# Proyecto
# Serengeti National Park- PARTE 2

## Objetivos:

Practicar de manera más autónoma los conceptos de:

- Jerarquía de clases, incluyendo clases abstractas e interfaces

## Herramientas que vamos a utilizar:

- Entorno de diseño StarUML
- Entorno de desarrollo Eclipse

## Entregables

La entrega de esta segunda parte del proyecto se compone de tres entregables.

- La **entrega** es **individual** y se subirá a eGela
- **Fecha límite** de entrega: **domingo 17** de **mayo** a las **23:55**

*Entregable1:* Diagrama de clases UML para la jerarquía `Specimen`. Se debe entregar el fichero **.uml** y una imagen exportada, que contengan el diagrama de clases UML del proyecto diseñado con StarUML.

- Nombre del fichero: *apellido_nombre_parte2*.uml

*Entregable2:* La implementación documentada (proyecto exportado) de esta segunda parte completa, tal y como se indica en las tareas a realizar.

- Nombre del proyecto: apellido_nombre_parte2
- Nombre del fichero: apellido_nombre_parte2.zip

*Entregable3:* Un documento en formato pdf (con nombre apellido_nombre_parte2) que incluya:

- Explicaciones de aquellas partes de la entrega que se duda de su correcto funcionamiento o directamente que no funciona correctamente
- Informe de todo lo realizado y los problemas que se han tenido en su desarrollo
- Estudio de casos de prueba considerados para la elaboración de los JUnit
- Captura de una ejecución completa

**NOTA: Aunque el proyecto se contextualiza también en el parque del Serengeti, lo realizaremos como un proyecto nuevo y sin considerar las características anteriormente descritas. Trabajaremos a partir de la clase Specimen y conservaremos sólo su nombre y posición inicial (clase GPS).**

## Refinamiento de aspectos concretos del parque y sus especies

El parque tiene *llanuras* de pastizales y *kopjes* donde conviven muchos **mamíferos**. La *llanura* es una pradera casi sin árboles y es el paisaje más emblemático del parque. Aquí es donde conviven la cebra, perro salvaje, hipopótamos y rinocerontes negros (entre otros). Los hipopótamos y rinocerontes negros se comunican barritando, mientras que la cebra relincha. Salvo el perro salvaje que es **carnívoro**, los demás animales de las llanuras son **herbívoros** que se alimentan de plantas, pastos y hierbas**.** Los perros salvajes como carnívoros se alimentan de otros animales y su modo de comunicación es mediante chillidos. Las cebras y perros salvajes tienen un andar ligero, mientras que rinocerontes e hipopótamos son más lentos.

Además, los kopjes son flotaciones de granito muy comunes en todo el parque, y son excelentes lugares de observación para depredadores felinos. Los felinos son una familia de mamíferos **carnívoros**, con un andar sigiloso, y un potente rugido utilizado para comunicarse. Los felinos más destacados del parque son los leones y leopardos.

Entre los **reptiles** del Parque están el cocodrilo, la tortuga y la víbora. Muchos de ellos se alimentan de pequeños animales e insectos, pero el cocodrilo lo hace de grandes animales. Por su parte, su modo de comunicación en general suele ser el siseo, aunque los cocodrilos se comunican con gruñidos.

Por su parte, las especies en peligro de extinción más prominentes en Serengeti son el rinoceronte negro y el perro salvaje, aunque el cocodrilo también está en peligro. El rinoceronte negro estuvo casi extinto en Serengeti en la década de 1990. Debido a la implementación de medidas rígidas de protección, la población de esta especie en peligro de extinción volvió a crecer, aunque aún está en peligro. Todos los animales que están o han estado en peligro de extinción, y solo ellos, tienen un nivel de peligro que es necesario conocer (`whatDangerLevel`). El nivel de peligro tiene unos valores que van desde 0.00 (actualmente han salido del peligro) a 3.00 (máximo peligro). Se considera que los rinocerontes negros tienen un nivel de peligro de 2.2, mientras que los perros están a un nivel de 1.7 y el cocodrilo a 0.5. Por supuesto es deseable poder saber si un determinado ejemplar está o no en peligro de extinción y si es así, saber cuál es su nivel de peligro.

Así, los aspectos que caracterizan a todos los animales del parque son su alimentación (`feeding`) y su modo de comunicación (`sound`). Además, los mamíferos se caracterizan por su modo de andar (`walk`). Para ello se les dotará de la funcionalidad necesaria para obtener dicha información. Para simplificar bastará con devolver un String con dicha información. Por ejemplo,

la alimentación de un Herbívoro, bastaría que devolviera "Plants, grasses and herbs", el sonido de un rinoceronte sería "Whinny", o el andar sigiloso de los felinos sería "Stealthy walk".

Por último, se desea obtener en un String el camino de herencia en la jerarquía de un objeto. Por ejemplo, si tengo un objeto cebra (Zebra c;) convenientemente inicializado y quiero obtener su camino de herencia: `System.out.println(c.inheritancePath());` debería escribir:
*" Zebra < Herbivorous < Mammal < Specimen".*

## Actividades a realizar:

A. **Diseña con StarUML el diagrama de clases para la jerarquía** de **`Specimen`** y considera las clases abstractas, concretas e interfaces surgidas. Presta atención a las características (atributos) y funcionalidades (métodos) a asociar en cada clase. Considera cómo deben ser los métodos, abstractos o concretos, y sitúalos explícitamente sólo en aquellas clases que lo precisen. Diseña todas las clases en un paquete `packSpecimenHierarchy`.

B. Implementa en el `packSpecimenHierarchy` y documenta toda la jerarquía recogida en el UML. Implementa de manera incremental `inheritancePath` (reutilizando al máximo el código). En general, debes pensar en reutilizar al máximo el código y seguir buenas prácticas de implementación que favorezcan un mejor manteninmiento.

C. Crea en un paquete `packTest` una clase de test JUnit por cada clase concreta de `Specimen`. Añade las pruebas necesarias (tantos @Test como sean necesarios), para que te permitan verificar, en los casos que lo requiera, la respuesta correcta al nombre, a la alimentación, al andar, al modo de comunicación, al camino de herencia y nivel de peligro.

D. Implementa en un paquete `packProof` una clase **`SerengetiProof`** que contenga:
   a. Un método (**`static`**) **`whoIsInSerengeti`** que dado un ArrayList de objetos de tipo **`Specimen`**, escriba por pantalla la información de cada uno de ellos (la información

```
Specimens in Serengeti:
      BlackRhino: 2
       - Feeding: Plants, grasses and herbs
       - Walk: Low walk
       - Sound:Trumpet
       - Level of Danger2.2
      WildDog: 3
       - Feeding: Others Specimens
     ...
```

   pertinente de feeding, walk, sound y nivel de peligro, dependiendo del tipo de clase concreta) pero agrupada tal y como se observa en el siguiente ejemplo:

   b. Un método **`main`** que agrupe en un ArrayList un grupo numeroso de objetos Specimen de todos los tipos concretos definidos en la jerarquía. Asegúrate que haya unos cuantos ejemplares en peligro de extinción (2 rinocerontes negros, 3 perros salvajes y un cocodrilo, por ejemplo). Llama al método **`whoIsInSerengeti`** pasándole como parámetro el `ArrayList` inicializado, y comprueba que funciona correctamente.

   c. Un método (**`static`**) **`howManyEndangeredSpecimen`** que, dada una lista de ejemplares y un nivel de peligro, devuelva el número de ejemplares totales en peligro de extinción que tengan un nivel de peligro mayor o igual que el `nivel` dado. Además, escribirá por

pantalla por cada ejemplar que tenga o supere dicho nivel de peligro (ver ejemplo), el nombre y la clase del ejemplar.
Por ejemplo, una ejecución de dicha operación podría ser:

```
Specimens in danger of extinction equal to or greater than level
1.0:
      Mario- BlackRhino
      Luigi- WildDog
      Gran colmillo- Crocodile
      Maria- WildDog
      Tha- WildDog
      Serekan- BlackRhino
```

Añade al **main** una llamada a este método con el ArrayList anteriormente creado y el nivel 1 de peligro y escribe por pantalla su resultado (Por ejemplo, "`TOTAL: 6 specimens`"). Comprueba que funciona correctamente.

d. Implementa el método **static** `checkExtinctionDanger(Mammal m)` que dado un mamífero indica si está o no en peligro de extinción (boolean).
   - Actualiza el main y declara las siguientes variables: `Zebra z; Mammal m; Specimen s; Object o; Reptile r;`
   - Inicialízalas con objetos compatibles al tipo definido (puedes usar referencias del ArrayList inicializado o variables que ya tengas inicializadas en el main). Considera para ello que esas variables serán las que se utilizarán para llamar al método `checkExtinctionDanger` y que en algún caso debería dar la respuesta true y en otros false.
   - Añade al **main** tantas llamadas como variables hemos declarado (al menos las 5 propuestas, como estas:

```
System.out.println("The variable z has Zebra as static type and "
          + z.getClass().getSimpleName()+" as dynamic type. "
            " Is it in danger?? "+ checkExtinctionDanger(z));
```

   Indica para cada una de esas llamadas que no se pueda ejecutar directamente: qué error surge, si es posible eliminar el error y cómo hacerlo. Para indicar el cómo ayúdate de terminología como: tipo estático, tipo dinámico, error de compilación, error de ejecución, casting, conversión automática, etc. Si no puedes evitar el error deja la sentencia en comentarios, pero tienes que lograr que no te genere ningún error de compilación.

**NOTA: Para todos los apartados, indica mediante comentarios donde corresponda (1) si aparecen variables/parámetros polimórficos y (2) si se activará el dispatching durante la ejecución del método.**