El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

CICLO II:

Programación Básica en Java

MisionTIC 2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Sesión 12:
Introducción a Java

Programación Orientada a Objetos (POO)

MisiónTIC2022

Objetivos de la sesión

Al finalizar esta sesión estarás en capacidad de:

1. Explicar los conceptos de Interfaz para manejar herencias múltiples
2. Desarrollar un programa donde aplique los conceptos de interfaz en conjunto con el concepto de herencia de una clase

Interfaz

- Una interfaz es una colección de métodos abstractos y constantes.
- Una interfaz no se puede instanciar.
- Una clase que implementa una interfaz debe implementar los métodos declarados en dicha interfaz.
- En una clase se pueden implementar múltiples interfaces.
- Java soporta herencia múltiple de interfaces.

Interfaz - Declaración

Sintaxis:

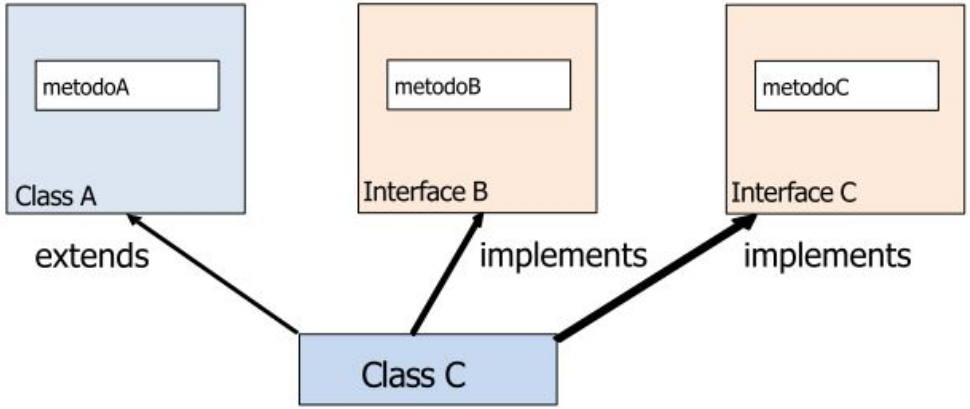
```
public interface nombreInterfaz {
    static final tipo CONSTANTE = valor;
    tipoDevuelto nombreMetodo(listaParam);
}
```

- No hay llaves. Al no estar implementado después de la declaración se pone sólo un ;
- Las constantes y métodos en la interfaz deben ser públicos.
- Los atributos son automáticamente public final static
- Todos los Métodos son automáticamente public abstract así no se diga explícitamente

Interfaz – Herencia Múltiple

- Herencia múltiple: capacidad para heredar métodos de dos o más clases.
- Las interfaces también pueden heredar de otras interfaces, consiguiendo así una nueva interfaz, se emplea el extends habitual.
- En las interfaces se especifica qué se debe hacer pero no su implementación. Serán las clases que implementen estas interfaces las que describen la lógica del comportamiento de los métodos.

Interfaz – Herencia Múltiple



- En Java una clase hereda de una única superclase
  o no existe la herencia múltiple
- Pero puede implementar varias interfaces

Interfaz - Ejemplo

```
public interface Vehiculo {

    public boolean tieneSeguro();

    public void recorreDistancia(int numKilometros);

    public double getCombustibleRestante();

}
```

Inicio de la definición de la interfaz "Vehiculo"

Definición de métodos

Interfaz - Implementación

Cuando una clase implementa una interfaz, debe implementar todos los métodos abstractos de esa interfaz:

- Para representar esto se utiliza la palabra reservada implements:

```
public class nombreClase implements Interfaz
{
    //cuerpo de la clase
}
```

Implementando una Interfaz

```
public interface Vehiculo {
    int VELOCIDAD_MAXIMA=120;

    String frenar(int cuanto);
    String acelerar(int cuanto);
}
```

Implementando una Interfaz

```
public class Taxi implements Vehiculo {
    int velocidad = 0;
    @Override
    public String frenar(int cuanto) {
        velocidad -= cuanto;
        return velocidad + "km/hora";
    }
    @Override
    public String acelerar(int cuanto) {
        String cadena = "";
        velocidad += cuanto;
        if (velocidad > VELOCIDAD_MAXIMA)
            cadena = "Exceso de velocidad";
        cadena += "El taxi ha acelerado a " +
velocidad + "km/hora";
        return cadena;
    }
  }
}
```

```
class Main {
    public static void main(String[]
args) {
        Taxi myTaxi = new Taxi();
        String frenar =
myTaxi.frenar(4);
        String acelerar =
myTaxi.acelerar(10);
        System.out.println(frenar);

System.out.println(acelerar);
    }
}
```

Clase principal.

Implementando una Interfaz

```
interface SerPadre {
    void criar();
}
interface SerEsposo {
    void amar();
}
interface SerEmpleado {
    void trabajar();
}
class EmpleadoDeOracle {
    public void trabajar() {
    }
}
class Programador extends EmpleadoDeOracle
implements SerPadre , SerEsposo, SerEmpleado {
    public void criar() {
    }
    public void amar() {
    }
}
```

Como se observa se implementaron varias interfaces y Programador está comprometido a implementar los métodos de las interfaces, sin embargo, serEmpleado y EmpleadoDeOracle tienen el mismo método trabajar(), así que no se proporciona explícitamente trabajar() porque lo toma de la clase EmpleadoDeOracle, esto quiere decir que cuando se cree un objeto Programador, este tomará automáticamente ese método de EmpleadoDeOracle.

Implementando una Interfaz

```
interface Galleta {
    void crearMasa();
    void darForma();
    void cocinar();
}

interface GalletaConChispas extends Galleta {
    public void anadirChispas();
}
class Cena implements GalletaConChispas {
    public void crearMasa() {
    }
    public void darForma() {
    }
    public void cocinar() {
    }
    public void anadirChispas() {
    }
}
```

Las interfaces también pueden heredar de otras interfaces, consiguiendo así una nueva interfaz, se emplea el extends habitual. Teniendo así que la clase que implementa a esa nueva interfaz recibirá los métodos tanto de la interfaz base como la derivada.

Operador instanceof

El operador instanceof se utiliza para conocer si un objeto es de un tipo determinado (clase o interfaz), es decir, si el objeto pasaría el test «ES UN» para esa clase o ese interfaz.

```
public class Vehiculos {
  public class Taxi extends Vehiculos{
       public static void main (String[] args){
       Taxi taxi1 = new Taxi();
       if(taxi1 instanceof Vehiculos)
           System.out.println("taxi1 es un Taxi y también un vehiculo.");

       Vehículo tax = new Taxi();
       ElCast(tax);
       }
       public static void ElCast(Vehiculo a){
            if (a instanceof Taxi)
               ((Taxi)a).Potencia();
   }
   public static void Potencia(){
     System.out.println("100");
   }
}
```

Operador instanceof

- Cuando utilicemos el operador instanceof, debemos recordar que sólo puede usarse con variables que contengan la referencia a un objeto.
- Puede ocurrir que el objeto que estamos comprobando con instanceof, no sea una instancia directa de la clase que aparece a la derecha del operador, aun así instanceof devolverá true si el objeto es de un tipo compatible con el objeto que aparece a su derecha.
- Un error de compilación de instanceof es que no se puede usar para tratar de comprobar dos clases de diferentes jerarquías.

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Seguimiento Habilidades Digitales en Programación

* De modo general, ¿Cuál es grado de satisfacción con los siguientes aspectos?

	Nada Satisfecho	Un poco satisfecho	Neutra	Muy satisfecho	Totalmente satisfecho
Sesiones técnicas sincrónicas	○	○	○	○	○
Sesiones técnicas asincrónicas	○	○	○	○	○
Sesiones de inglés	○	○	○	○	○
Apoyo recibido	○	○	○	○	○
Material de apoyo: diapositivas	○	○	○	○	○

Completa la siguiente encuesta para darnos retroalimentación sobre esta semana ▼▼▼

https://www.questionpro.com/t/ALw8TZlxOJ

MisiónTIC2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

Ejercicios para practicar

Mision TIC 2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

¡GRACIAS

POR SER PARTE DE ESTA EXPERIENCIA DE APRENDIZAJE!

Mision TIC 2022