Object Oriented Programming

Basic, Encapsulation and Data Abstraction

" Programming paradigm which provides a means of structuring program so that **properties** and **behaviour** are bundled into individual **objects**. „

Properties

- Color
- Length
- Width

Behavior

- Accelerate
- Break

Properties

- Fur
- Leg
- Tail

Behavior

- Make Sound
- Eat
- Jump

" **Properties** determined by the values of its attributes "

" **Behavior** determined by how the objects acts or reacts to requests "

OOP Fundamental Concept

Data Abstraction

Encapsulation

Information Hiding

Woof!

Meow!

Quack!

Encapsulation

Basic encapsulation analogy:

- Class

- Attributes

- Method

Encapsulation - Class

Class is a *"template"* or *"blueprint"* that is used to create object.
"Special code" template in Java to make object:

- Contain of:
  - Properties
  - Method
- Has an init method to initiate object

```java
public class Cat { // Define class name using CamelCase
    private String name;
    private String color;
}
```

Make instance of Object

<div class="grid grid-cols-2 gap-4"> <div>

```java
public class Cat {
    private String name;
    private String color;

    // Constructor block
    public Cat(String name, String color) {
        this.name = name;
        this.color = color;
    }

    // Setter getter method
}

public static void main(String[] args) {
    Cat cat = new Cat("Peter", "White");
    cat.getName(); // Peter
```

Attributes Type

<div class="grid grid-cols-2 gap-4"> <div>

```java
public class Cat {
    private String name;
    private String color;

    // Constructor block
    public Cat(String name, String color) {
        this.name = name;
        this.color = color;
    }

    // Setter getter method
}
```

</div> <div>

# Method and Function

<div class="grid grid-cols-2 gap-4"> <div>

```java
public class Cat {
    private String name;
    private String color;

    public Cat(String name, String color) {
        this.name = name;
        this.color = color;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
```

Data Abstraction

"Hiding background process from user"

Main Goal:

- Handle complexity by hiding unnecessary detail

- It should only reveal operations relevant for the other objects
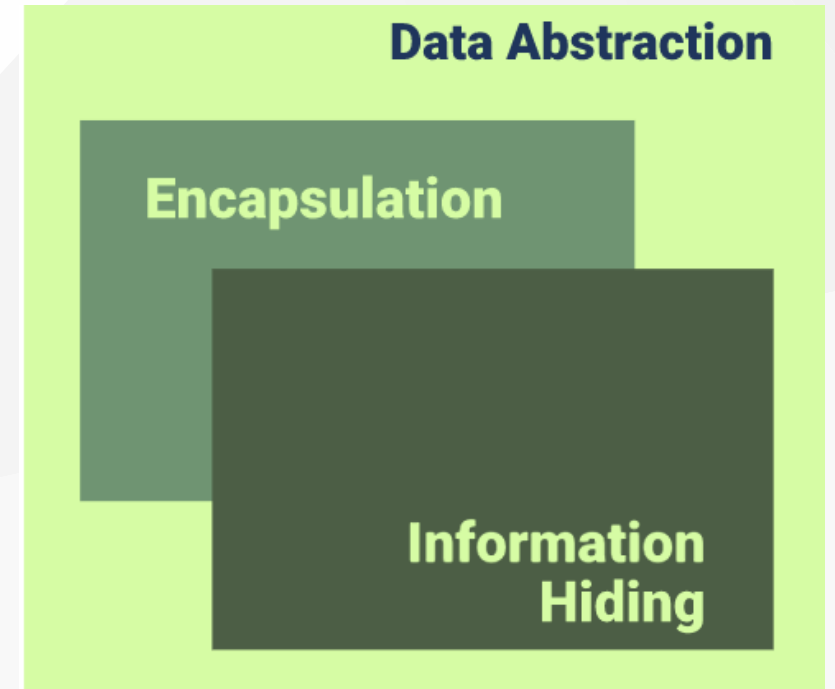
Data Abstraction **Analogy**

Data Abstraction - Setter Getter

<div class="grid grid-cols-2 gap-4"> <div>

```java
public interface Motorcycle {
    void startEngine();
}


public class Vario implements Motorcycle {
    @Override
    public void startEngine() {
        System.out.println("Use electric starter!");
    }
}


public class RXKing implements Motorcycle {
    @Override
    public void startEngine() {
        System.out.println("Use kick starter!");
```

Task

Define 5 classes freely related to the type of animal, plant or vehicle. Use encapsulation concepts such as public, protected and private according to analogy examples in the real world.

Example:
`Cat, Fish, Flower, Car, etc.`

Add instance variables and methods in each class created. Then create code to prove **encapsulation** is running as expected.

For example, can `Frog` access these `public`, `protected` or `private` variables? or other things that produce returns as expected.