

# Interfaces

"Defining, Implementing, and Applications"  
**Advanced Programming**

Shakirullah Waseeb  
shakir.waseeb@gmail.com

Nangarhar University

March 1, 2017



# Agenda

- 1 Introduction
- 2 Interfaces in Java
  - Interfaces in Java
  - Interface Definition
  - Interface Implementation
- 3 Applications of Interface
- 4 Questions and Discussion



# Introduction

- Abstraction of a class from its implementation



# Introduction

- Abstraction of a class from its implementation
- Specify what a class should do, but not how it does it



# Introduction

- Abstraction of a class from its implementation
- Specify what a class should do, but not how it does it
- Syntactically similar to classes, but lack instance variables, and methods are declared without definition



# Introduction

- Abstraction of a class from its implementation
- Specify what a class should do, but not how it does it
- Syntactically similar to classes, but lack instance variables, and methods are declared without definition
- Implementing class must create the complete set of methods defined by the interface



# Introduction

- Abstraction of a class from its implementation
- Specify what a class should do, but not how it does it
- Syntactically similar to classes, but lack instance variables, and methods are declared without definition
- Implementing class must create the complete set of methods defined by the interface
- However, each class is free to provide its own implementation



# Agenda

- 1 Introduction
- 2 Interfaces in Java
  - Interfaces in Java
  - Interface Definition
  - Interface Implementation
- 3 Applications of Interface
- 4 Questions and Discussion





# Interfaces in Java

- Java allows you to fully utilize the “one interface, multiple methods” aspect of polymorphism



# Interfaces in Java

- Java allows you to fully utilize the “one interface, multiple methods” aspect of polymorphism
- Designed to support dynamic method resolution at runtime



# Interfaces in Java

- Java allows you to fully utilize the “one interface, multiple methods” aspect of polymorphism
- Designed to support dynamic method resolution at runtime
- Disconnect the definition of a method or set of methods from the inheritance hierarchy



# Interfaces in Java

- Java allows you to fully utilize the “one interface, multiple methods” aspect of polymorphism
- Designed to support dynamic method resolution at runtime
- Disconnect the definition of a method or set of methods from the inheritance hierarchy



# Agenda

- 1 Introduction
- 2 Interfaces in Java
  - Interfaces in Java
  - **Interface Definition**
  - Interface Implementation
- 3 Applications of Interface
- 4 Questions and Discussion



# Defining an Interface

- Variables declared in interface are implicitly **static** and **final**, and must be **initialized**



# Defining an Interface

- Variables declared in interface are implicitly **static** and **final**, and must be **initialized**
- General form of java interface:



# Defining an Interface

- Variables declared in interface are implicitly **static** and **final**, and must be **initialized**
- General form of java interface:

```
access interface name {  
    return-type method-name1 (parameter-list);  
    return-type method-name2 (parameter-list);  
    .  
    .  
    return-type method-nameN (parameter-list);  
    type variable-name1 = value1;  
    type variable-name2 = value2;  
    .  
    .  
    type variable-nameN = valuen;  
}
```





# Interface Example Code

## Example

```
public interface calculator {  
    int add (int x, int y);  
    int sub (int x, int y);  
    float dev (int x, int y);  
    long mul (int x, int y);  
    void display ();  
}
```



# Agenda

- 1 Introduction
- 2 Interfaces in Java
  - Interfaces in Java
  - Interface Definition
  - Interface Implementation
- 3 Applications of Interface
- 4 Questions and Discussion



# Implementing an Interface

- One or more classes can implement that interface



# Implementing an Interface

- One or more classes can implement that interface
- To implement an interface, include the **implements** clause in a class definition, and then create the methods defined by the interface



# Implementing an Interface

- One or more classes can implement that interface
- To implement an interface, include the **implements** clause in a class definition, and then create the methods defined by the interface
- Methods that implement an interface must be declared public



# Implementing an Interface

- One or more classes can implement that interface
- To implement an interface, include the **implements** clause in a class definition, and then create the methods defined by the interface
- Methods that implement an interface must be declared public
- type signature of the implementing method must match exactly the type signature specified in the interface definition



# Implementing an Interface

- One or more classes can implement that interface
- To implement an interface, include the **implements** clause in a class definition, and then create the methods defined by the interface
- Methods that implement an interface must be declared public
- type signature of the implementing method must match exactly the type signature specified in the interface definition
- General form of implementation:

```
access class classname [extends superclass][implements interface  
[,interface...]] {  
    // class-body  
}
```



# Interface Implementation Example Code

## Example

```
public class ClassicCalculator implements calculator {
    int addResult=0;
    int subResult=0;
    int divResult=0;
    int mulResult=0;
    int add (int x, int y){
        addResult = x+y;
        return addResult;
    }
    int sub (int x, int y){
        subResult = x-y;
        return subResult;
    }
    float dev (int x, int y){
        divResult = x/y;
        return divResult;
    }
    long mul (int x, int y){
        mulResult = x*y;
        return mulResult;
    }
    void display (){
        System.out.println("Result of addition: "+addResult);
        System.out.println("Result of subtraction: "+subResult);
        System.out.println("Result of division: "+divResult);
        System.out.println("Result of multiplication: "+mulResult);
    }
}
```



- To understand the real power of interface let's elaborate a practical example of Stack data structure



- To understand the real power of interface let's elaborate a practical example of Stack data structure
- Stack has two functions: **push** and **pop**



- To understand the real power of interface let's elaborate a practical example of Stack data structure
- Stack has two functions: **push** and **pop**
- Use in interface having above two functions



- To understand the real power of interface let's elaborate a practical example of Stack data structure
- Stack has two functions: **push** and **pop**
- Use in interface having above two functions
- Implement given interface for a fixed size stack



- To understand the real power of interface let's elaborate a practical example of Stack data structure
- Stack has two functions: **push** and **pop**
- Use in interface having above two functions
- Implement given interface for a fixed size stack
- Implement given interface for a dynamic size stack



# Your Turn: Time to hear from you!



1



<sup>1</sup><https://fensafitters.files.wordpress.com/2013/07/3d095.jpg>

# References



Herbert Schildt

*The complete reference Java2, 5th Edition .*

McGraw-Hill/Osborne, 2002.

