

JAVAGURU INTRODUCTION TO JAVA

LESSON 10

JAVA ABSTRACTION: INTERFACES OVERVIEW

- ▶ A bit like class, except:
 - ▶ Interface **can only contain** method **signatures** and **fields**
- ▶ Methods defined in interfaces **cannot contain** the implementation of method, **only** signature (return type, name, parameters, exceptions)
- ▶ **Describes** an object by actions it **can perform**
 - ▶ Sometimes interface names end with '**-able**' postfix (e.g. **comparable**)

1. JAVA ABSTRACTION: INTERFACE CODE EXAMPLE

Interface keyword
instead of class

Interface name

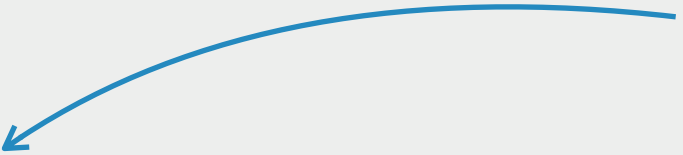
```
public interface Singer {  
    void sing();  
}
```

Singers can sing,
but we don't care
how they do so

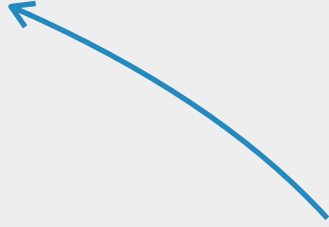
2. JAVA ABSTRACTION: INTERFACE CODE EXAMPLE

```
public class ElvisPresley implements Singer {  
  
    ...  
  
    @Override  
    public void sing() {  
        System.out.println("Love me tender, baby..");  
    }  
  
    ...  
}
```

Special keyword
to guarantee that
we support
interface specified
behaviour



Concrete implementation
of singers behaviour



3. JAVA ABSTRACTION: INTERFACE CODE EXAMPLE

```
public class MichaelJackson implements Singer {  
  
    ...  
  
    @Override  
    public void sing() {  
        System.out.println("Billie Jean is not my lover");  
    }  
  
    ...  
  
}
```

4. JAVA ABSTRACTION: INTERFACE CODE EXAMPLE

```
public class BritneySpears implements Singer {  
  
    ...  
  
    @Override  
    public void sing() {  
        System.out.println("Hit me baby one more time");  
    }  
  
    ...  
  
}
```

JAVA ABSTRACTION: ABSTRACT CLASS OVERVIEW

- ▶ Mostly like a class, except:
 - ▶ Can contain method signatures without implementation among other methods
 - ▶ Cannot be instantiated

1. JAVA ABSTRACTION: INTERFACE VS ABSTRACT CLASS

- ▶ Type of methods
 - ▶ Interface can have only **abstract** methods (since Java 8 supports static and default methods as well)
 - ▶ Abstract class can have **abstract** and **non-abstract** methods
- ▶ Final variables
 - ▶ Variables declared in a Java interface are by default **final**
 - ▶ Abstract class may contain **non-final** variables

2. JAVA ABSTRACTION: INTERFACE VS ABSTRACT CLASS

- ▶ Type of variables
 - ▶ Interface has only **static** and **final** variables
 - ▶ Abstract class can have **final**, **non-final**, **static** and **non-static** variables
- ▶ Implementation
 - ▶ Interface **can't provide** the implementation of abstract class
 - ▶ Abstract class **can provide** the implementation of interface

3. JAVA ABSTRACTION: INTERFACE VS ABSTRACT CLASS

▶ Inheritance vs Abstraction

- ▶ Interface can be **implemented** using keyword "implements"
- ▶ Abstract class can be **extended** using keyword "extends"

▶ Multiple Implementation

- ▶ Interface **can extend** another Java **interface only**
- ▶ Abstract class **can extend** another Java class and **implement multiple** Java interfaces

▶ Accessibility of data members

- ▶ Access modifiers of interface members are **public** by default and **cannot be changed**
- ▶ Access modifiers of abstract class members **can have any** access modifiers (except private abstract methods)

POLYMORPHISM

OVERVIEW

POLYMORPHISM OVERVIEW

- ▶ Polymorphism is the **ability** of an object to take on **many** forms
- ▶ Capability of a method **to do** different things based on the object that it is **acting upon**
- ▶ Which implementation to be used is **decided** at runtime **depending** upon the situation

1. POLYMORPHISM: CODE EXAMPLE

Code

```
Singer elvis = new ElvisPresley();  
Singer jackson = new MichaelJackson();  
Singer spears = new BritneySpears();  
  
elvis.sing(); jackson.sing(); spears.sing();
```

Console output

```
Love me tender, baby..  
Billie Jean is not my lover  
Hit me baby one more time
```

2. POLYMORPHISM: CODE EXAMPLE

Code

```
Singer[] singers = new Singer[2];  
singers[0] = new ElvisPresley(); singers[1] = new BritneySpears();  
  
for (Singer singer : singers) {  
    singer.sing();  
}
```

Console output

```
Love me tender, baby..  
Hit me baby one more time
```

3. POLYMORPHISM: CODE EXAMPLE

Code

```
Shape circle = new Circle("Red", 3);  
Shape rectangle = new Rectangle("Blue", 2, 4);  
  
System.out.println("Circle area = " + circle.area());  
System.out.println("Rectangle area = " + rectangle.area());
```

Console output

```
Circle area = 28.259999999999998  
Rectangle area = 8.0
```

REFERENCES

- ▶ <https://stackify.com/oops-concepts-in-java/>
- ▶ https://www.tutorialspoint.com/java/java_inheritance.htm
- ▶ <https://beginnersbook.com/2013/03/oops-in-java-encapsulation-inheritance-polymorphism-abstraction/>
- ▶ <https://www.geeksforgeeks.org/abstraction-in-java-2/>
- ▶ <http://tutorials.jenkov.com/java/interfaces.html>
- ▶ <https://docs.oracle.com/javase/tutorial/java/landl/subclasses.html>