Interface & Abstract : -

دا عباره عن كلاس بس مختلف شويه عن الكلاسيز العاديه بتاعتنا
هو كلاس مينفعش اعمل منه اوبجيكتس و فكرته ان انا اقدر اعمل فيه ميثودز من غير امبليمنتيشن واورث الكلاس دا للسبكلاسيز بحيث ان كل واحد منهم لازم يعمل امبلمنتيشن للميثودز دي بالطريقه اللي هو عايزها (override) .

ممكن الاقي فيه :
ميثودز عاديه بالامبليمنتيشن بتاعتها concrete methods
ميثودز منغير امبليمنتيشن abstract methods
كونستراكتورز
اتربيوتس
فاينال & استاتك 4ميثودز

اتفقنا ان انا بورثه للكلاسيز التانيه من خلال ال extends keyword
لازم الابستراكت ميثودز تبقي جوا كلاس ابستراكت
لو مديني ال uml بتاع الكلاس اما بيبقي قايلي انه ابستراكت كلاس او بيبقي الفونت بتاع الاسم *italic*

📢📢📢📢

ينفع يبقي في حاجه اسمها ابستراكت فاريابل ؟ ❌
ينفع يبقي في حاجه اسمها ابستراكت كونستراكتور ؟ ❌
ينفع تيجي مع ميثودز بروتيكتيد ؟ ✔️
طب مع برايفت ؟❌

طب اي لازم الكونستراكتورز بقا ادام انا مش هعرف اعمل منها اوبجيكتس ؟؟
احنا قولنا ان عندنا اتربيوتس عادي ف لو عايزه اديهم فاليوز قدامي طريقتين
اما بالكونستراكتور او بال setter & gettars
فكل الفكره ان انا هعمل كونستراكتور ف الابستراكت كلاس واخليه يسند قيم عادي وابقي اعمله call في السبكلاسيز عادي بدل م اقعد في كل كلاس استدعيهم من الاول .

نشوف مثال بسرعه 🙂

```
public abstract class Animal {

    public static int animalCount = 0;
    protected String name;
    protected int age;
    public final String type = "Mammal";

    public Animal(String name, int age) {
     this.name = name;
     this.age = age;
     animalCount++;
     System.out.println("Animal constructor called for: " + name);
    }

    public abstract void makeSound();

    public void sleep() {
        System.out.println(name + " is sleeping...");
    }

    // Final method (cannot be overridden)
    public final void showType() {
        System.out.println(name + " is a " + type);
    }

    // Static method (اقدر استدعيها بدون وساطه اوبجيكت )
    public static void showAnimalCount() {
        System.out.println("Total animals created: " + animalCount);
    }
}
```

```java
// Concrete subclass
public class Dog extends Animal {
    public Dog(String name, int age) {
        super(name, age); // call constructor of Animal
    }

    // Implementation of abstract method
    @Override
    public void makeSound() {
        System.out.println(name + " says Woof!");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Dog d1 = new Dog("Buddy", 3);
        d1.makeSound();          // Buddy says Woof!
        d1.sleep();              // Buddy is sleeping...
        d1.showType();           // Buddy is a Mammal

        Dog d2 = new Dog("Rocky", 2);
        d2.makeSound();          // Rocky says Woof!

        // Call static method from the abstract class
        Animal.showAnimalCount();  // Total animals created: 2
    }
}
```

```
Animal constructor called for: Buddy
Buddy says Woof!
Buddy is sleeping...
Buddy is a Mammal
Animal constructor called for: Rocky
Rocky says Woof!
Total animals created: 2
```

طب اي اللي يحصل لو معملتتش امبليمنتيشن لكل الميثودز اللي ف الابستراكت كلاس ؟؟
Erroooooooooor

طب والحل ؟؟
ان انا مثلا لو عملتت امبليمنتيشن لواحده منهم
اكتب الباقي منغير امبليمنتيشن عادي واحلي الكلاس دا نفسه ابستراكت ولعد كدا اقور اورثه واكمل عليه

بمعني تاني اخليه هو نفسه ابستراكت كلاس واحول الميثود تاللي عملتلها امبليمنتيشن ل كونكريت ميثود

```java
package com.mycompany.javaoop;

public abstract class SUVAli extends Car {
    @Override
    void autopilot(){
        System.out.println("SUVAli");
    }

    abstract void streamingServices();

    abstract void parkingSensors();
}
```

Interface : -

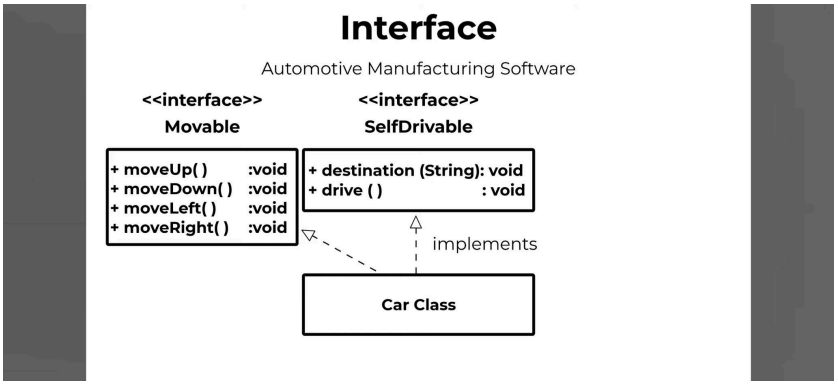
اول معلومه انها مش زي واجهه المستتخدم ولا gui ولا ui /ux

هي شبه الابستراكت كدا (عباره عن كلاس فيه ميثودز برضو منغير امبليمنتيشن واحنا هنعملهم امبليمنتيشن في السبكلاسيز)

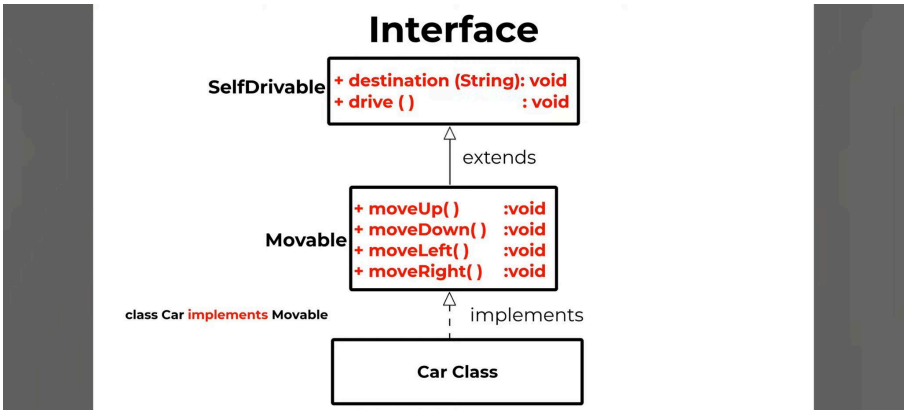
```
public interface InterfaceName{
    void method1();
    void method2();
    void method3();
}
public class ClassName implements InterfaceName{
    @Override
    void method1() {..}

    @Override
    void method2(){..}

    @Override
    void method3(){..}
}
```

, مع وجود شويه اختلافات :

- اسم الكلاس بيبقي اخره able او بيبدأ ب can وبيبقي كاتب ان الكلاس دا <<interface>>
- الكلاس نفسه بكتب قبله كلمه interface وانا بعرفه
- الوراثه بتتم عن طريق كلمه implements وممكن الكلاس الواحد يعمل امبليمنتيشن لاكتر من انترفيس



وممكن كمان انترفيس تورث extends من انترفيس تانيه



وكالعادي لو كلاس عامل امبليمنت لانترفيس والانترفيس وارثه من اكتر من انترفيس هيبقي مسموحله يتعامل مع كل الميثودز الموجوده

- الكلاسيز والابستراكت كلاسيز ممكن عادي جدا يعملوا امبليمنتيشن للانترفيسيز .

ملحوظه 👍
وانا بعرف الانترفيس بعرف فاريابلز وميثودز عادي
الفاريابل بيقي public static final
وال ميثودز بتبقي public abstract
ودا بشكل تلقائي .
ناخد مثال :

```java
public interface Animal {

    final String TYPE = "Living Being";

    void makeSound();

    default void sleep() {
        System.out.println("Animal is sleeping...");
    }

    static void info() {
        System.out.println("This is the Animal interface.");
    }
}
```

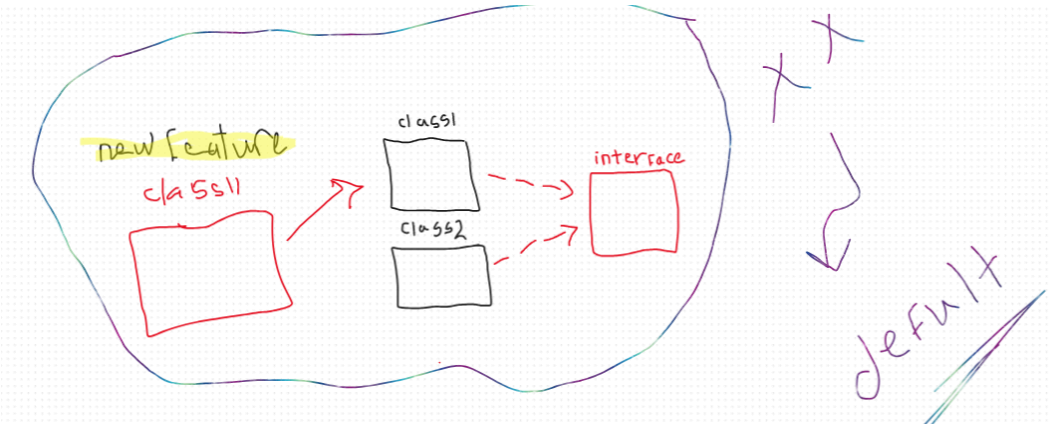
```java
public class Dog implements Animal {
    private String name;

    public Dog(String name) {
        this.name = name;
    }

    // لازم تنفذ الميثود makeSound
    @Override
    public void makeSound() {
        System.out.println(name + " says Woof!");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Dog d = new Dog("Buddy");
        d.makeSound();           // Buddy says Woof!
        d.sleep();               // Animal is sleeping...
        Animal.info();           // This is the Animal interface.
        System.out.println("Type: " + Animal.TYPE); // Type: Living Being
    }
}
```

اي بقا قصه ال default دي ؟

مبديا احنا عارفين ان الانترفيس لحد سنه 2014 كان عباره عن 100% ابستراكت يعني مينفعش يتكتب جواه concrate methods

نحكي قصه 👍.....



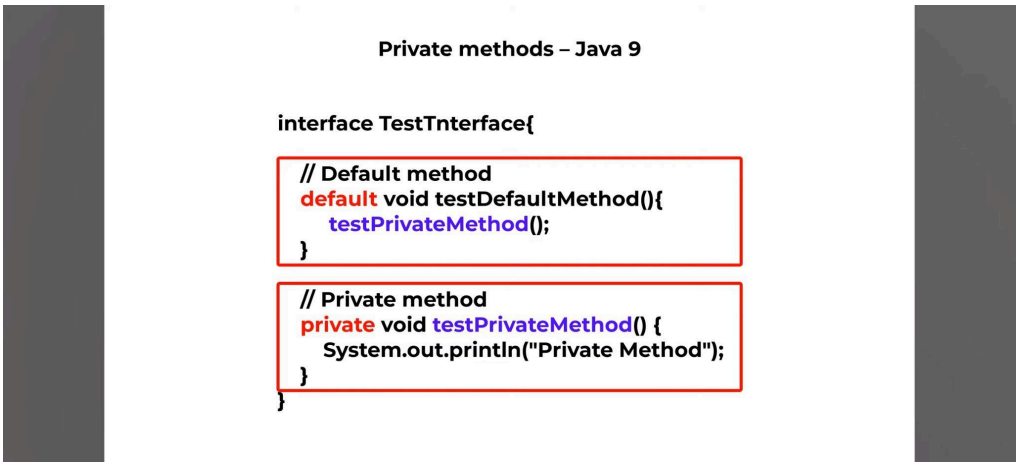
الdefault method معناها ان انا بقول للانترفيس انت ممكن يبقي جواكي ميثود وليها امبليمنتيشن في اي وقت اقدر استخدمها وبيبقي الديفولت بتاعها انها public
اذن اي ميثود جمبها الكيوورد default لازم هيبقي ليها امبليمنتيشن واقدر اعملها اوفررايد والاكسيز موديفير بتاعها public

ملحوظه
احنا عارفين ان ال multible enheritance مش موجود ف جافا غير ف حاله الانترفيس بس ف ..
لو عندي كلاس عامل امبليمنتيشن ل 2 انترفيس وهما الاتنين فيهم نفس الميثود بنفس السيجنيتشر بنفس الامبليمنتيشن . نتوقع عادي كدا ؟؟ اكييد لا هيجيب ايرور وبالتالي لازم اغير الاسماء او اخلي واحده منهم ابستراكت ميثود واعملها امبليمنتيشن

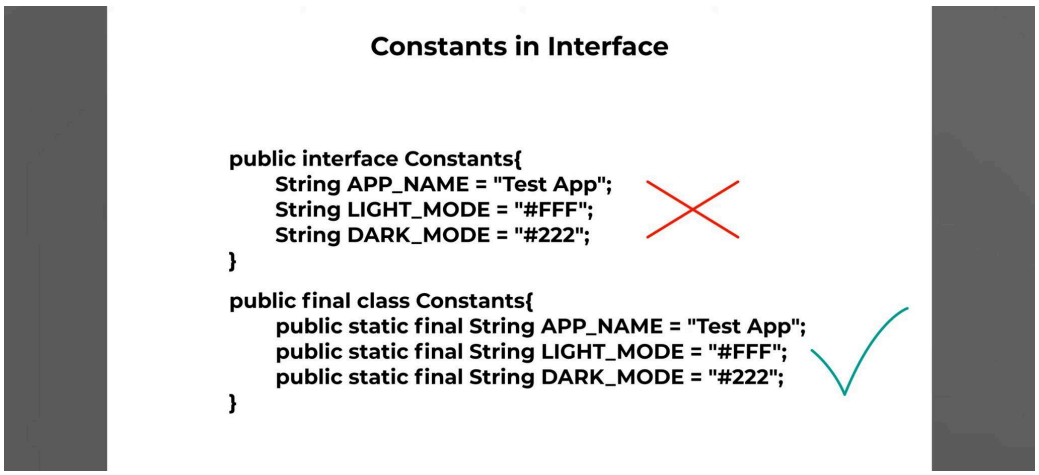
طب اي بقا ال static method دي ؟؟
دي ميثود كامله بالامبليمنتيشن بتاعها

طب اي ال private method ؟؟
غريبه دي انا اصلا عامله الانترفيس علشان اورث منه ف البرايفت متخلنيش اورثها ف اي اهميتها ؟؟

ميثود عاديه خالص ب امبليمنتيشن عادي بس ممكن استدعيها في ال default method بتاعتي

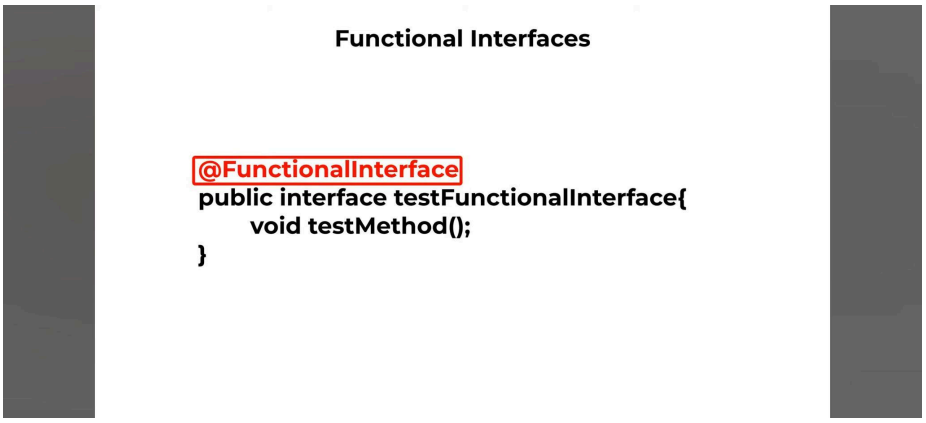


وبالمناسبه احنا قولنا ان الفاريابلز في ال interface == > public static final
يعني اكنها كونستانت ثابت كدا
ف منقولش بقا في ثوابت وفي ميثودز برايفت واستخدم الانترفيس ك انها مكان مخزنه فيه ثوابت دا كدا استخدام غلك الصح اعملهم كلاس واخلي الكلاس نفسه final

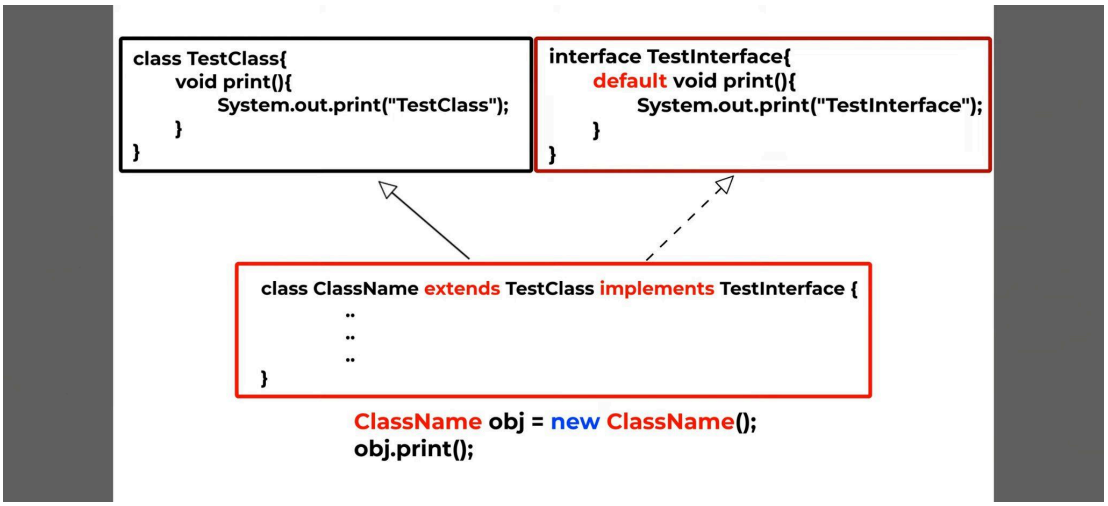


ملحوظه :

ال functional interface مفيهاش غير ميثود واحده بس



- ماذا لو عندنا 2 ميثود بنفس كل حاجه ماعدا الامبليمنتيشن واحده ف انترفيس وواحده ف كلاس وانا عامله كلاس تالت وارث منهم هما الاتنين انهي واحده اللي هتتنفذ ؟ اللي ف الكلاس لانه هنا هو اللي ليه الاولويه



طب ممكن اعمل nested interface ؟؟
ايوا عادي جدا نشوف مثال

```java
public interface Animal {

    String TYPE = "Living Being";

    void makeSound();

    default void sleep() {
        System.out.println("Animal is sleeping...");
    }

    static void info() {
        System.out.println("This is the Animal interface.");
    }

    interface Movement {        // nested interface
        void move();
    }
}
```

```java
public class BirdMovement implements Animal.Movement {
    @Override
    public void move() {
        System.out.println("Bird is flying.");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog("Buddy");
        dog.makeSound();        // Buddy says Woof!
        dog.sleep();            // Animal is sleeping...
        Animal.info();          // This is the Animal interface.
        System.out.println("Type: " + Animal.TYPE); // Type: Living Being

        // استخدام الواجهة المتداخلة
        Animal.Movement bird = new BirdMovement();
        bird.move();            // Bird is flying.
    }
}
```

كل الفكره ان انا ونا بتعامل معاها هكتب outer interface .innerinterface

اخر حاجه هنتكلم عنها ...

- Marker or tagging interface ⇒ empty

بستخدمه لو عايزه اعمل اذن معين او حاجه معينه لو الاوبجيكت دا ليه الاكسيز علي الميثود دي من خلال ال instance of

Object instance of class

- Generic interface

بيحدد الdata type بتاع البراميتيرز اللي هنبعتها للميثودز زي كدا

```
public class StringContainer implements Container<String> {
    private String item;

    public void add(String item) {
        this.item = item;
    }

    public String get() {
        return item;
    }
}
```