Java for beginners

Interface

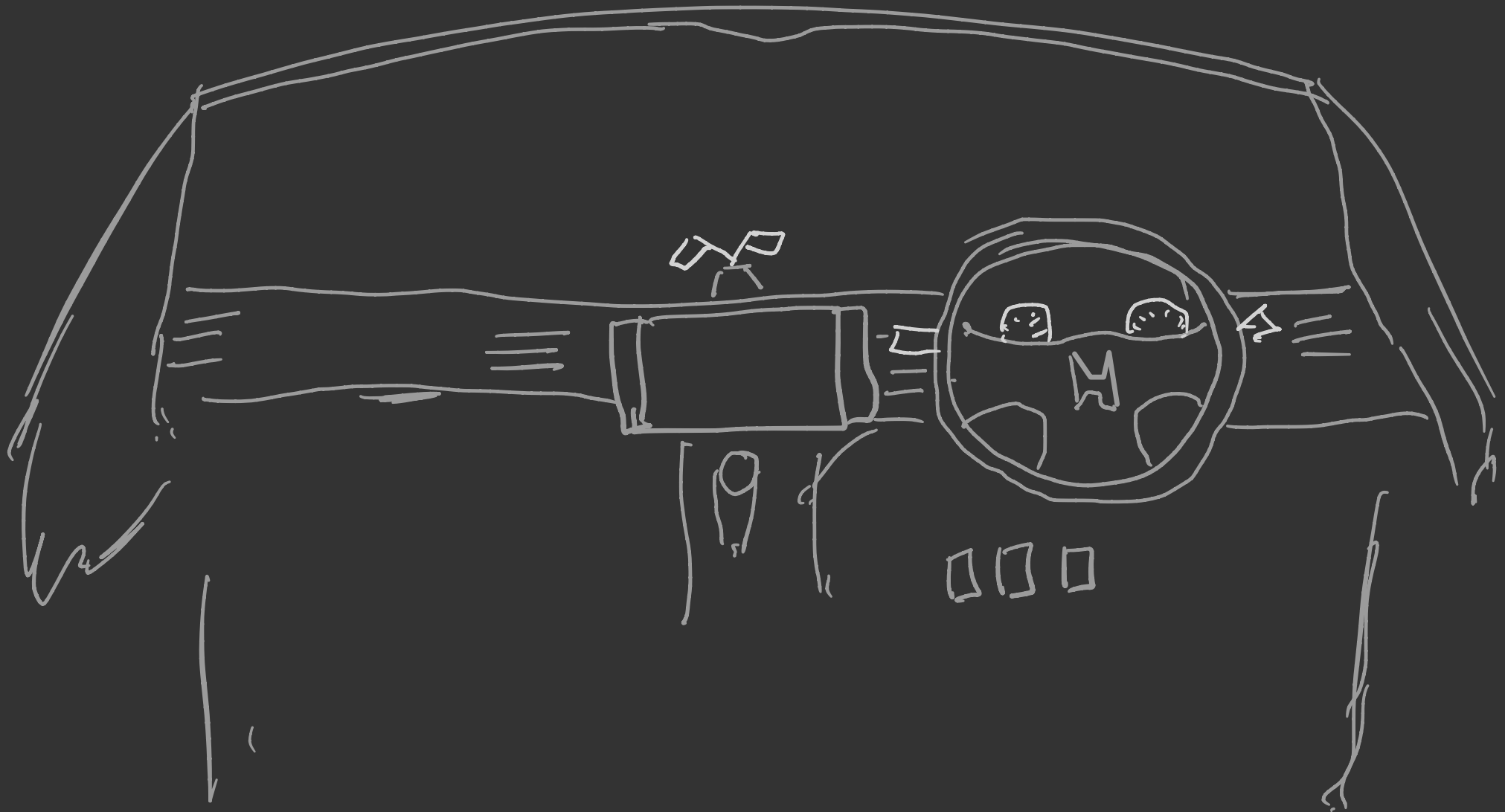
Saurabh Shukla (MySirG)

Agenda

1. Interface meaning
2. Interface Syntax
3. About Interface
4. Defining Interface
5. How to use an interface
6. Why interface is required?
7. Polymorphism
8. Multiple implementations & extensions
9. Abstract class vs Interface
10. New Features

Interface

- Face to interact with.



Interface Syntax

Interface definition begins with a keyword interface.

```
interface InterfaceName {

}
```

About Interface

An interface like that of an abstract class cannot be instantiated.

Although you can create a reference variable of interface.

Interface do not have constructors.

Interface just specify the method declaration (implicitly public and abstract) and can only contain fields (implicitly public, static and final)

Defining an interface

```
interface Area {
        double PI = 3.14;
        double getArea();
}
```

How to use an interface?

```
class Circle implements Area {
    private double radius;
    public void setRadius ( double radius) {
        this.radius = radius;
    }
    public double getRadius() {
            return radius;
    }
    public double getArea() {
            return PI * radius * radius;
    }
}
```

Reference variable of interface can refer to an object of class implementing that interface.

```
Area a1 = new Circle();
a1.setRadius(5);                    Error
double x = a1.getRadius();          Error
double y = a1.getArea();            Correct
```

a1 is a reference variable of type Area, it can refer to an object of Circle but can only access those members of Circle which belongs to interface Area

If a class that implements an interface does not define all the methods of the interface, then it must be declared abstract and the method definitions must be provided by the subclass that extends the abstract class.

Why interface is required?

Interfaces are used in java to achieve abstraction.

It is also helpful in capturing similarities among unrelated classes without artificially forcing a class relationship.

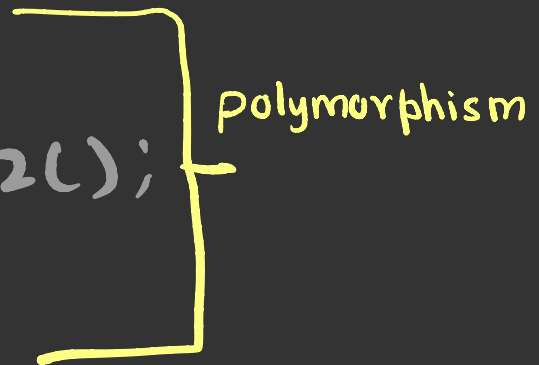
Polymorphism

```
interface I1 {
        void f1();
}

class A1 implements I1 {
    ...
    public void f1() { ... }
}

class A2 implements I1 {
    ...
    public void f1() { ... }
}
```

```
I1 obj = new A1();
obj.f1();
obj = new A2();
obj.f1();
```



```
void fun( I1 x )
{
    x.f();
}
```

```
interface I1 {
    void f();
}
```

```
class A1 implements I1 {
    public void f(){
    }
}
```

```
fun(new A1());
```

```
interface ActionListener {

    void actionPerformed(ActionEvent);

}

class JButton {
public void addActionListener(ActionListener x1)
    {

    x1.actionPerformed(event);
      =
    }

}
```

```
class A implements
      ActionListener {

public void actionPerformed(E..)
{
  =
}

}
```

```
class MyButton extends JButton
      implements ActionListener;

}

A obj = new A();

MyButton b1 = new MyButton();
b1.addActionListener(obj);
```

Multiple Implementations

A class can extend only one class but can implements any number of interfaces.

```
class B extends A implements I1, I2, I3 {

        ==

}
```

More Examples

- Car wants to implement MusicPlayer
- SportsWatch also wants to implement MusicPlayer
- Car also wants to implement Metrics to show various driving metrics
- Batch wants to implement Admission
- Television wants to implement MagicStick to become a SmartTV.

Multiple extension

Multiple extension is allowed when extending interfaces, i.e. one interface can extend none, one or more interfaces.

```
interface I1 {
  —
}
interface I2 {
  —
}
```

```
interface I3 extends I1, I2 {
  =
}
```

Remember

- class extends class
- class implements interface
- interface extends interface

Abstract Class vs Interface

1) Abstract class will have constructor but Interfaces do not have constructors

2) Abstract class can have abstract and non abstract methods, but interfaces can have only abstract methods (surprise is waiting for you)

3) All members of an interface are public, but in Abstract class you can set any access modifier.

4) Interface can have only static variables, but Abstract class can have static and instance variables.

5) Interface can have only final variables, but Astract class can have non final variables as well.

New Features

Before Java 8, interfaces can only have abstract methods, but from Java 8 onwards, you can write methods with definitions.

But then how interfaces are different from abstract classes?

There were some interfaces in Java (before Java 8), like List which were used heavily in programming projects.

From Java 8 onwards some of the interfaces like List were modified with extra abstract methods in it.

So if some project was running on Java 7 and we updated Java version from 7 to 8, suddenly project should stop working because, classes implemented List do have definitions for extra abstract methods

To solve this problem Oracle, instead of making methods abstract, they define them as default methods.

```
interface I1 {

    default void f1() {
        ==
    }

}
```

Default method.

A class cannot extends multiple classes but can implements multiple interfaces.

```
interface I1 {
    default void f1() {
        ....
    }
}
```

```
interface I2 {
    default void f1() {
        ....
    }
}
```

```
class A implements I1, I2 {

}
```

```
A obj = new A();
obj.f1();
```

which one will bind?

To solve this problem Java forces to define default method in class A.

Normally you need not to override default method of interface in the class.

Another Scenario

```
interface I1 {
    default void f1() {
    }
}
```

```
class A {
    public void f1() {
    }
}
```

```
class B extends A implements I1 {

}
```

```
B obj = new B();
obj.f1();
```

which function will execute

```
interface I1 {
    default void f1() {
    }
}
```

```
class A {
    public void f1() {
    }
}
```

```
class B extends A implements I1 {

}

B obj = new B();
obj.f1();
```

f1() of class A

Interfaces are not allowed to define default methods which are already available in Object class.

Static Methods in interfaces

```
interface I1 {
    static void f2(){
        ==
    }
}
```

```
I1.f2();
```

works

Conclusion

Java interfaces can have

- variables (public, static, final)
- methods (abstract & public)
- default methods
- static methods.