Metería:	Desarrollo Móvil Multiplataforma	Unidad:	1. Marco de referencia para DAM
Tarea:	Ejercicio 1 Lenguaje Dart	Grupo:	GDS0341
Alumno:	Cristian Israel Buclon Pedroza		

Transcribir el siguiente segmento de código:

```
1  void main() {
2    //Se crea un objeto de la calse Heroe
3    final wolverine = new Heroe(nombre: 'Logan', poder: 'Regeneración');
4    print(wolverine);
5  }
6
7  class Heroe {
8    String nombre;
9    String poder;
10
11   Heroe({required this.nombre, required this.poder});
12 }
```

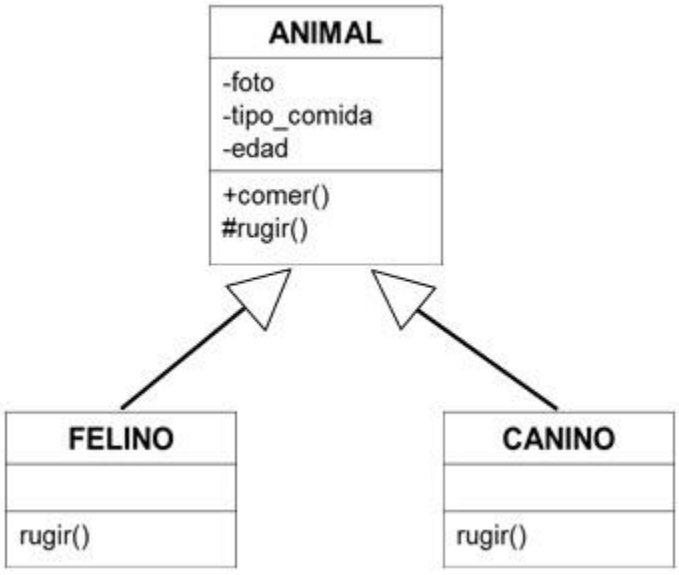
Puedes contestar aquí mismo las preguntas correspondientes al trabajo con el lenguaje Dart.

1. ¿Cuál es la salida del código especificado? **La salida que dicta la consola es: Instance of 'Heroe'**
2. ¿En que línea de código se implementa el constructor con parámetros por nombre? **La línea que implementa el constructor es la línea 3**
3. ¿Para qué sirve la expresión **required** en el constructor? **Para indicar que todo aquel parámetro que tenga esa extensión sea requerida u obligatoria**
4. ¿Qué indican las llaves en el constructor? **Indica la necesidad del atributo que este contiene**
5. ¿Qué difiere de acuerdo al siguiente? **Una de las diferencias más notables es que en la clase indica como es que el atributo de la línea 10 "poder" tiene el símbolo "?" eso quiere decir que este es opcional, de igual forma en la línea 12 falta el required del constructor Heroe**

```
1   void main() {
2     //Se crea un objeto de la calse Heroe
3     final wolverine = new Heroe(nombre: 'Logan');
4     print(wolverine.nombre);
5     print(wolverine.poder);
6   }
7
8   class Heroe {
9     String nombre;
10    String? poder;
11
12    Heroe({required this.nombre, this.poder});
13  }
14
```

6. ¿Para qué sirve entonces el símbolo de interrogación en la línea 10? **Sirve mas que nada para aceptar atributos opcionales o bien nulos**
7. Modifica el código de tal manera que el método t**oString** imprima la información, correspondiente al objeto.

Investiga lo necesario para poder implementar en código el siguiente diagrama de clases, en donde se representan relaciones de herencia.



Agrega el código resultante, para desarrollo de los métodos puedes utilizar mensajes que simulen la acción requerida para cada método.

Lo que interesa saber como entiendes este diagrama y como lo puedes implementa con el lenguaje Dart.

```dart
void main() {
  final felino= new Felino();
  final canino= new Canino();
  felino.foto='gato';
  felino.tipo_comida='Sobres';
  felino.edad= 5;
  canino.foto='Lobo';
  canino.tipo_comida='Carne';
  canino.edad=4;
  print("Datos del Felino");
  print('Foto: ${felino.foto}, Comida: ${felino.tipo_comida} Edad: ${felino.edad}' );
  felino.comer();
  felino.rugir();
  print("Datos del Canino");
  print('Foto: ${canino.foto}, Comida: ${canino.tipo_comida} Edad: ${canino.edad}' );
  canino.comer();
  canino.rugir();
}

class Animal{
String? foto;
String? tipo_comida;
int? edad;
  void comer(){print('Estoy comiendo');}
  void rugir(){print('Estoy rugiendo');}
}

class Felino extends Animal{
 @override
  void rugir(){
    print('soy un gato y estoy rugiendo');
  }
}
class Canino extends Animal{

 @override
  void rugir(){
    print('soy un perro y estoy rugiendo');
```