OOPS

Class is a template for objects, and an

Object is an instance of a class.

In Java , in general everything is an object

Graphical user interface, website

Description automatically generated



Constructor

Modifier

Diagram

Description automatically generated

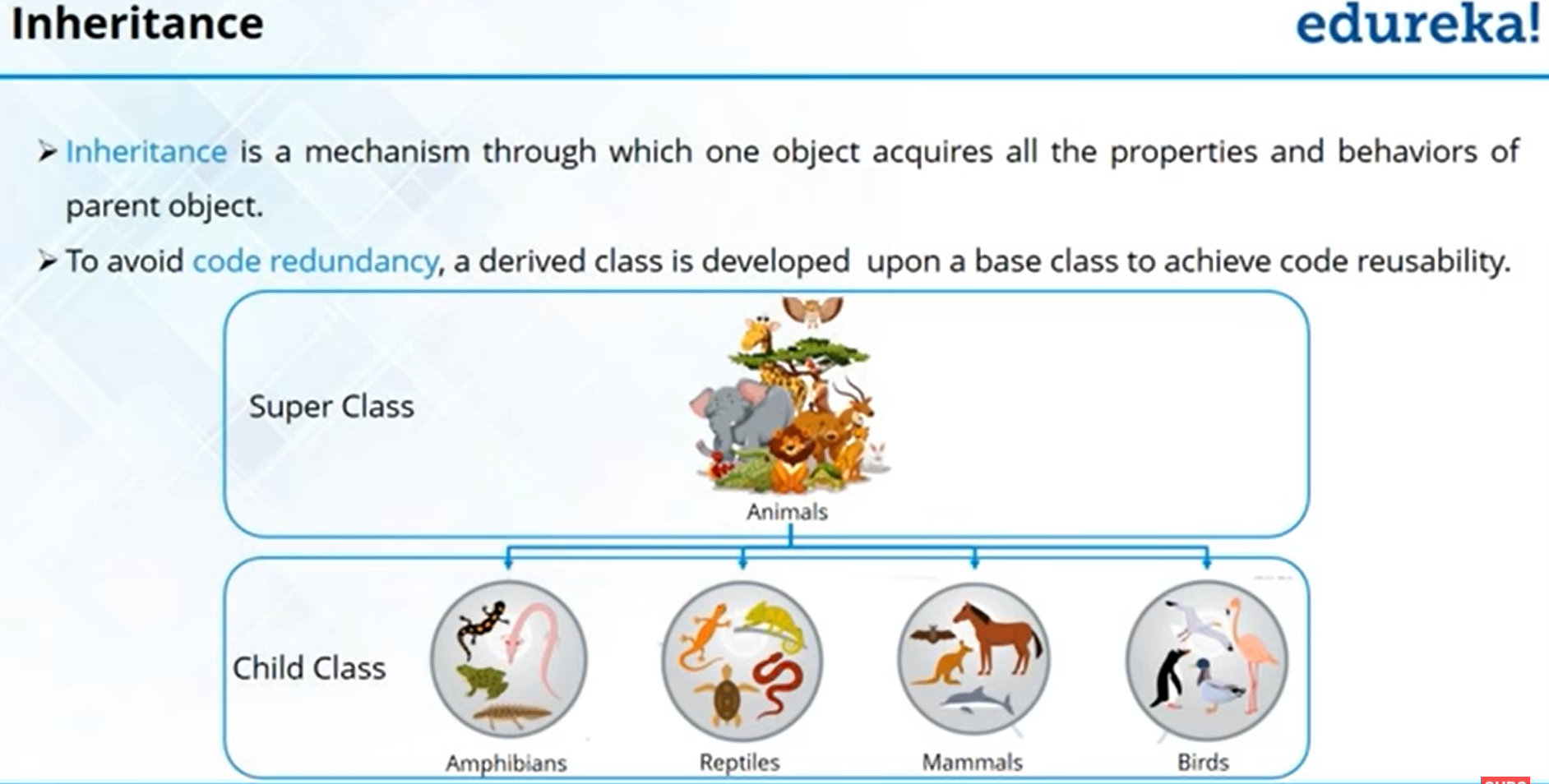
For example car is a Class - audi, volvo, benz are objects created from the class.

Example:

public static void SubscribeEmailForInvalidData(String name, String email) throws AWTException, InterruptedException {  
 *driver*.get(Utils.*BASE\_URL*);  
 HomePageClass homePageClass = new HomePageClass(*driver*);  
 homePageClass.SubscribeEmailForInvalidData(name, email);

Graphical user interface

Description automatically generated





example

public class HomePageClass extends PageObject {

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated



Examples

<https://www.programiz.com/java-programming/polymorphism>

add(a, b)

{

C= a+b

}

add(a,b,c,d)

{

Z= a+b;

}

Graphical user interface, diagram, application

Description automatically generated

Diagram

Description automatically generated



**Java Encapsulation**

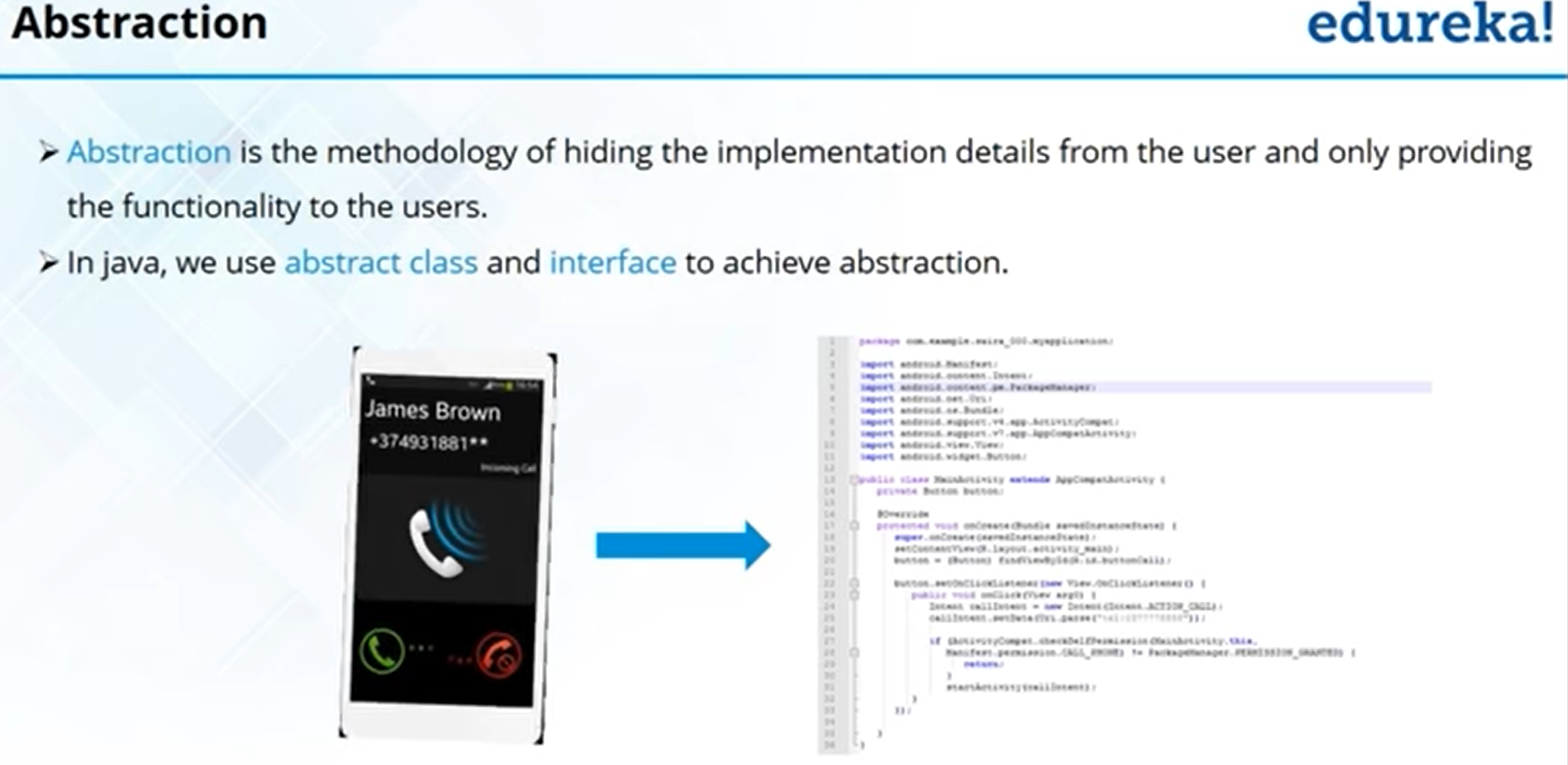
Encapsulation is one of the key features of object-oriented programming. Encapsulation refers to the bundling of fields and methods inside a single class.

It prevents outer classes from accessing and changing fields and methods of a class. This also helps to achieve **data hiding**.

https://www.programiz.com/java-programming/encapsulation

A picture containing text, businesscard, screenshot

Description automatically generated





Example:

abstract class Animal {

abstract void makeSound();

public void eat() {

System.out.println("I can eat.");

}

}

class Dog extends Animal {

// provide implementation of abstract method

public void makeSound() {

System.out.println("Bark bark");

}

}

class Main {

public static void main(String[] args) {

// create an object of Dog class

Dog d1 = new Dog();

d1.makeSound();

d1.eat();

}

}

In the above example, we have created an abstract class Animal. The class contains an abstract method makeSound() and a non-abstract method eat().

We have inherited a subclass Dog from the superclass Animal. Here, the subclass Dog provides the implementation for the abstract method makeSound().

We then used the object d1 of the Dog class to call methods makeSound() and eat().

https://www.programiz.com/java-programming/abstract-classes-methods

<https://www.freecodecamp.org/news/java-object-oriented-programming-system-principles-oops-concepts-for-beginners/>

https://www.guru99.com/java-oops-concept.html

https://www.edureka.co/blog/object-oriented-programming/

Java Basics

<https://www.youtube.com/watch?v=aqHhpahguVY>