PROGRAMACIÒN ORIENTADA A OBJETOS

TALLER No. 5 JAVA	PROGRAMACIÓN ORIENTADA A OBJETOS

Profesor: Jaime Alberto Guzmán Luna

Contenido del taller:

1) Herencia
2) Polimorfismo
3) Ligadura dinámica y estática

1. EJERCICIO

```
1  package paquete1;
2
3  public class Auto {
4
5        protected static int num_autos = 0;
6        public int velocidad = 10;
7        public String pitar = "Piiiiii";
8
9        public Auto(){
10             num_autos++;
11       }
12
13       void adelantar() {
14             System.out.println("Puedo adelantar autos");
15       }
16
17       protected void pitar() {
18             System.out.println(this.pitar);
19       }
20
21        public void arrancar() {
22             System.out.println("Encendido");
23       }
24
25       public void acelerar() {
26             System.out.println("Avanzo constantemente");
27       }
28
29       public void frenar() {
30             System.out.println("Me detengo");
31       }
32
33       public int getVelocidad(){
34             return(this.velocidad);
35       }
```

```
36  }
```

```
1   package paquete2;
2
3   import paquete1.Auto;
4
5   public class Bus extends Auto{
6
7         private String placa;
8         private int capacidad;
9         public int velocidad = 20;
10
11        public Bus(String placa, int capacidad){
12              this.placa = placa;
13              this.capacidad = capacidad;
14        }
15
16        public void acelerar() {
17              System.out.println("Avanzo lento");
18        }
19
20  }
```

```
1   package paquete2;
2
3   import paquete1.Auto;
4
5   public class Moto extends Auto{
6         private String placa;
7         private String modelo;
8         public int velocidad = 30;
9
10        public Moto(String placa, String modelo){
11              this.placa = placa;
12              this.modelo = modelo;
13              this.pitar();
14        }
15
16        public void acelerar() {
17              System.out.println("Avanzo muy rapido");
18        }
19
20  }
```

```
1   package paquete2;
2
3   public class ObjTaller5H {
4         public static void main(String[] args) {
5               Moto moto = new Moto("XYZ123", "2019");
6               Bus bus = new Bus("ABC345", 20);
7               moto.adelantar();
8               bus.arrancar();
```

```
9               bus.pitar();
10              moto.getVelocidad();
11      }
12  }
```

Para pensar…

1. Si deseo modificar el mensaje del método pitar al crear un objeto moto sin alterar la clase Auto ¿Que debo agregarle al código? (Por ejemplo, al llamar el método pitar imprima Las motos no pitan)
2.
3. Suponga que se definió el método

   public void arrancar() {

   System.out.println("Arrancando");

   }

   en la clase Moto y luego en el método main se declara e inicializa un objeto de tipo Auto de esta manera:

   Auto auto;

   auto = moto;

   ¿Qué imprimiría auto.arrancar()? Justifique.
4. Suponga que se declara una nueva clase llamada Cuatrimoto que hereda de la clase Moto y en el main se declaran e inicializan los siguientes objetos:

   Auto auto = new Cuatrimoto();

   Moto moto2 = (Moto) auto;

   ¿Son correctas esas dos líneas? ¿Se genera algún error? Justifique.
5. En la línea 13 de la clase moto, ¿Por qué puedo utilizar el método pitar?
6. Haciendo una pequeña modificación a la clase Auto y utilizando la variable num_autos, sin modificar su modificador de acceso, ¿Como puedo obtener el número de autos creados desde la clase ObjTaller5H?
7. En la línea 7 de la clase ObjTaller5H, ¿Por qué no puedo utilizar el método adelantar, si este fue heredado?
8. En la línea 8, ¿Por qué es posible utilizar el método arrancar si la clase Bus no lo define?
9. En la línea 9 de la clase ObjTaller5H, ¿Por qué no puedo utilizar el método pitar, si este fue heredado?
10. En la línea 10 de la clase ObjTaller5H, ¿Qué imprime el método getVelocidad()? ¿Por qué?
11. Si quisiera obtener el valor de la placa de las clases Moto y Bus, además de su modelo y capacidad respectivamente, ¿Que debo agregar al código?

PROGRAMACIÒN ORIENTADA A OBJETOS

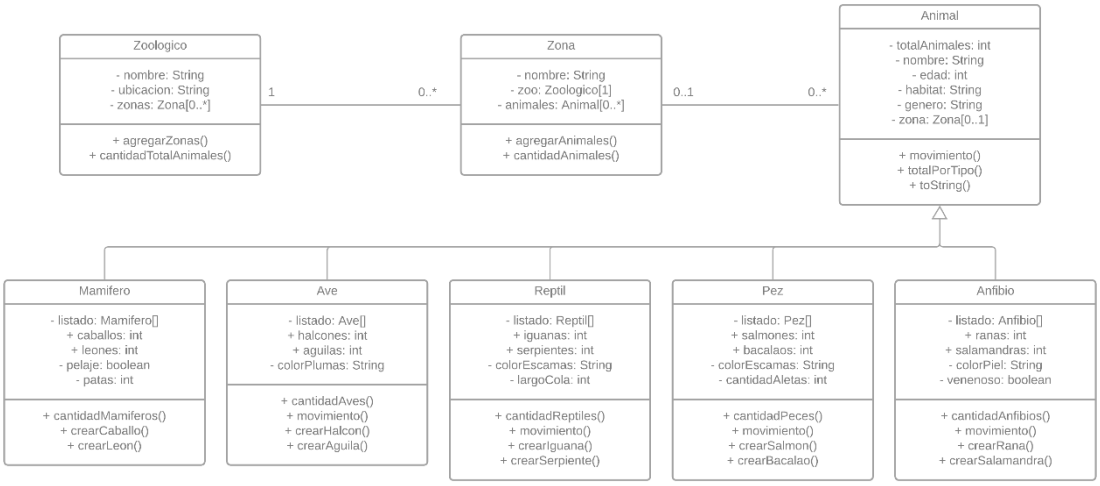
2. EJERCICIO PRACTICO GITHUB

Ingrese al siguiente enlace para aceptar y empezar el ejercicio:
https://classroom.github.com/a/cADGZekw

Mi propio Zoológico

Supongamos que la fábrica de televisores fue un éxito, y ahora decides tener tu propio zoológico virtual… (en el proceso de realización del taller no se lastima ningún ser vivo).

Queremos entonces crear un software que lleve el control y modelado de mi zoológico, para esto se le dieron ciertas indicaciones:



- Se le brindo el anterior diagrama de clases, guíese para crear las clases, atributos y métodos que se le están pidiendo.
- Las clases Zoologico y Zona créelas en un paquete llamado gestion y las clases Animal y sus subclases añádalas al paquete zooAnimales.
- Para cada caso cree un constructor con los parámetros de los atributos proporcionados en cada clase en el orden que aparecen los atributos, en caso de herencia van primero los atributos de la clase que hereda, y además cree el constructor vacío para cada clase.
- Para la clase Zoológico el método cantidadTotalAnimales, retornara la cantidad de animales total de todas las zonas que tengan relación con el Zoológico, agregarZonas será el método encargado de agregar nuevas zonas al zoologico.
- Para la clase Zona el método agregarAnimales añadira un nuevo animal al listado de animales y el método cantidadAnimales retornara la cantidad de animales en la zona.
- Para la clase Animal tenemos ciertas particularidades, queremos llevar el conteo del total de animales creados y que este valor podamos almacenarlo en el atributo totalAnimales, el método totalPorTipo, devolverá el siguiente formato con la cantidad de animales por cada subclase de animales:

  “Mamiferos: #
  Aves: #
  Reptiles: #
  Peces: #
  Anfibios: #“

es el numero de animales por cada subclase.

- El método toString de la clase Animal retornara el siguiente formato:
  "Mi nombre es #nombre, tengo una edad de #edad, habito en #habitat y mi genero es #genero, la zona en la que me ubico es #zona, en el #zoo" cambie respectivamente los valores contiguos con el #, tenga en cuenta que los valores de zoo y zona son los valores del nombre del respectivo objeto, en case de que el animal no tenga una zona solo imprima hasta el genero.
- movimiento es un método que retornará el valor del como el animal se mueve, para la clase Animal este valor será de "desplazarse", para la clase Ave el valor será de "volar", para Reptil será "reptar", para Pez será "nadar" y para Anfibio será "saltar"
- Las subclases de Animal tendrán un atributo listado en donde se almacena cada objeto al ser creado, del tipo de la subclase.

Los siguientes métodos de creación retornaran el objeto creado:

- La clase Mamifero tendrá algunos comportamientos particulares, en el atributo caballos y leones se quiere llevar el conteo de veces que se usó el método crearCaballo y el método crearLeon y el metodo cantidadMamiferos retornara la cantidad de mamíferos creados en el sistema.
- Para la clase Mamifero el método crearCaballo, creará un Mamifero con los valores para pelaje = true, patas = 4 y hábitat = "pradera", los demás valores los recibirá como parámetros.
- Para la clase Mamifero el método crearLeon, creará un Mamifero con los valores para pelaje = true, patas = 4 y hábitat = "selva", los demás valores los recibirá como parámetros.
- La clase Ave tendrá algunos comportamientos particulares, en el atributo halcones y aguilas se quiere llevar el conteo de veces que se usó el método crearHalcon y el método crearAguila y el metodo cantidadAves retornara la cantidad de aves creadas en el sistema.
- Para la clase Ave el método crearHalcon, creará un Ave con los valores para colorPlumas = "cafe glorioso" y hábitat = "montanas", los demás valores los recibirá como parámetros.
- Para la clase Ave el método crearAguila, creará un Ave con los valores para colorPlumas = "blanco y amarillo" y hábitat = "montanas", los demás valores los recibirá como parámetros.
- La clase Reptil tendrá algunos comportamientos particulares, en el atributo iguanas y serpientes se quiere llevar el conteo de veces que se usó el método crearIguana y el método crearSerpiente y el metodo cantidadReptiles retornara la cantidad de reptiles creados en el sistema.
- Para la clase Reptil el método crearIguana, creará un Reptil con los valores para colorEscamas = "verde", largoCola = 3 y hábitat = "humedal", los demás valores los recibirá como parámetros.
- Para la clase Reptil el método crearSerpiente, creará un Reptil con los valores para colorEscamas = "blanco", largoCola = 1 y hábitat = "jungla", los demás valores los recibirá como parámetros.
- La clase Pez tendrá algunos comportamientos particulares, en el atributo salmones y bacalaos se quiere llevar el conteo de veces que se usó el método crearSalmon y el método crearBacalao y el metodo cantidadPeces retornara la cantidad de peces creados en el sistema.

- Para la clase Pez el método crearSalmon, creará un Pez con los valores para colorEscamas = “rojo”, cantidadAletas = 6 y hábitat = “oceano”, los demás valores los recibirá como parámetros.
- Para la clase Pez el método crearBacalao, creará un Pez con los valores para colorEscamas = “gris”, cantidadAletas = 6 y hábitat = “oceano”, los demás valores los recibirá como parámetros.
- La clase Anfibio tendrá algunos comportamientos particulares, en el atributo ranas y salamandras se quiere llevar el conteo de veces que se usó el método crearRana y el método crearSalamandra y el metodo cantidadAnfibios retornara la cantidad de anfibios creados en el sistema.
- Para la clase Anfibio el método crearRana, creará un Anfibio con los valores para colorPiel = “rojo”, venenoso = true y hábitat = “selva”, los demás valores los recibirá como parámetros.
- Para la clase Anfibio el método crearSalamandra, creará un Anfibio con los valores para colorPiel = “negro y amarillo”, venenoso = false y hábitat = “selva”, los demás valores los recibirá como parámetros.

Nota1: note que, en las clases Zoológico, y Zona se creo un atributo adicional llamado zonas y zoo estos atributos no se ponen explícitamente dentro del diagrama normalmente, se sobre entiende de las relaciones que hay entre dos clases, esto quiere decir que si hay una relación entre X y Y, existirá un atributo para X de tipo Y, y en Y uno de tipo X, y este será un listado dependiendo de la cardinalidad (0..* o 1..*).
Nota2: los atributos que están definidos como private debe crear sus métodos get y set.

Para pensar…

1. ¿Por qué es útil la herencia?
2. ¿De qué manera puede hacer que una lista almacene cualquier valor o objeto, como las listas en Python?