondeo se realiza hacia abajo.

InputHelper

Crea una clase que se encargue de la entrada de datos por el usuario, verificando que introduce el tipo adecuado, en el rango establecido si fuese el caso y pidiendo que lo vuelva a introducir si no cumple los parámetros.

Incluye tantos métodos como consideres necesarios y úsalos en todo el sistema cuando requieras que el usuario introduzca cualquier dato.

Sistema de registros

Se van a registrar las acciones del usuario de dos formas distintas, mediante una *transcripción* detallada de la partida y mediante un *log* pormenorizado de las acciones realizadas.

Para ello, crea una o más clases que añadan al final del fichero correspondiente dicha información.

Las *transcripciones* se guardan en **nombre_partida.tr** en la carpeta *transcripciones*, mientras que el *log* en **nombre_partida.log** en la carpeta *logs*.

Ambos ficheros y directorios se crean en el método *init* al iniciar una partida, siendo *nombre_partida* el nombre dado por el usuario a esa partida.

La clase o clases han de ser un *Singleton* y los *streams* han de ser abiertos cuando se crea la instancia y cerrados al terminar el sistema, ya sea por cierre del usuario o una excepción no controlada que rompa el sistema. Esto es así por ser unos archivos de acceso continuo a escribir en cada operación del sistema. Simplemente recuerda limpiar el buffer tras cada escritura y gestionar correctamente las excepciones que rompan el sistema.

Sistema de transcripción

A continuación, se indican las acciones ya implementadas, aquellas nuevas serán especificadas en sus respectivas secciones.

Inicio

Cuando se inicia la partida, se añade una serie de información inicial a modo de informe.

Arranque, línea indicando que empieza el juego con el nombre del sistema/simulación/empresa.

Inicio de la simulación nombre_partida.

Dinero inicial, *Dinero: X monedas*.

Peces implementados con su nombre y ordenados por el tipo de piscifactoría.

Río:
-Pez1
-Pez2
Mar:
…

Extras implementados, indicando el de cada uno nombre, tal cual el documento, en una línea cada uno. Solo aparecerá esta sección si se implementa alguno.

Cada una de estas secciones se separará con una especie de cabecera generada a base de "=".

========= Extras =========

En este caso, sería el de los extras. El número de iguales y la longitud de la barra va a consideración tuya.

Tras todo este texto inicial, se añade una línea de "-" para separar la información de inicio del resto.

Por último, como **primera entrada**, una línea con el nombre de la primera piscifactoría.

Piscifactoría inicial: nombre_piscifactoría.

Comprar comida

Se indicará la cantidad de comida comprada, cuantas monedas se gastaron y dónde se guardó, ya sea una piscifactoría o el almacén central.

X de comida comprada por Y monedas. Se almacena en la piscifactoría Z.

X de comida comprada por Y monedas. Se almacena en el almacén central.

Comprar peces

Se indicará el nombre del pez, su sexo, el tanque al que va y el dinero gastado.

X (M/H) comprado por Y monedas. Añadido al tanque Y de la piscifactoría Z.

Vender peces

Cuando se vendan peces de forma manual, se registrará la venta en conjunto junto con la cantidad ganada.

Vendidos X peces de la piscifactoría Y de forma manual por Z monedas.

Limpiar tanque

Cuando se limpie un tanque, se especificará dicho tanque.

Limpiado el tanque X de la piscifactoría Y.

Vaciar tanque

Cuando se vacíe un tanque, se especificará dicho tanque.

Vaciado el tanque X de la piscifactoría Y.

Comprar edificio

Cada vez que se compre un elemento, se especificará dicho elemento comprado, su nombre en el caso de las piscifactorías y las monedas gastadas.

Comparada la piscifactoría de río/mar X por Y monedas.

Comprado un tanque número X de la piscifactoría Y.

Comprado el almacén central.

Mejorar edificio

En el caso de las mejoras, se especifica qué se ha mejorado, dónde y por cuánto, así como la información relevante de la misma.

Mejorada la piscifactoría X aumentando su capacidad de comida hasta un total de Y por Z monedas.

Pasar día

Al pasar el día, se indican tres cosas, el número de día que termina, el número de peces del sistema divididos en río y mar y luego el número de monedas ganadas por la venta automática y las totales que quedan en el sistema.

Fin del día X.
Peces actuales, X de río Y de mar.
X monedas ganadas por un total de Y.

Por último, se añade una línea divisoria creada con "-" y se indica el día que comienza precedido por tres ">".

>>>Inicio del día X.

Opciones ocultas

Al utilizar alguna de las opciones ocultas del sistema, la de añadir peces y la de añadir monedas, se registran indicándolo.

Añadidos peces mediante la opción oculta a la piscifactoría X.

Añadidas 1000 monedas mediante la opción oculta. Monedas actuales, X.

Sistema de log

Como se expuso anteriormente, este sistema es similar, pero incluyendo fechas y menos información. El formato de fecha es *[aaaa-mm-dd hh:mm:ss]* corchetes incluidos.

A continuación, se indican las acciones ya implementadas, aquellas nuevas serán especificadas en sus respectivas secciones.

Inicio
Cuando se inicia la partida, se añade una línea indicando que empieza la partida con la fecha y hora actual y el nombre de la misma.

[aaaa-mm-dd hh:mm:ss] Inicio de la simulación nombre_partida.

Tras esto, como **primera entrada**, una línea con la fecha y hora actual y el nombre de la primera piscifactoría.

[aaaa-mm-dd hh:mm:ss] Piscifactoría inicial: nombre_piscifactoría.

Comprar comida
Se indicará la cantidad de comida comprada, cuantas monedas se gastaron y dónde se guardó, ya sea una piscifactoría o el almacén central.

[aaaa-mm-dd hh:mm:ss] X de comida comprada. Se almacena en la piscifactoría Z.

[aaaa-mm-dd hh:mm:ss] X de comida comprada. Se almacena en el almacén central.

Comprar peces
Se indicará el nombre del pez, su sexo y el tanque al que va.

[aaaa-mm-dd hh:mm:ss] X (M/H) comprado. Añadido al tanque Y de la piscifactoría Z.

Vender peces
Cuando se vendan peces de forma manual, se registrará la venta en conjunto junto con la cantidad ganada.

[aaaa-mm-dd hh:mm:ss] Vendidos X peces de la piscifactoría Y de forma manual.

Limpiar tanque
Cuando se limpie un tanque, se especificará dicho tanque.

[aaaa-mm-dd hh:mm:ss] Limpiado el tanque X de la piscifactoría Y.

Vaciar tanque
Cuando se vacíe un tanque, se especificará dicho tanque.

[aaaa-mm-dd hh:mm:ss] Vaciado el tanque X de la piscifactoría Y.

Comprar edificio
Cada vez que se compre un elemento, se especificará dicho elemento comprado, su nombre en el caso de las piscifactorías.

[aaaa-mm-dd hh:mm:ss] Comparada la piscifactoría de río/mar X.

[aaaa-mm-dd hh:mm:ss] Comprado un tanque para la piscifactoría X.

[aaaa-mm-dd hh:mm:ss] Comprado el almacén central.

Mejorar edificio
En el caso de las mejoras, se especifica qué se ha mejorado y dónde.

[aaaa-mm-dd hh:mm:ss] Mejorada la piscifactoría X aumentando su capacidad de comida.

Pasar día

Al pasar el día, se indica con una línea.

[aaaa-mm-dd hh:mm:ss] Fin del día X.

Opciones ocultas

Al utilizar alguna de las opciones ocultas del sistema, la de añadir peces y la de añadir monedas, se registran en el log.

[aaaa-mm-dd hh:mm:ss] Añadidos peces mediante la opción oculta a la piscifactoría X.

[aaaa-mm-dd hh:mm:ss] Añadidas monedas mediante la opción oculta.

Salir

Cuando se sale de la partida, se registra la hora en la que se hizo.

[aaaa-mm-dd hh:mm:ss] Cierre de la partida.

Registro de errores

Se registrarán los errores del sistema de forma general con un archivo compartido para todos los sistemas denominado **0_errors.log** del directorio *logs*. Este se crea en el método *init* si no existe ya.

En él se guardan los mensajes de error del sistema cada vez que ocurre una excepción. En cada línea se incluye la fecha y un mensaje acorde.

[aaaa-mm-dd hh:mm:ss] Error de escritura del fichero X.

Todas ellas **dejan de ser mostradas por pantalla** para ser registradas, tal como se haría en un sistema real.

Las únicas excepciones que no se registran son las del input de datos por parte del usuario, que pueden ser utilizadas para realizar la lógica de repetición de las peticiones de dichos datos.

Para asegurarse de que todo es recogido, se añadirá un try-catch en el *main* recogiendo *Exception*. Estas, a mayores de registrarlas, pueden ser mostradas por pantalla para cuestiones de pruebas y resolución de las mismas, ya que son, a priori, fallos no detectados o problemas ajenos al funcionamiento normal.

En este caso, como es habitual, los *streams* serán de un solo uso.

Sistema de recompensas

A partir de ahora, la partida puede generar y obtener ciertas recompensas como comida gratis, tanques, edificios, etc.

Estas recompensas se representan mediante archivos XML en el directorio *rewards*. Cada una tiene un nombre específico, pero comparten una estructura.

```
<reward>
    <name>Nombre de la recompensa</name>
    <origin>Código del usuario</origin>
    <desc>Descripción</desc>
    <rarity>0-4</rarity>
    <give>
        <!--
            <food type="general/algae/animal">Cantidad de comida</food>
            <coins>Monedas</coins>

            <building code="código de la construcción">Nombre construcción</building>
            <part>Fragmento de la construcción</part>
            <total>Fragmentos necesarios</total>
        -->
    </give>
    <quantity>Cantidad disponible</quantity>
</reward>
```

Hay tres tipos básicos de recompensa, de comida, de monedas y de materiales de edificio. Las dos primeras otorgan la recompensa inmediatamente, las de edificio, tienen un número de partes a conseguir y se necesitan todas para poder canjearlas.

El directorio puede almacenar todos los tipos disponibles, pero solo uno de cada. Cada repetición, se modificará en la respectiva etiqueta y, si llega a cero, se eliminará el archivo.

Creación

Crea una clase que genere los distintos archivos de recompensas basándote en los XML facilitados en **rewards.rar**. Esta ha de disponer un método para cada tipo, admitiendo el nivel o letra como parámetro. El nombre de las recompensas es el mismo que se estipula en los archivos facilitados.

Ten en cuenta que, si el archivo ya existe, has de aumentar la cantidad en lugar de crearlo.

Uso

Las recompensas son compartidas por todas las partidas, es decir, pueden ser creadas en una y utilizadas en otra.

Para ello, crea una nueva opción en el menú debajo de la opción de mejorar que permita seleccionar la bonificación a canjear. Solo han de aparecer aquellas de las que se dispone, existe el fichero, y ha de mostrarse que otorga.

En caso de los materiales de los edificios, solo se mostrará una opción, pero indicando las partes de las que se dispone. Por ejemplo, si de la recompensa de almacén central se dispone de A, C y D, se mostrará AxCD. En caso de que el usuario seleccione la opción, se indicará que no dispone de todos los materiales y volverá a preguntarle una opción.

Ten en cuenta que, a la hora de utilizarlas, si hay más de una (*quantity* es mayor que 1), se reduce en uno dicha cantidad, en caso contrario, se elimina el archivo.

Las recompensas de comida van al almacén central y, si este no está disponible, se distribuirá entre todas las piscifactorías como haría este. En caso de que haber espacio para toda la comida, la sobrante se pierde.

Registro

Cuando se efectúa una acción con alguna de estas recompensas, ha de registrarse.

Log

En el log se registrará cuando una de estas recompensas se crea por el sistema, se recibe (externamente o por algún método de pruebas o generación) o se usa.

[aaaa-mm-dd hh:mm:ss] Recompensa creada.

[aaaa-mm-dd hh:mm:ss] Recompensa recibida por X.

[aaaa-mm-dd hh:mm:ss] Recompensa X usada.

Transcripción

La transcripción registrará cuando se crea o se usa una recompensa.

Recompensa X creada.

Recompensa X usada.

Sistema de guardado

Añade un sistema de guardado mediante el uso de JSON. Para ello, has de volcar toda la información necesaria para luego ser recuperada.

Has de respetar exactamente los nombres y tipos.

- En *implementados* se estipula el nombre de cada pez implementado en el sistema, **en el orden exacto en el que se añaden a la librería Orca**.
- *Empresa* es el nombre del sistema/empresa tal como se le dio al iniciarlo.
- *Dia* es el día actual.
- *Monedas* es la cantidad de monedas actual.
- *Orca* es el código de guardado que genera la librería Orca.
- *Edificios* el conjunto de edificios de sistema
    - *Almacen* guarda la información del almacén central.
        - *Disponible* si se dispone del almacén central.
        - *Capacidad* la capacidad máxima del almacén central.
        - *Comida* un objeto que representa la cantidad de comida almacenada en el almacén central.
            - *General* la cantidad de comida multipropósito almacenada.
- *Piscifactorias* es el conjunto de piscifactorías y sus datos.
    - *Nombre* es el nombre de la piscifactoría.
    - *Tipo* es el tipo de piscifactoría, 0 para río, 1 para mar.

- *Capacidad* es la capacidad de comida máxima de la piscifactoría.
- *Comida* un objeto que representa la cantidad de comida que dispone actualmente.
  - *General* la cantidad de comida multipropósito almacenada.
- *Tanques* es el conjunto de tanques y sus datos.
  - *Pez* el nombre del pez que hay dentro. “” en caso de que no tenga.
  - *Num* el número de peces que hay en su interior.
  - *Datos* los datos informativos del tanque.
    - *Vivos* el número de peces vivos del tanque.
    - *Maduros* el número de peces maduros del tanque.
    - *Fertiles* el número de peces fértiles del tanque.
  - *Peces* los datos de los peces del interior del tanque.
    - *Edad* la edad del pez en concreto.
    - *Sexo* el sexo del pez. True hembra, false macho.
    - *Vivo* si el pez está vivo.
    - *Maduro* si el pez es adulto.
    - *Fertil* si el pez es fértil.
    - *Ciclo* el número de días hasta volver a ser fértil.
    - *Extra* es el conjunto de elementos adicionales que necesites para el correcto funcionamiento de los peces según tu implementación.

Puedes ver un ejemplo de la estructura en el archivo ***Estructura guardado.json***.

Menú de inicio

Cuando se inicie el sistema, se buscará si existe el directorio *saves*, si no existe se crea. Una vez realizado esto, puede haber dos opciones:

Si este no existía o está vacío, se creará una partida como hasta ahora.

Si existe, se le ofrece al usuario la posibilidad de cargar una de las partidas o de crear una nueva.

Guardado

Hay tres momentos en los que se guarda la partida, cuando se crea la partida por primera vez, guardado el estado inicial, al pasar un día o cuando se sale del sistema por orden del usuario.

Para ello, se genera y se escribe la información antes expuesta en un archivo **nombre_sistema.save** en la carpeta *saves*, siendo *nombre_sistema* el nombre del sistema/partida/empresa.

También tendrás que generar los datos de Orca, para ello utiliza el método ***exportarDatos*** de la clase *estadisticas.Estadisticas*.

Por último, has de registrar en el *log* que se ha guardado el sistema *[aaaa-mm-dd hh:mm:ss] Sistema guardado.*

Carga

La carga se realiza en el método *init*, antes de preguntar nada, buscará un archivo de guardado. Si lo encuentra, mostrará un mensaje de “Cargando partida”, realizará todas las operaciones necesarias llamando a un método ***load*** y luego mostrará un mensaje de “Partida cargada” iniciando el sistema de nuevo en el menú.

También tendrás que recuperar los datos de Orca, para ello utiliza el ***constructor con dos parámetros*** para crear el objeto de la clase *estadisticas.Estadisticas*.

Por último, has de registrar en el *log* que se ha cargado el sistema *[aaaa-mm-dd hh:mm:ss] Sistema cargado.*

Peces

Implementa los peces que necesites para asegurarte de que dispones de un pez filtrador, un omnívoro y un carnívoro para cada tipo de piscifactoría. Los dobles valen como el tipo correspondiente en ambas.