Trabajo de Laboratorio Nro 3

Paradigmas y Lenguajes de Programación - Trelew

Lic. Pablo Toledo Margalef Lic. Lautaro Pecile

Año 2024

Integrantes:

- Toro Santiago
- Perez Luciana
- Gonzalez Alejo

DECISIONES DE DISEÑO

Estructuración de clases

A la hora de encarar las clases con las que trabajaremos, decidimos emplear en principio una clase denominada Refugio, en torno a la cual iremos realizando las modificaciones necesarias para cubrir los distintos incisos del objetivo 1 propuesto en el laboratorio.

Ya en el objetivo 2, ampliaremos nuestro paquete de clases con una nueva jerarquía, que comienza con una clase genérica llamada Sala, de la que heredarán las subclases Cocina, Cisterna, Generador y Dormitorio. La clase refugio se mantiene, agregando los cambios necesarios para la gestión de salas. Cabe destacar que el refugio trata a todas las salas por igual, cada sala será la responsable de saber cómo responder a cada mensaje enviado por el Refugio (uso del polimorfismo).

Por último, para el objetivo 3 llegamos a la conclusión de que simplemente bastaba con agregar una variable “totalOperadoresNecesarios” dentro de Refugio, mientras que cada sala posee la cantidad de operadores requeridos para sí misma.

A continuación se muestra un listado de todas las clases a implementar con sus correspondientes métodos:

Refugio:

- agregarPersonas: unaCantPersonas
- agregarSala: unaSala
- agua
- comida
- diasOperando
- energia
- estaVivo
- hayCisternas
- hayCocinas
- hayDormitorios
- hayGeneradores
- haySuficientesOperadores
- initialize
- mostrarEstado
- pasarDia
- personas
- puedoBeber
- puedoComer
- puedoGastarEnergia
- puedoPasarElDia
- tengoEspacio: unaCantPersonas
- totalAguaGeneradaPorDia
- totalComidaGeneradaPorDia

totalEnergiaConsumidaPorDia
totalEnergiaGeneradaPorDia

Sala:

cantOperadoresNecesarios
consumoEnergia
initialize

Cisterna:

aguaGeneradaPorDia
cantOperadoresNecesarios
consumoEnergia
initialize

Cocina:

cantOperadoresNecesarios
comidaGeneradaPorDia
consumoEnergia
initialize

Dormitorio:

cantOperadoresNecesarios
capacidadExtraAgregada
consumoEnergia
initialize

Generador:

cantOperadoresNecesarios
consumoEnergia
energiaGeneradaPorDia
initialize

IMPLEMENTACIÓN (MÉTODOS MÁS IMPORTANTES)

Refugio

initialize: establece los valores iniciales y las estructuras de datos necesarias para el funcionamiento del refugio.

```
initialize
    super.
    energia := 10.
    comida := 10.
    agua := 10.
    personas := 0.
    diasOperando := 0.
    capacidadPersonas := 5.
    salas := OrderedCollection new.
    totalOperadoresNecesarios := 0
```

pasarDia: simula el paso de un día en el refugio y verifica si el refugio está vivo y tiene suficientes operadores. Si ambas condiciones se cumplen, incrementa el contador de díasOperando. Genera energía si hay generadores disponibles. Si puede cubrir las necesidades del día, consume la energía diaria requerida, genera y consume el agua y comida necesaria. Si no puede, intenta consumir la energía, comida y agua disponibles, sin generar nuevos recursos. De no ser suficientes, reduce los recursos a cero. Por último muestra el estado actual del refugio en el Transcript.

```
pasarDia
    (self estaVivo and: self haySuficientesOperadores) ifTrue:[
        diasOperando := diasOperando + 1.

        self hayGeneradores  ifTrue: [
            energia := energia + self totalEnergiaGeneradaPorDia.
        ].

        self puedoPasarElDia ifTrue: [
            energia := energia - self totalEnergiaConsumidaPorDia.

            self hayCocinas  ifTrue: [
                comida := comida + self totalComidaGeneradaPorDia.
            ].

            self puedoComer ifTrue: [
                comida := comida - personas.
            ].

            self hayCisternas  ifTrue: [
                agua := agua + self totalAguaGeneradaPorDia.
            ].

            self puedoBeber ifTrue:[
                agua := agua - personas.
            ].

        ] ifFalse: [

            self puedoGastarEnergia ifTrue: [
                energia := energia - self totalEnergiaConsumidaPorDia.
            ] ifFalse: [
                energia := 0.
            ].

            self puedoComer ifTrue: [
                comida := comida - personas.
            ] ifFalse: [
                comida := 0.
            ].

            self puedoBeber ifTrue:[
                agua := agua - personas.
            ] ifFalse: [
                agua := 0.
            ].
        ].
    ].
    self mostrarEstado.
```

agregarSala: agrega una sala al refugio solo si hay suficientes operadores disponibles para la nueva sala(**dormitorio, cocina, cisterna, generador**). Si la condición se cumple, se agrega la sala al refugio. Si es un dormitorio se incrementa la capacidad máxima de personas en el refugio. Por último actualiza el total de operadores que se necesita, sumando los operadores necesarios para la nueva sala.

```
agregarSala: unaSala
    self personas >= (totalOperadoresNecesarios + unaSala cantOperadoresNecesarios) ifTrue:[
        salas add: unaSala.
        (unaSala class == Dormitorio) ifTrue: [
            capacidadPersonas := capacidadPersonas + unaSala capacidadExtraAgregada.
        ].
        totalOperadoresNecesarios := totalOperadoresNecesarios + unaSala cantOperadoresNecesarios
    ]
```

hayCocinas: devuelve verdadero si hay al menos una sala de tipo cocina en el refugio, sino devuelve falso. Utilizamos un select para filtrar todas las salas y seleccionar solo aquellas que son de tipo cocina y después se verifica si el tamaño de la colección es mayor a 0.

```
hayCocinas
    ^((salas select: [:each | each class == Cocina]) size) > 0
```

puedoComer: verifica si hay suficiente comida disponible para todas las personas en el refugio. Compara la cantidad de personas con la cantidad de comida disponible. Devuelve **true** si la cantidad de personas es menor o igual que la cantidad de comida lo que significa que hay suficiente comida sino devuelve **false** lo que significa que no hay suficiente comida.

```
puedoComer
    ^personas <= comida
```

puedoPasarElDia: verifica si el refugio puede operar durante un día completo comprobando que se cumplan tres condiciones, que haya suficiente energía disponible(**puedoGastarEnergia**), y que haya suficiente comida y agua (**puedoComer,puedoBeber**) para proveer a todas las personas en el refugio.

```
puedoPasarElDia
    ^ self puedoGastarEnergia
    and: self puedoComer
    and: self puedoBeber.
```

totalEnergiaConsumidaPorDia: calcula el total de energía que se consume diariamente en el refugio. Inicialmente el consumo fijo por día es 1. Utilizamos un inject para recorrer las demás salas del refugio sumando el consumo de energía de cada una al consumo fijo para obtener el total de energía consumida por día.

```
totalEnergiaConsumidaPorDia
    ^(1 + (salas inject: 0 into: [:sum :each | sum + each consumoEnergia]))
```

totalEnergiaGeneradaPorDia: calcula la cantidad total de energía extra generada por los generadores que hayan el refugio por día. Utilizamos un select para filtrar todas las salas que son instancias de la clase **'generador'**, aplicamos un inject para recorrer todas estas salas y sumar la cantidad de energía generada por cada una.

```
totalEnergiaGeneradaPorDia
    ^((salas select: [:each | each class == Generador])
        inject: 0 into: [:sum :each | sum + each energiaGeneradaPorDia])
```

Sala

consumoEnergia: Como cada sala consume una cantidad de energía diferente, la clase Sala delega la responsabilidad a sus subclases de indicar su consumo.

```
consumoEnergia
    self subclassResponsibility
```

Cocina

initialize: Cada subclase de sala sabe la cantidad de operarios requeridos para su funcionamiento y su consumo de energía, por lo que en sus initialize es donde se hace esta distinción

```
initialize
    cantOperadoresNecesarios := 1.
    consumoEnergia := 1.
    comidaGeneradaPorDia := 5
```

SCRIPT DE PRUEBA

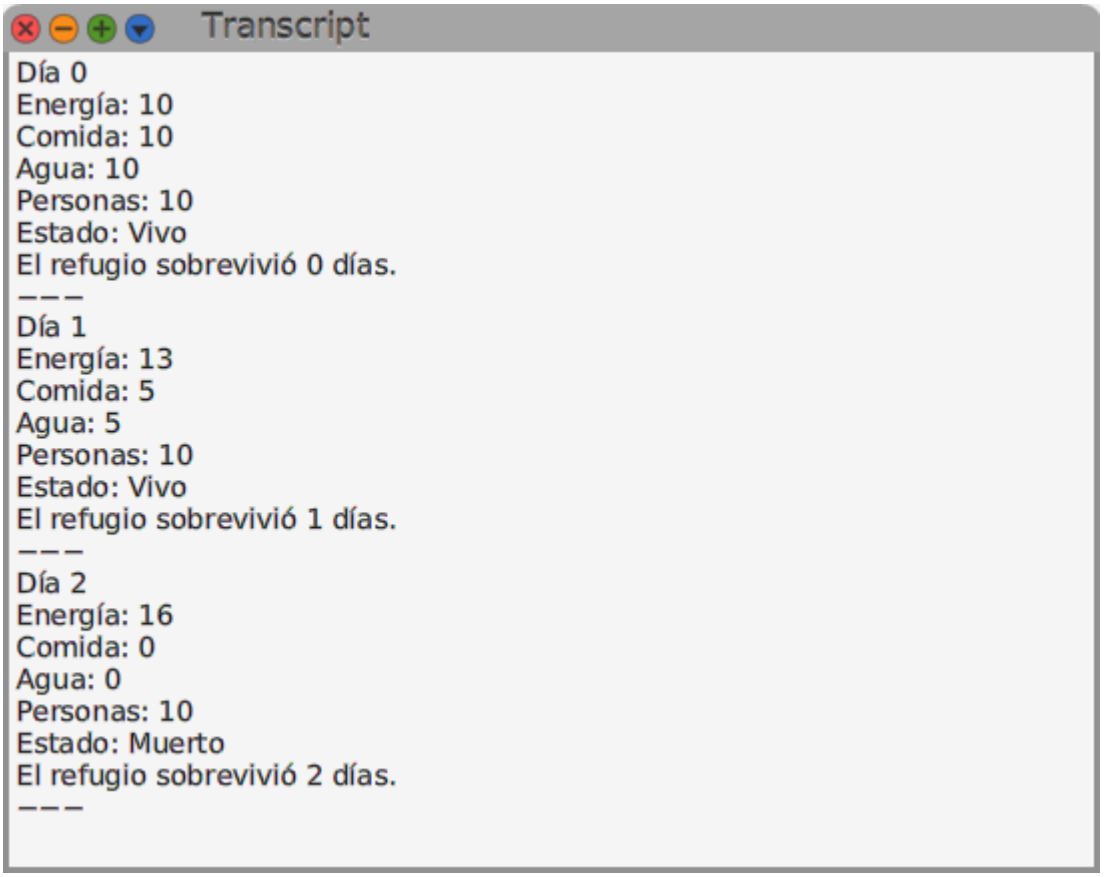
Primer Script: Refugio muere por falta de recursos (agua y comida)

Workspace

```
"--------MUERTE POR FALTA DE COMIDA/AGUA--------"
"Crear instancias de cada tipo de sala"
cocina := Cocina new.
cisterna := Cisterna new.
generador1 := Generador new.
generador2 := Generador new.
dormitorio := Dormitorio new.

"Crear una instancia de Refugio"
refugio1 := Refugio new.
refugio1 agregarPersonas: 5.

"Agregar salas al refugio"
refugio1 agregarSala: cocina.
refugio1 agregarSala: cisterna.
refugio1 agregarSala: generador1.
refugio1 agregarSala: dormitorio.
refugio1 agregarPersonas: 5.
refugio1 agregarSala: generador2.
refugio1 mostrarEstado.

refugio1 pasarDia.
```

Inspect: a Refugio

self	energia: 10
all inst vars	comida: 10
energia	agua: 10
comida	personas: 5
agua	diasOperando: 0
personas	capacidadPersonas: 5
diasOperando	salas: an OrderedCollection(a Cocina a Cisterna a Generador)
capacidadPersonas	totalOperadoresNecesarios: 5
salas	
totalOperadoresNecesarios	



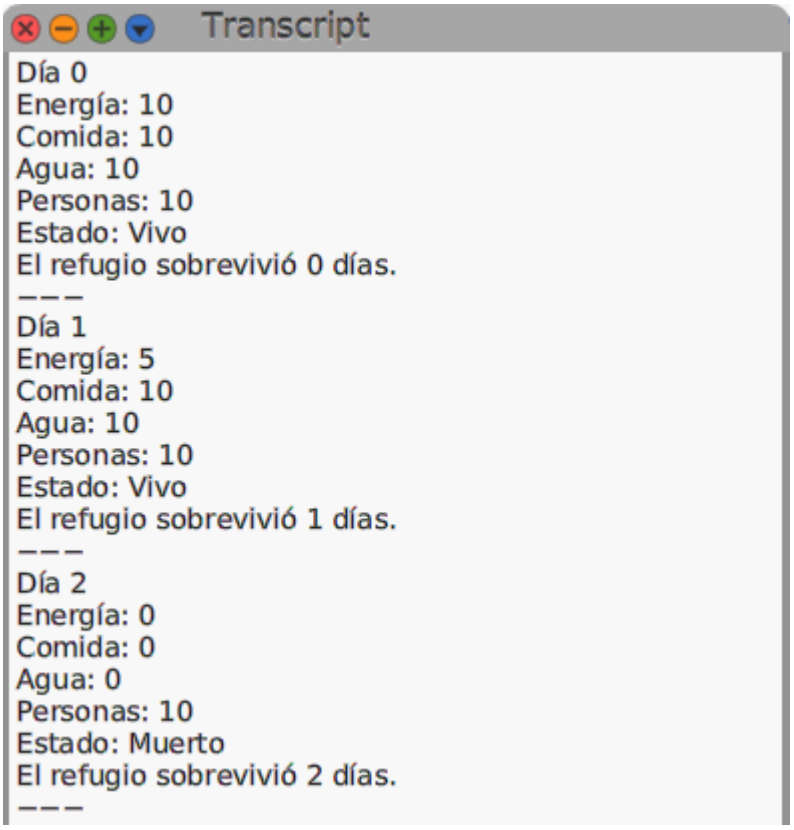
Segundo Script: Refugio muere por falta de energía

Workspace

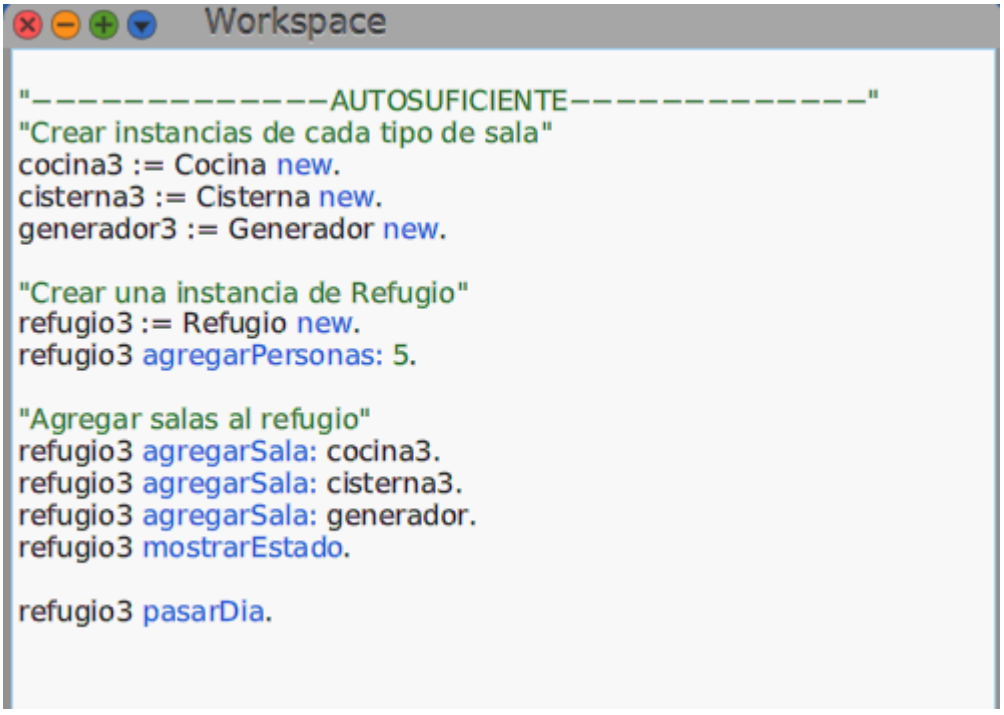
```
"----------MUERTE POR FALTA DE ENERGIA----------"
"Crear instancias de cada tipo de sala"
cocina1 := Cocina new.
cocina2 := Cocina new.
cisterna1 := Cisterna new.
cisterna2 := Cisterna new.
generador := Generador new.
dormitorio1 := Dormitorio new.
dormitorio2 := Dormitorio new.

"Crear una instancia de Refugio"
refugio2 := Refugio new.
refugio2 agregarPersonas: 5.

"Agregar salas al refugio"
refugio2 agregarSala: cocina1.
refugio2 agregarSala: cocina2.
refugio2 agregarSala: cisterna1.
refugio2 agregarSala: dormitorio1.
refugio2 agregarPersonas: 5.
refugio2 agregarSala: cisterna2.
refugio2 agregarSala: generador.
refugio2 agregarSala: dormitorio2.
refugio2 mostrarEstado.

refugio2 pasarDia.
```

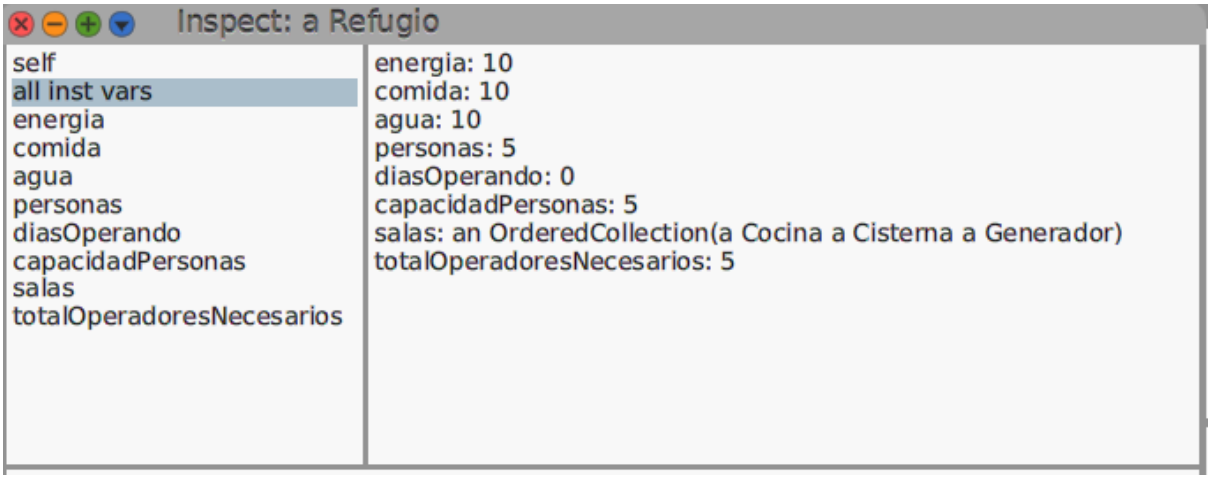
Inspect: a Refugio

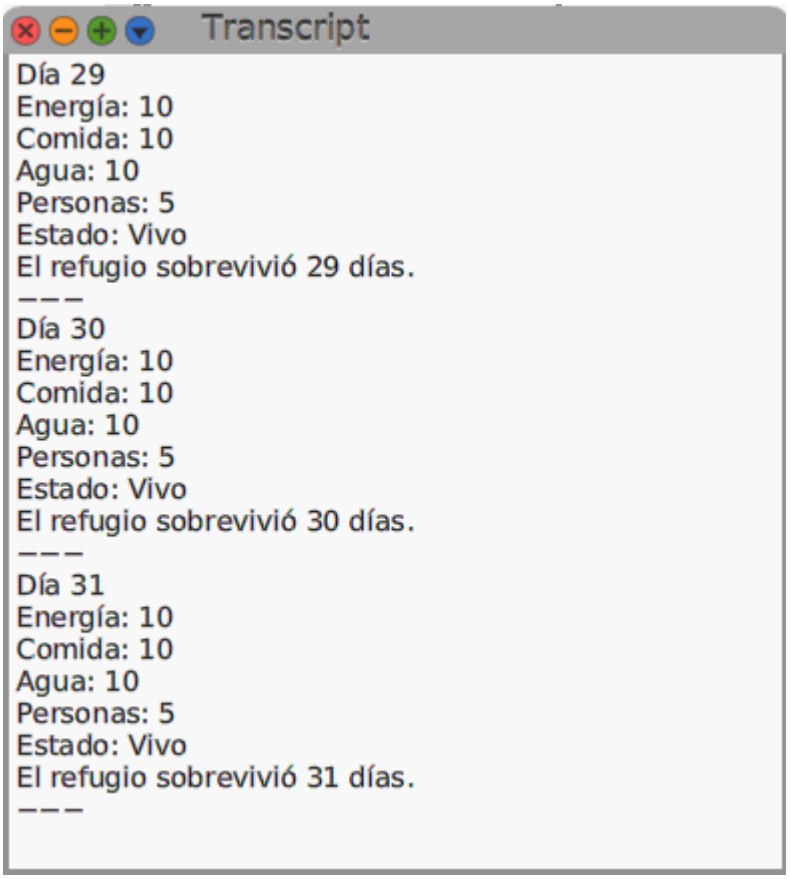
self	energia: 10
all inst vars	comida: 10
energia	agua: 10
comida	personas: 10
agua	diasOperando: 0
personas	capacidadPersonas: 15
diasOperando	salas: an OrderedCollection(a Cocina a Cocina a Cisterna a Dormitorio a Cisterna a Generador a Dormitorio)
capacidadPersonas	totalOperadoresNecesarios: 8
salas	
totalOperadoresNecesarios	



Tercer Script: Refugio no muere, es autosuficiente




Inspect: a Refugio
self
all inst vars
energia
comida
agua
personas
diasOperando
capacidadPersonas
salas
totalOperadoresNecesarios
energia: 10
comida: 10
agua: 10
personas: 5
diasOperando: 0
capacidadPersonas: 5
salas: an OrderedCollection(a Cocina a Cisterna a Generador)
totalOperadoresNecesarios: 5


Transcript
Día 29
Energía: 10
Comida: 10
Agua: 10
Personas: 5
Estado: Vivo
El refugio sobrevivió 29 días.

Día 30
Energía: 10
Comida: 10
Agua: 10
Personas: 5
Estado: Vivo
El refugio sobrevivió 30 días.

Día 31
Energía: 10
Comida: 10
Agua: 10
Personas: 5
Estado: Vivo
El refugio sobrevivió 31 días.
