# DESARROLLO ORIENTADO POR OBJETOS [DOPO-POOB]

**Herencia e interfaces ADEMAS Java desde consola 2025-2**

**Laboratorio 3/6**

# OBJETIVOS

Desarrollar competencias básicas para:

1. Aprovechar los mecanismos de la herencia y de interfaces.
2. Organizar las fuentes en paquetes.
3. Usar la utilidad jar de java para entregar una aplicación.
4. Extender una aplicación cumpliendo especificaciones de diseño, estándares y verificando su corrección.
5. Vivenciar las prácticas XP :  Code must be written to agreed [standards](http://www.extremeprogramming.org/rules/standards.html). Code the [unit test first](http://www.extremeprogramming.org/rules/testfirst.html).
6. Utilizar los programas básicos de java (javac, java, javadoc, jar), desde la consola.

# ENTREGA

* + Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
  + Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin.

DESARROLLO

## Contexto

El modelo clásico de **Lotka–Volterra** fue propuesto originalmente para explicar variaciones en las poblaciones de peces en el Mediterráneo, pero desde entonces se ha utilizado para explicar la dinámica de cualquier sistema depredador-presa en el cual ciertas suposiciones sean válidas. [[VER SIMULADOR]](https://preylife.org/)

Nuestro sistema Lotka-Volerra será de lobos (predadores) , ovejas (presa) y heno (recursos). Todos habitan un valle, representado en una cuadricula bidimensional.

El valle se modifica en una secuencia de pasos. En cada paso cada unidad (lobo, oveja o heno), que no ha actuado, atúa en orden de izquierda a derecha y de norte a sur.

**Conociendo** [En lab03.doc y valley.asta ]

1. En el directorio descarguen los archivos contenidos en valley.zip. Revisen el código:

a) ¿Cuántos paquetes tiene?

Tiene dos paquetes el de presentation y el domain

b) ¿Cuál es el propósito del paquete presentación?

El paquete presentación tiene la misma funcionalidad que tendría un apartado Front-End-

c)¿Cuál es el propósito del paquete dominio?

Es el que se encarga de la lógica del programa que será como el back

1. Revisen el paquete de dominio,

a)¿Cuáles son los diferentes tipos de componentes de este paquete?

Están presentes 2 clases abstractas, 1 interfaz y 2 clases normales

b) ¿Qué implica cada uno de estos tipos de componentes?

La clase Mammal está implementado por la interfaz Unidad, y a la vez es una clase hija de la clase Animal. A su vez, Wolf es una clase hija de Mammal, lo que significa que Wolf tiene heredados metodos y atributos de Mammal, Animal y Unit. Mientras que Valley solo utiliza la interfaz Unit, la clase Wolf y Mammal

1. Revisen el paquete de presentación,

a) ¿Cuántos componentes tiene?

Solo tiene una clase concreta que es ValleyGUI

b) ¿Cuántos métodos públicos propios (no heredados) ofrece?

Tiene dos metodos publicos propios que son **gettheValley** y **main**

1. Para ejecutar un programa en java, ¿Qué método se debe ejecutar? ¿En qué clase se encuentra?
2. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?

## (Deben ejecutar la aplicación java, no crear un objeto como lo veníamos haciendo)

**Arquitectura general.** [En lab03.doc y valley.asta]

1. Consulte el significado de las palabras package e import de java. ¿Qué es un paquete? ¿Para qué sirve? ¿Para qué se importa? Explique su uso en este programa.

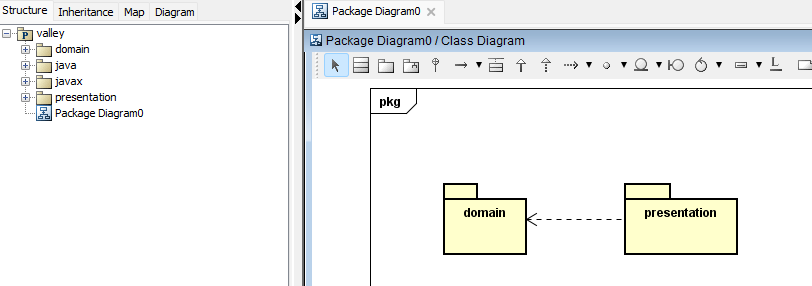
Un paquete puede contener una o más definiciones de interfaces y clases, distribuyéndose habitualmente como un archivo.

es una palabra clave que se utiliza para hacer visibles otras clases o paquetes enteros dentro de una clase actual. Esto permite utilizar clases de otros paquetes sin tener que hacer referencia a sus nombres completos.

1. Revise el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido. ¿Qué coincidencia hay entre paquetes y directorios?

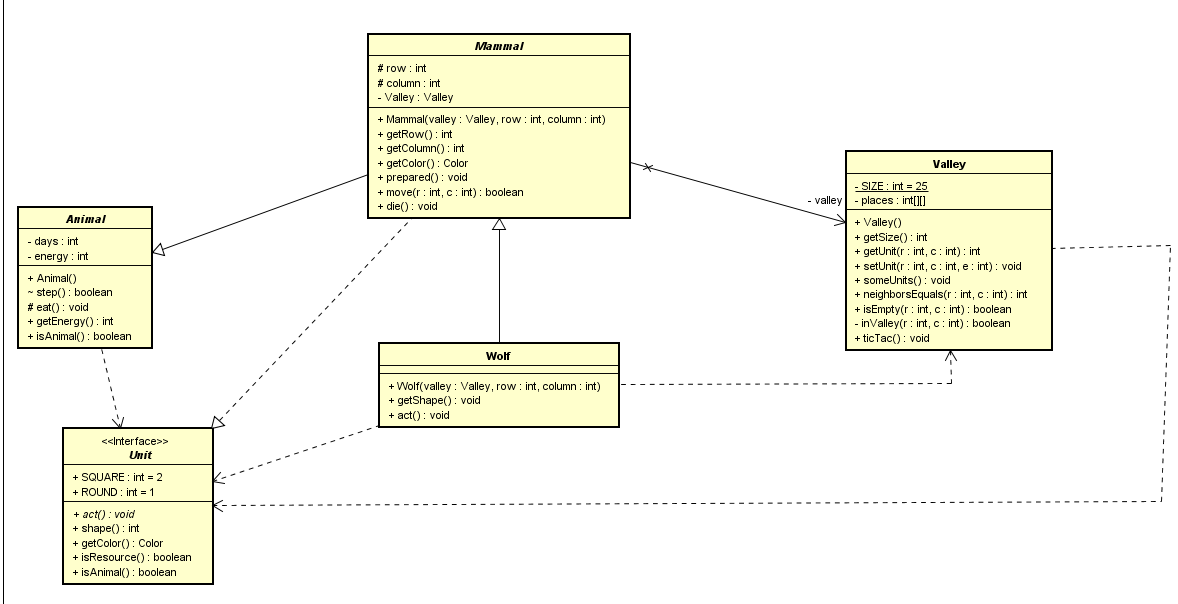
Se encuentran los paquetes que son los de la parte visual y otro que contiene la logica del programa, que uno es el que contiene los archivos y el otro indica su hubicacion.

1. Adicione al diseño la arquitectura general con un diagrama de paquetes en el que se presente los paquetes y las relaciones entre ellos. Consulte la referencia en moodle. **En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)**



**Arquitectura detallada.** [En lab03.doc y valley.asta]

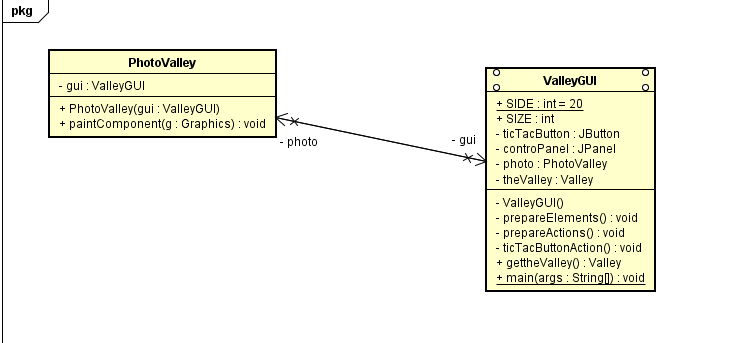
* 1. Para preparar el proyecto para **BDD.** Completen el diseño detallado del paquete de dominio. Adicionen el diagrama de clases en el paquete correspondiente.



a) ¿Qué componentes hacían falta?

Hacían falta la interfaz de Unit y la clase Wolf, además del atributo places en Valley, y el atributo Valley en Mannal.

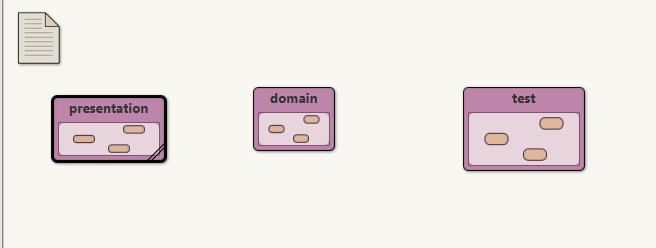
* 1. Completen el diseño detallado del paquete de presentación. Adicionen el diagrama de clases al paquete correspondiente.

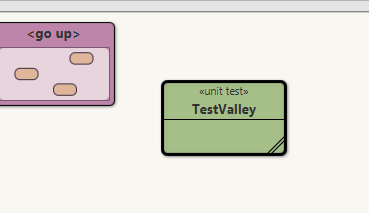


a) ¿Por qué hay dos clases y un archivo .java?

Porque la clase photoValley es una clase de apoyo que es utilzada por ValleyGUI. Tambien como están muy relacionadas se facilita el mantenimiento de la clase.

* 1. Adicione la clase de pruebas unitarias necesaria para **BDD** en un paquete independiente de test**.** (No lo adicione al diagrama de clases)





a) ¿Qué clase debe importar?

Debe importar la clase Valley

b) ¿Por qué?

Debido a que ahí está presente toda la lógica interna del paquete al que se le requiere hacer pruebas

**Ciclo 1. Iniciando con los lobos [En lab03.doc y \*.java]**

# (NO OLVIDE BDD – MDD)

1. Estudie la clase Valley ¿Qué tipo de colección usa para albergar cosas? ¿Puede recibir lobos? ¿Por qué?

Utiliza el tipo de colección Unit para albergar elementos dentro

1. Estudie el código asociado a la clase Wolf,

a) ¿de qué color es?

b) ¿qué forma usa para pintarse?

c) ¿cuándo pierde su energia?

d) ¿qué componentes definen la clase Wolf ? Justifique sus respuestas.

1. Wolf por ser un Mammal, ¿qué atributos tiene? ¿qué puede hacer (métodos)? ¿qué decide hacer distinto (sobreescritura)? ¿qué no puede hacer distinto (métodos finales)? ¿qué debe aprender a hacer (métodos abstractos)? Justifique sus respuestas.
2. Por comportarse como un Unit, ¿qué sabe hacer? ¿qué decide hacer distinto? ¿qué no puede hacer distinto? ¿qué debe aprender a hacer? Justifique sus respuestas.
3. De acuerdo a lo anterior un Wolf, ¿Cómo actúa?
4. Ahora vamos a crear dos lobos en diferentes posiciones (10,10) (15,15) , llámelos akela y larka , usando el método someUnits() . Ejecuten el programa, ¿Qué pasa con los lobos? ¿Por qué? Capturen una pantalla significativa.
5. Diseñen, construyan y prueben el método llamado ticTac() de la clase Valley.
6. ¿Cómo quedarían akela y larka después de uno, sesenta y ciento uno **Tic-tac**? Ejecuten el programa. Capturen pantallas significativas en los momentos correspondientes. ¿Es correcto?

**Ciclo 2. Incluyendo a las ovejas** [En lab03.doc y valley.asta]

# (NO OLVIDE BDD – MDD)

El objetivo de este punto es permitir recibir ovejas. Ellas (i) son rectangulos de color gris claro (ii) inician con 5 puntos de energía; (ii) se mueven en siempre hacia el norte y si se llegan al norte cambian de dirección camino hacia el sur (iii) cada paso pierden 1 punto de energía (iv) al llegar al norte recuperan toda su energía (v) si quedan vecinas a una oveja ganan 1 punto de energia (vi) si quedan vecinas a un lobo mueren.

1. Para implementar este nuevo Mammal ¿cuáles métodos se sobre-escriben (overriding)?
2. Diseñen, construyan y prueben esta nueva clase. (Mínimo dos pruebas de unidad)
3. Adicione una pareja de ovejas, llámelas shaun y woolly*,* (a) ¿Cómo quedarían después de tres **Tic-tac**? Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa. (b) ¿Es correcto?

**Ciclo 3. Adicionando heno** [En lab03.doc, valley.asta y \*.java]

El objetivo de este punto es incluir paquetes de heno (sólo vamos a permitir el tipo básico de paquetes de heno) los paquetes de heno son cuadrados y van cambiando de color: rojo, amarillo, rojo, amarillo, rojo.

# (NO OLVIDE BDD – MDD)

1. Para poder adicionar paquetes de heno, ¿debe cambiar en el código de Valley en algo? ¿por qué?
2. Diseñen , construyan y prueben esta nueva clase. (Mínimo dos pruebas de unidad)
3. Adicionen dos paquetes de heno en las esquinas superiores de el valle, llámenlos alarm y alert*,* (a) ¿Cómo quedarían después de cuatro **Tic-tac**? Ejecuten el programa y hagan cuatro clics en el botón. Capturen una pantalla significativa. (b)

¿Es correcto?

## Ciclo 4. Nuevo Mamifero: Proponiendo y diseñando

El objetivo de este punto es permitir recibir en un nuevo tipo de Mamifero.

# (NO OLVIDE BDD – MDD)

1. Propongan, describan e implementen el nuevo tipo de Mamifero. (Mínimo dos pruebas de unidad)
2. Considerando una pareja de ellos con el apellido de ustedes. (a) Piensen en otra prueba significativa y expliquen la intención. (b) Codifiquen la prueba de unidad correspondiente y capturen la pantalla de resultados de ejecución de la prueba. (c) Ejecuten el programa con esa prueba como prueba de aceptación y capturen las pantallas correspondientes.

## Ciclo 5. Nuevo Unit: Proponiendo y diseñando

El objetivo de este punto es permitir recibir un nuevo Unit (no Animal) en el valle

# (NO OLVIDE BDD – MDD)

1. Propongan, describan e implementen el nuevo tipo de Unit. (Mínimo dos pruebas de unidad)
2. Considerando un par de ellos con el nombre de ustedes. (a) Piensen en otra prueba significativa y expliquen la intención. (b) Codifiquen la prueba de unidad correspondiente y capturen la pantalla de resultados de ejecución de la prueba. (c) Ejecuten el programa con esa prueba como prueba de aceptación y capturen las pantallas correspondientes.

## Ciclo 6. BONO. Lobo y Oveja lotVol

Los mamiferos LotVol se rigen por las siguientes reglas:

* + Los animales se mueven aleatoriamente por todo el valle. Un paso a la vez.
  + Si un lobo queda cerca a una oveja, se la come y gana el 90% de la energía de la oveja.
  + Si una oveja queda cerca al heno, se lo come y aumenta 10% de su energía.
  + Si dos animales de la misma raza quedan a un espacio de distancia y ese espacio está vacío, se reproducen. El nuevo mamífero no actua sino hasta la siguiente vuelta.

**Empaquetando la versión final para el usuario.** [En lab03.doc, valley.asta , \*.java, valley.jar]

1. Revise las opciones de BlueJ para empaquetar su programa entregable en un archivo .jar. Genere el archivo correspondiente.
2. Consulte el comando java para ejecutar un archivo jar. ejecutennlo ¿qué pasa?
3. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.

# DE BLUEJ A CONSOLA

En esta sección del laboratorio vamos a aprender a usar java desde consola. Para esto se va a trabajar con el proyecto del punto anterior.

**Comandos básicos del sistema operativo** [En lab03.doc]

es de iniciar debemos repasar los comandos básicos del manejo de la consola.

* 1. Investiguen los comandos para moverse en la estructura de directorios: crear, borrar, listar su contenido y copiar o eliminar un archivo.
  2. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y capturen el contenido de su directorio

lotVol

src

domain presentation test

* 1. En el directorio copien únicamente los archivos \*.java del paquete de aplicación .

Consulte y capture el contenido de src/domain

**Estructura de proyectos java** [En lab03.doc]

En java los proyectos se estructuran considerando tres directorios básicos.

lotVol

src bin docs

1. Investiguen los archivos que deben quedar en cada una de esas carpetas y la organización interna de cada una de ellas.
2. ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué? Cópielos y consulte y capture el contenido del directorio que modificó.

**Comandos de java** [En lab03.doc]

1. Consulte para qué sirven cada uno de los siguientes comandos:

javac java javadoc jar

1. Cree una sesión de consola y consulte en línea las opciones de los comandos java y

javac. Capture las pantallas.

1. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.

**Compilando** [En lab03.doc]

1. Utilizando el comando javac, **desde el directorio raiz (desde lotVol con una sóla instrucción)**, compile el proyecto. ¿Qué instrucción completa tuvo que dar a la consola para compilar TODO el proyecto? Tenga presente que se pide un único comando y que los archivos compilados deben quedar en los directorios respectivos.
2. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?

**Documentando** [En lab03.doc]

1. Utilizando el comando javadoc, desde el directorio raiz, genere la documentación (API) en formato html, en este directorio. ¿cuál es el comando completo para generar esta documentación?
2. ¿Cuál archivo hay qué abrir para empezar a navegar por la documentación? Ábralo y capture la pantalla.

**Ejecutando** [En lab03.doc]

4. Empleando el comando java, desde el directorio raiz, ejecute el programa. ¿Cómo utilizó este comando?

**Probando** [En lab03.doc]

1. Adicione ahora los archivos del directorio pruebas y trate de compilar nuevamente el programa.Tenga en cuenta que estas clases requieren la librería junit 4.8. ¿Cómo se incluye un paquete para compilar? ¿Qué instrucción completa tuvo que dar a la consola para compilar?
2. Ejecute desde consola las pruebas . ¿Cómo utilizó este comando?. Puede ver ejemplos de cómo ejecutar el“test runner”en [How to run JUnit test cases from the command line](https://stackoverflow.com/questions/2235276/how-to-run-junit-test-cases-from-the-command-line)
3. Pegue en su documento el resultado de las pruebas

**Empaquetando** [En lab03.doc]

1. Consulte como utilizar desde consola el comando jar para empaquetar su programa entregable en un archivo .jar, que contenga los archivos bytecode necesarios (no las fuentes ni las clases de prueba), y que se pueda ejecutar al instalarlo en cualquier directorio, con solo tener la máquina virtual de java y su entorno de ejecución (JRE).

¿Cómo empaquetó jar ?

1. ¿Cómo se ejecuta el proyecto empaquetado?

# RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)
2. ¿Cuál es el estado actual de laboratorio? ¿Por qué? (Para cada método incluya su estado)
3. Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son importe?
4. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?
5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?
6. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.

<https://www.campusmvp.es/recursos/post/paquetes-en-java-que-son-para-que-se-utilizan-y-como-se-usan.aspx?srsltid=AfmBOoqrkrtTzTBNbRF6azwrMUH9s3py0yzBZ29PjW6qUJI9td-Aa8hn>

<https://www.datacamp.com/es/doc/java/import>