Desarrollo Avanzado de Aplicaciones I

Programación Orientada a Objetos 4

idat

i d a t

inicio desarrollo aplicación término

idat

Inicio

Logro de aprendizaje - Introducción

idat

Logro de Aprendizaje

“Al finalizar la sesión, el participante podrá implementar el concepto de abstracción usando interfaces y aplicando herencia múltiple.

idat

Introducción

idat

- Revisión Rápida de Temas de Sesión Anterior
- Revisión de Ejercicios de Sesión Anterior
- Inquietudes y/o Preguntas
- Agenda de Sesión
  - Interfaces
  - Herencia Múltiple
  - Revisión de Conceptos de POO
  - Ejemplos y Ejercicios

Desarrollo

Desarrollo del Contenido de la Sesión

idat

Interfaces

- ✓ Una interface es una especie de plantilla para la construcción de clases, definen un protocolo de comportamiento y proporcionan un formato común para implementarlo en las clases.
- ✓ Las interfaces se compone de un conjunto particulares de métodos y de atributos definidos como constantes. Estos atributos se deben inicializar en la misma instrucción de declaración.
- ✓ Una interface no se pueden instanciar, por lo que no podemos crear objetos a partir de una interface. Y las interfaces tampoco contienen constructores.
- ✓ Los nombres de las interfaces suelen acabar en “able” aunque no es necesario: Configurable, Arrancable, Dibujable, Comparable, Clonable, etc.

Interfaces (cont.)

- ✓ La interface puede definirse como pública (“public”) o sin modificador de acceso (predeterminado) y tiene el mismo significado que para las clases.
- ✓ Si tiene el modificador de acceso “public” el archivo .java que la contiene debe tener el mismo nombre que la interface.
- ✓ Al compilar el archivo .java de la interface, su byte-code se guarda también en un archivo .class, igual que las clases.
- ✓ Una clase puede implementar una o varias interfaces. En ese caso, la clase debe proporcionar la declaración y definición de todos los métodos de cada una de las interfaces o bien declararse como clase abstracta.
- ✓ La clase que implementa una o más interfaces utiliza la palabra reservada “implements”.

Contenido de las Interfaces

- ✓ Métodos abstractos (no tienen implementación y no tienen necesidad de usar el modificador de comportamiento “abstract”).
- ✓ Atributos constantes (se deben inicializar en la misma declaración y pueden omitir el uso de los modificadores “public”, “static” y “final”.).
- ✓ Métodos por defecto (A partir de Java 8, si tienen implementación y usan el modificador de comportamiento “default”)
- ✓ Métodos estáticos (A partir de Java 8, no se pueden redefinir y usan el modificador de comportamiento “static”)
- ✓ Métodos privados (A partir de Java 9, solo se pueden invocar en la interface y usan el modificador de acceso “private”)
- ✓ Tipos anidados (A partir de Java 8, definir interfaces dentro de interfaces)

Herencia en Interfaces

- ✓ Se puede establecer una jerarquía de herencia entre interfaces igual que con las clases.
- ✓ Cada interface hereda el contenido de las interfaces que están por encima de ella en la jerarquía y puede añadir nuevo contenido o modificar lo que ha heredado cumpliendo las siguientes condiciones:
  - ➢ Los métodos abstractos (“abstract”) heredados se pueden convertir en métodos por defecto (“default”).
  - ➢ Los métodos por defecto (“default”) se pueden redefinir o convertir en métodos abstractos (“abstract”).
  - ➢ Los métodos estáticos (“static”) no se pueden redefinir.
- ✓ En las interfaces sí se permite herencia múltiple.
- ✓ Las interfaces para aplicar la herencia deben utilizar la palabra reservada “extends”, al igual que las clases.

Sintaxis de las Interfaces

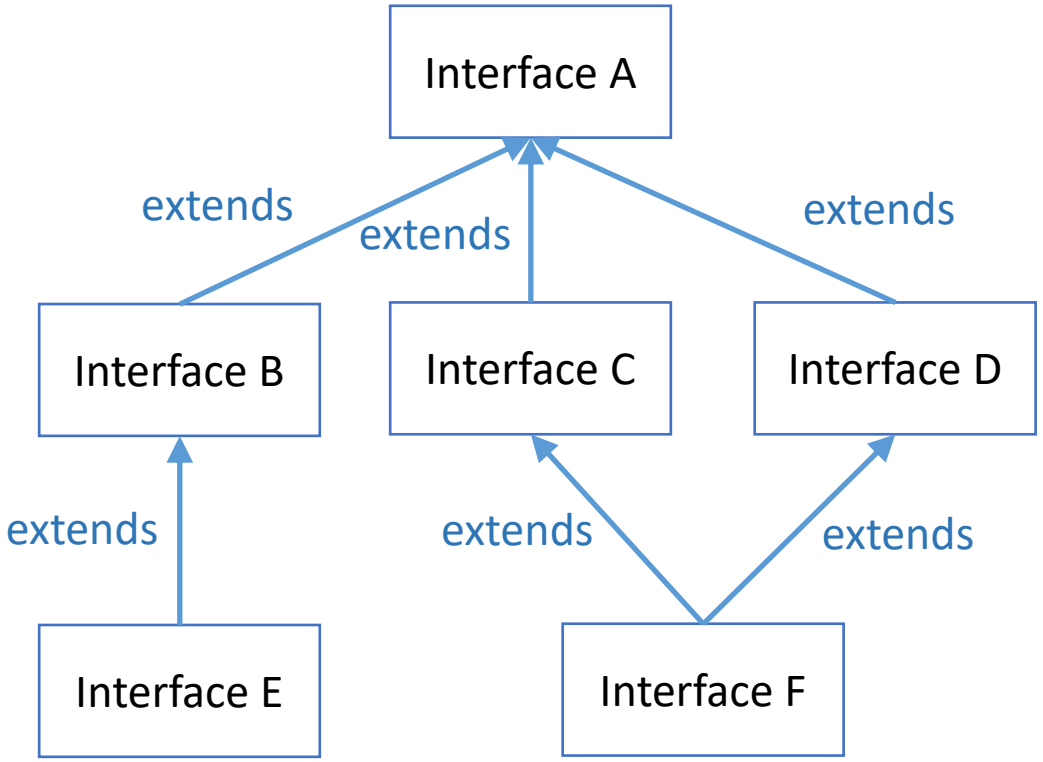
idat

```
[public] interface NombreInterface [extends Interface1, Interface2, ...]{
        [métodos abstractos]
        [métodos default]
        [métodos static]
        [métodos privados]
        [atributos constantes]
        [tipos anidados]
}
```

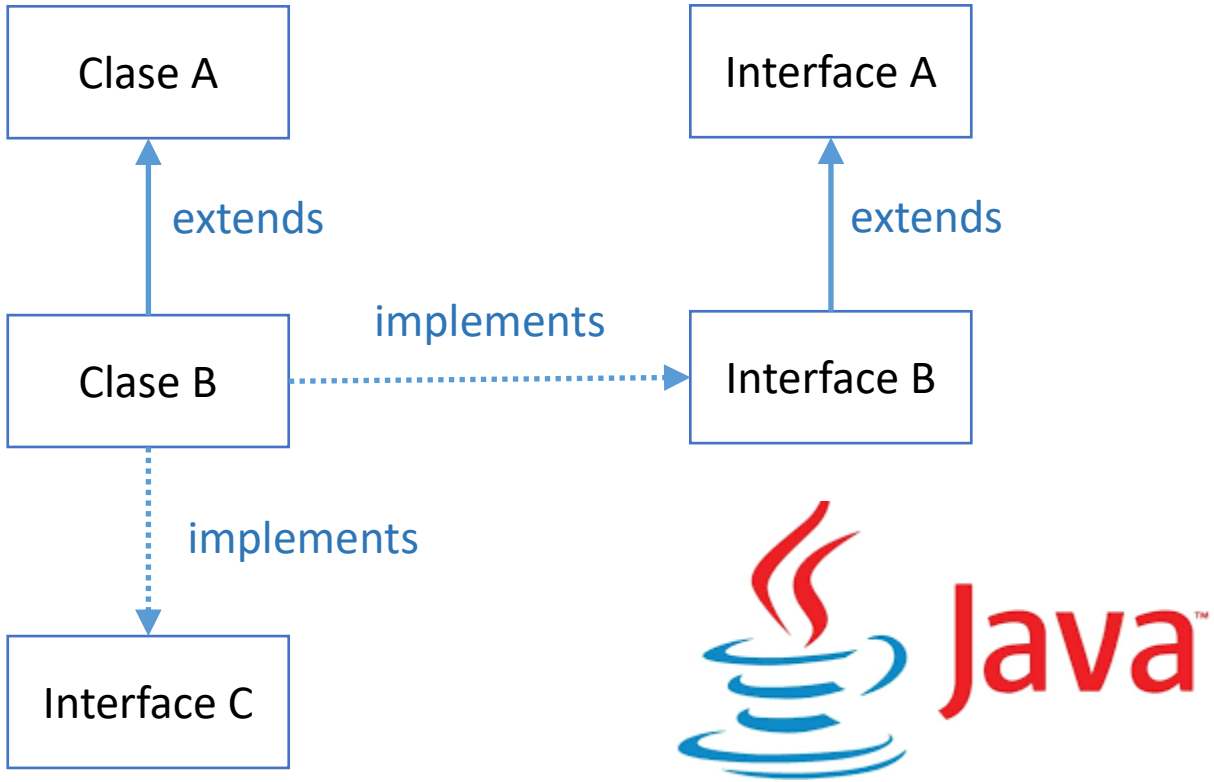
Ejemplo de Interfaces

idat

Declaración de Interfaces



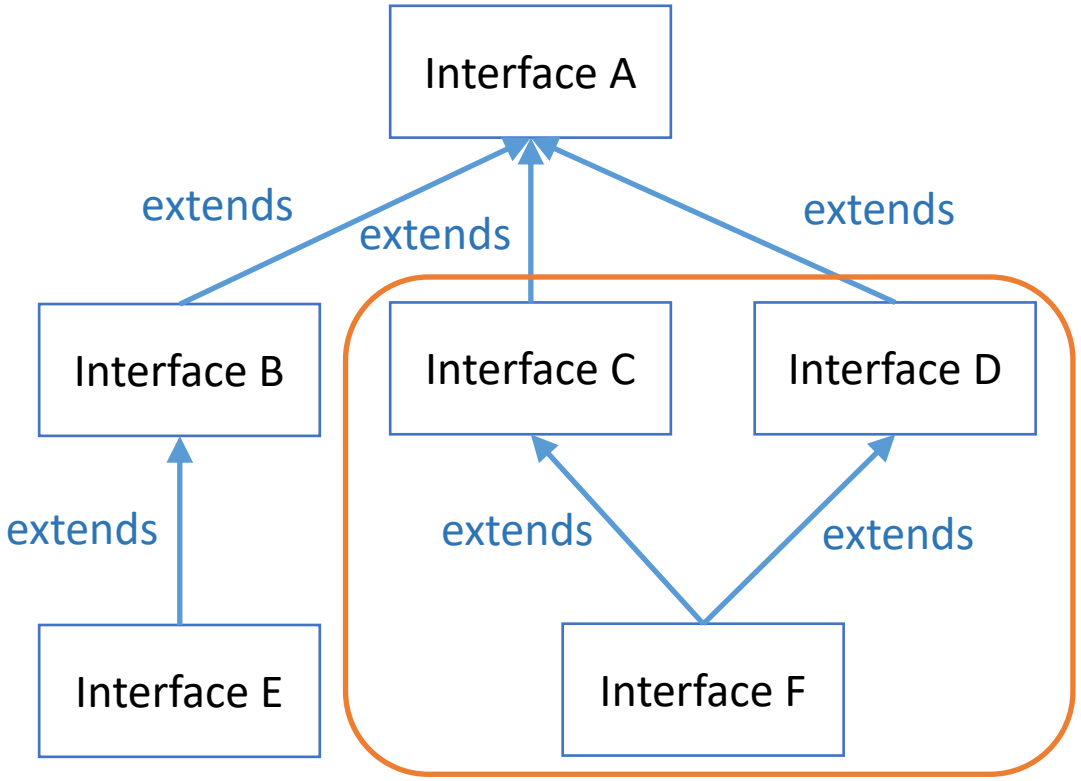
Declaración de Interfaces y Clases



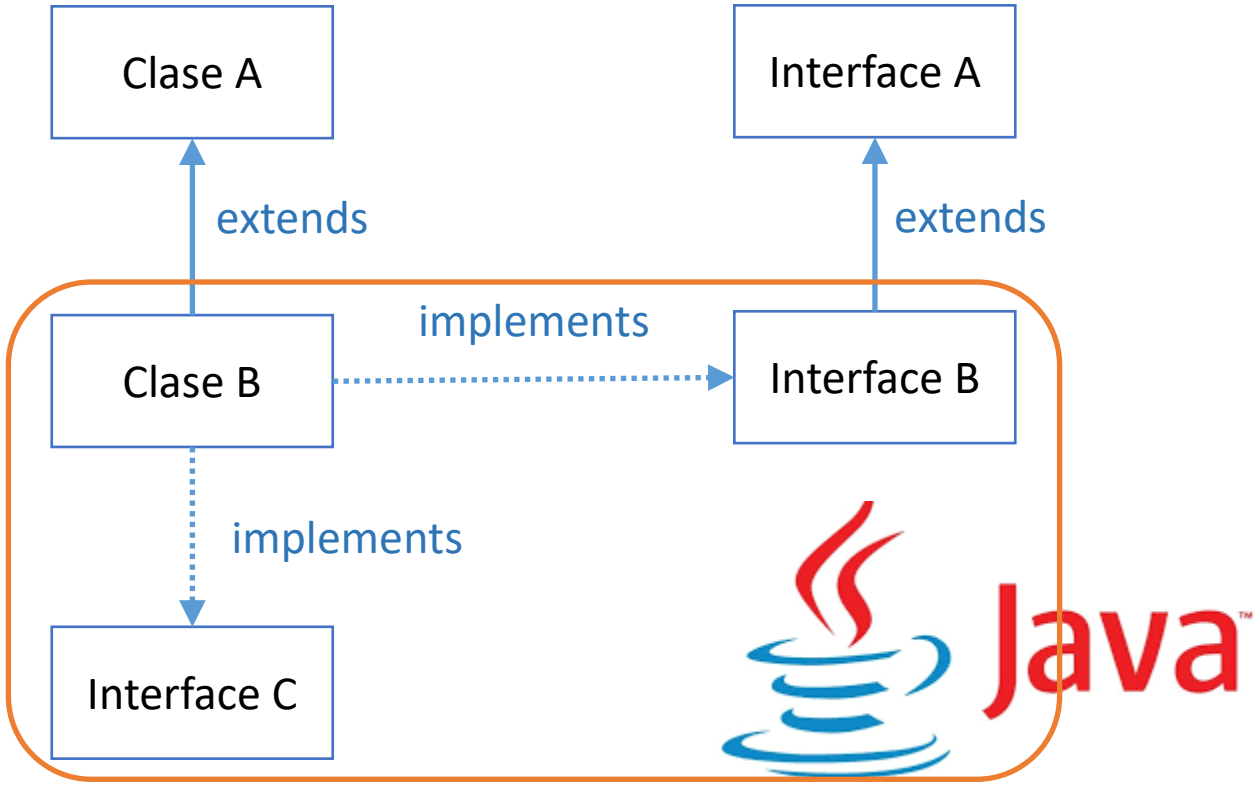
Herencia Múltiple

idat

Aplicado en Interfaces



Aplicado en Clases usando Interfaces



Ejemplo de Uso de Interfaces


```
interface Nadable {
    void nadaLibre();
    void nadaMariposa();
}
class Nadador implements Nadable {
    @Override
    public void nadaLibre() {
        System.out.println( x: "Nado libremente....");
    }
    @Override
    public void nadaMariposa() {
        System.out.println( x: "Nado con estilo mariposa....");
    }
}
```

Ejemplo de Uso de Interfaces (cont.)


```
public class MiClase7 {
    public static void main(String[] args) {
        Nadador nadador = new Nadador();
        nadador.nadaLibre();
        nadador.nadaMariposa();
    }
}
```

```
Nado libremente....
Nado con estilo mariposa....
```

Ejemplo de Herencia Múltiple

idat

```java
public class MiClase7 {
    public static void main(String[] args) {
        Deportista depor = new Deportista();
        depor.nadaLibre();
        depor.nadaMariposa();
        depor.caminar();
        depor.correr();
    }
}
interface Nadable {
    void nadaLibre();
    void nadaMariposa();
}
interface Ejercitable {
    void caminar();
    void correr();
}
```

```
Nado libremente....
Nado con estilo mariposa....
Estoy caminando...
Estoy corriendo...
```

```java
class Deportista implements Nadable, Ejercitable {
    @Override
    public void nadaLibre() {
        System.out.println( x: "Nado libremente....");
    }
    @Override
    riposa() {
    ntln( x: "Nado con estilo mariposa....");

    r() {
    ntln( x: "Estoy caminando...");

    @Override
    public void correr() {
        System.out.println( x: "Estoy corriendo...");
    }
}
```

Aplicación

Revisar ejemplos y realizar ejercicios prácticos

idat

Término

Indicaciones generales y/o Resumen de Sesión

idat

Resumen de Sesión

idat

- Interfaces
- Herencia Múltiple
- Revisión de Conceptos de POO
- Ejemplos y Ejercicios

GRACIAS