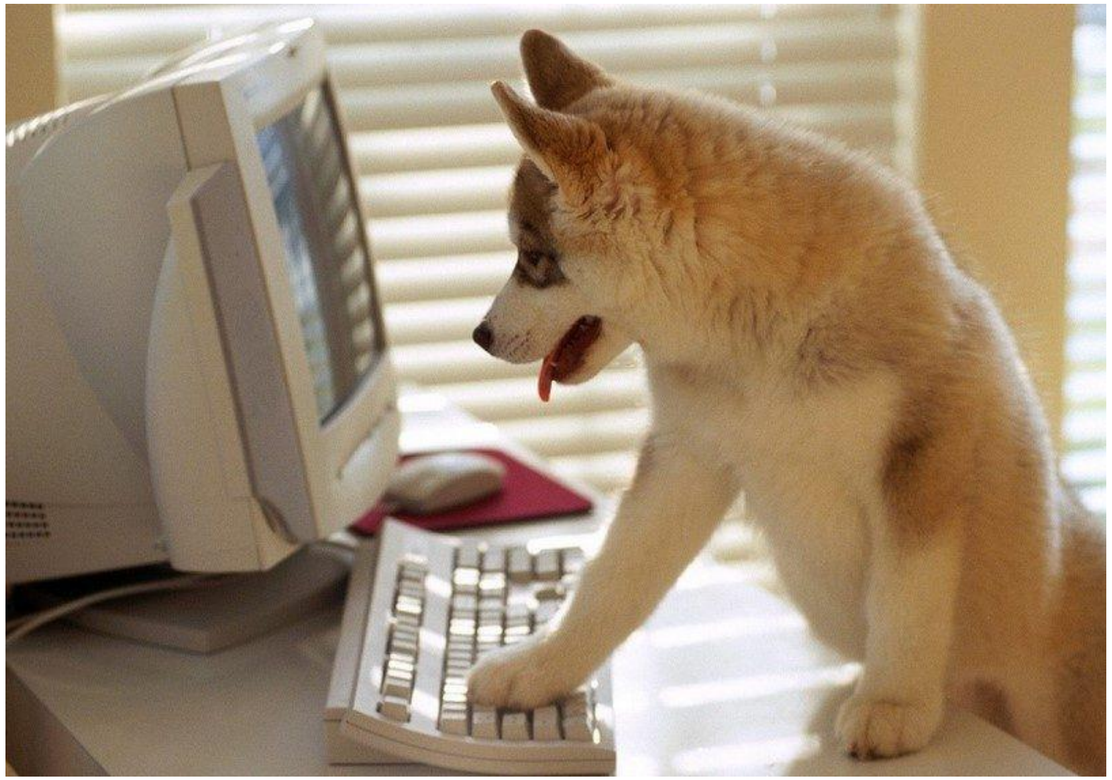
Java Programming



Organizational Stuff

23.09.: No Students :(
24.09.: Structures
25.09.: Methods & Recursion
26.09.: Arrays
27.09.: Strings

30.09.: OOP
01.10.: Generics & Linked Lists
02.10.: Exceptions & Testing
03.10.: Holiday
04.10.: GUI

OOP1

Do you remember `public static`? Today comes the explanation ;)

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

Attributes

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

Attributes

Constructor

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

Attributes

Constructor

Methods/Behaviour

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

Overloading: Same name, different parameters.

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

How do we create a *Book*?

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

```
Public static void main(String[] args){
   Book harryPotter = new Book("Harry Potter");
   harryPotter.setIsbn(1234567);
}
```

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

```
Public static void main(String[] args){
   Book harryPotter = new Book("Harry Potter");
   harryPotter.setIsbn(1234567);
}
```

What about the author?

Classes

```
public class Book{
   private String title;
   private int isbn;
   private Author author;
   public Book(String title){
      this.title = title;
   }
   public Book(String title, Author author){
      This.title = title;
      This.author = author;
   }
   public void setIsbn(int isbn){
      This.isbn = isbn;
   }
   public void getIsbn(){
      Return isbn;
   }
}
```

```
Public static void main(String[] args){
   Book harryPotter = new Book("Harry Potter");
   harryPotter.setIsbn(1234567);

   Author author = new Author("J.R.R. Tolkien");
   Book lotr = new Book("Lord of the Rings",
                        author);

}
```

Classes

`public` : visible to everyone

`private` : visible just in the class

`protected` : visible just in class and subclasses

Classes

```
Book b = new Book(“Some awesome title”);
b.setIsbn(9876543);

Book.read();
```

non-static

static

Abstract

```
public abstract class Dog{

   protected String name;

   public abstract void bark();

}
```

Abstract

```
public abstract class Dog{

   protected String name;

   public abstract void bark();

}
```

`abstract` : Declaration without implementation

Abstract

```
public abstract class Dog{

   protected String name;

   public abstract void bark();

}
```

An abstract class is a class with at least one abstract method.

Inheritance

```
public abstract class Dog{

   protected String name;

   public abstract void bark();

}
```

```
public class Bulldog extends Dog{

   public void bark(){
      System.out.println(“Woof!”);
   }

}
```

Interface

```
public interface iWalkable{

   void walk();

}
```

An Interface is like a service, not like a relation.
It contains just constants and abstract methods (no keyword abstract needed)

Interface

```
public interface iWalkable{

   void walk();

}
```

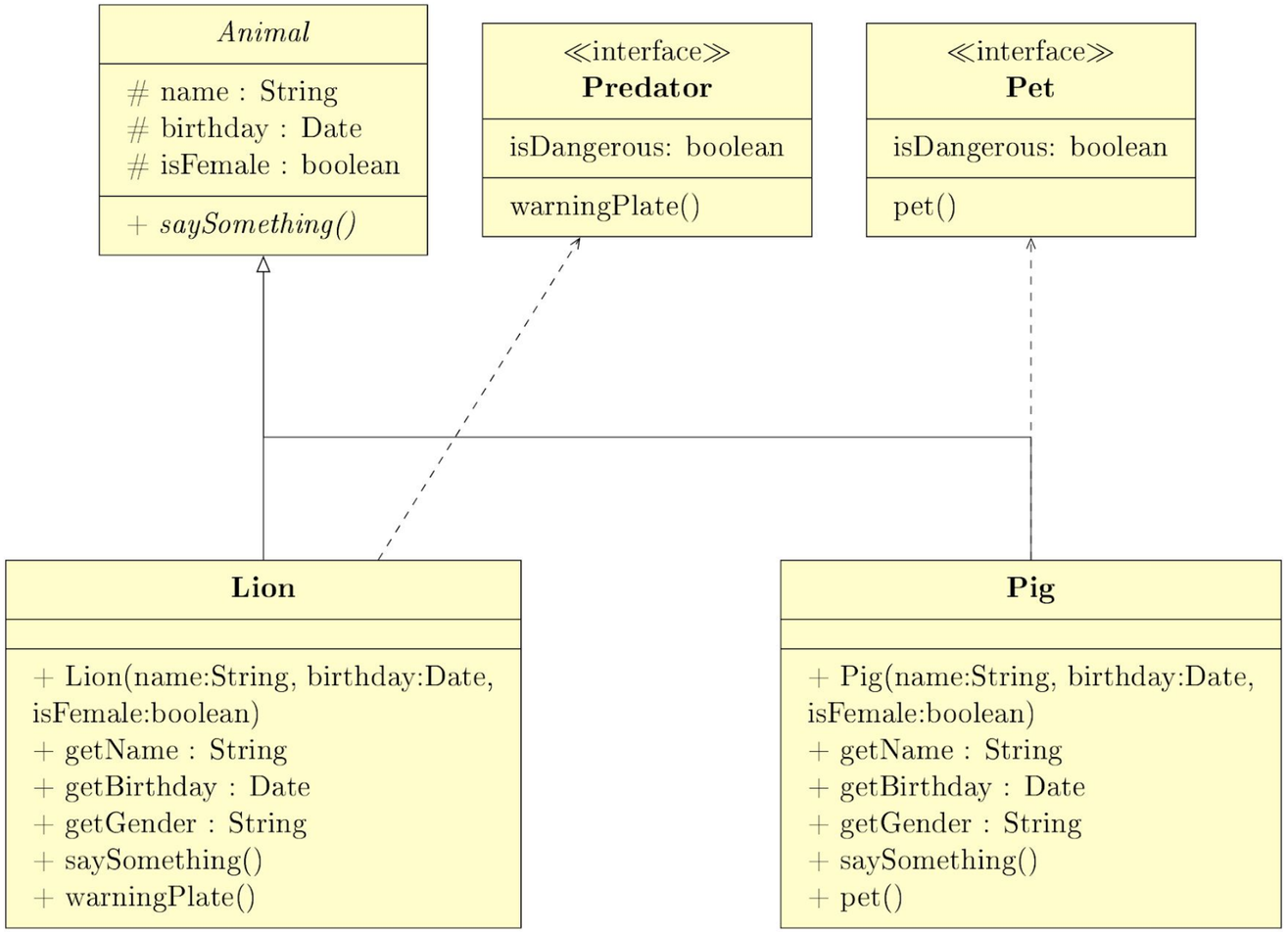
```
public class Bulldog extends Dog implements iWalkable{

   public void bark(){
      System.out.println(“Woof!”);
   }

   public void walk(){
      System.out.println(“I’m walking like a Bulldog”);
   }

}
```

Class Diagram



OOP

Today's Assignment:
https://classroom.github.com/a/K8kRLcr-