POO: Interfaces en Java

CREAMÁS
GRUPO DE INVESTIGACIÓN

e-qüid
Semillero de investigación

SENA
Más trabajo

Centro de
Comercio y Turismo

Interfaces

Cuando hablamos de interfaces es muy común confundir el termino con interface grafica de usuario, esto es un grabe error ya que una Interface en java es un concepto asociado a la POO.

Una Interface es una Clase completamente Abstracta, como regla, sabemos que las **clases abstractas** poseen como mínimo un método abstracto, pero hablando de una **interface**, todos sus métodos y atributos tienen que serlo.

Interfaces

las interfaces muchas veces son definidas como un tipo de contrato entre las clases concretas que la implementen, ya que la clase que lo haga se encuentra obligada a definir los métodos abstractos que la componen.

Cuando creamos una Interface, lo que hacemos es definir lo que la clase que la implemente podrá hacer, pero no indicamos la forma en que lo hará.

Interfaces

Las interfaces son el mecanismo que utiliza Java para simular la herencia múltiple, como mencionamos en Java solo se puede extender de una sola clase, mediante el uso de interfaces esto se puede simular ya que el lenguaje permite implementar el numero de interfaces que necesitemos.

Adicionalmente las interfaces pueden heredar uno o mas números de interfaces mediante la palabra **extends,** pero jamás podrán heredar clases concretas...

Interfaces

Al decir que las interfaces son clases completamente abstractas significa que todos sus métodos lo son y por ende no poseen implementación, no requieren el uso de la palabra reservada **abstract**, ya que al ser completamente abstracta todo dentro de ella lo es, al igual que las clases abstractas la implementación de los métodos depende de las clases concretas que las usen.

Se debe tener en cuenta que toda variable definida en una interfaz automáticamente se convierte en una constante, además tampoco se puede instanciar una interfaz

Interfaces

En java se representan con la palabra **interface** y se usan con la palabra **implements** así:

```
interface InterfacePrincipal {

 public void metodoAbstracto();

 public String otroMetodoAbstracto();

}
```

```
public class Principal implements InterfacePrincipal{

 public void metodoAbstracto() {
  /**Implementación definida por la clase concreta*/
 }

 public String otroMetodoAbstracto() {
  /**Implementación definida por la clase concreta*/
  return "retorno";
 }
}
```

Interfaces

Características de las Interfaces.

- Todos los métodos de una interfaz son implícitamente **public abstract**, no es necesario especificarlo en la declaración del mismo.
- Todas las variables y atributos de una interfaz son implícitamente constantes (**public static final**), no es necesario especificarlo en la declaración del misma
- Los métodos de una interfaz no pueden ser: **static, final, strictfp ni native.**
- Una interfaz puede heredar (**extends**) de una o más interfaces.
- Una interfaz no puede heredar de otro elemento que no sea una interfaz.
- Una interfaz no puede implementar (**implements**) otra interfaz.
- Una interfaz debe ser declarada con la palabra clave **interface**.
- Los tipos de las interfaces pueden ser utilizados polimórficamente.
- Una interfaz puede ser **public** o **package** (valor por defecto).
- Los métodos toman como ámbito el que contiene la interfaz.

Interfaces

Basado en lo anterior, cualquiera de las siguientes declaraciones es valida.

```
/**Las interfaces Heredan de cualquier cantidad de interfaces usando extends
 public interface PrimerInterface extends SegundaInterface, TercerInterface
{

/**Las siguientes son declaraciones validas para los atributos*/
   int ATRIBUTO1= 5;
   public int ATRIBUTO2= 5;
   public static int ATRIBUTO3= 5;
   public final int ATRIBUTO4= 5;
   static int ATRIBUTO5= 5;
   final int ATRIBUTO6= 5;
   static final int ATRIBUTO7= 5;
   public static final int ATRIBUTO8= 5;

/**Las siguientes son declaraciones validas para los métodos*/
   void metodoPrimerInterface1();
   public void metodoPrimerInterface2();
   abstract void metodoPrimerInterface3();
   public abstract void metodoPrimerInterface4();
 }
```

Interfaces

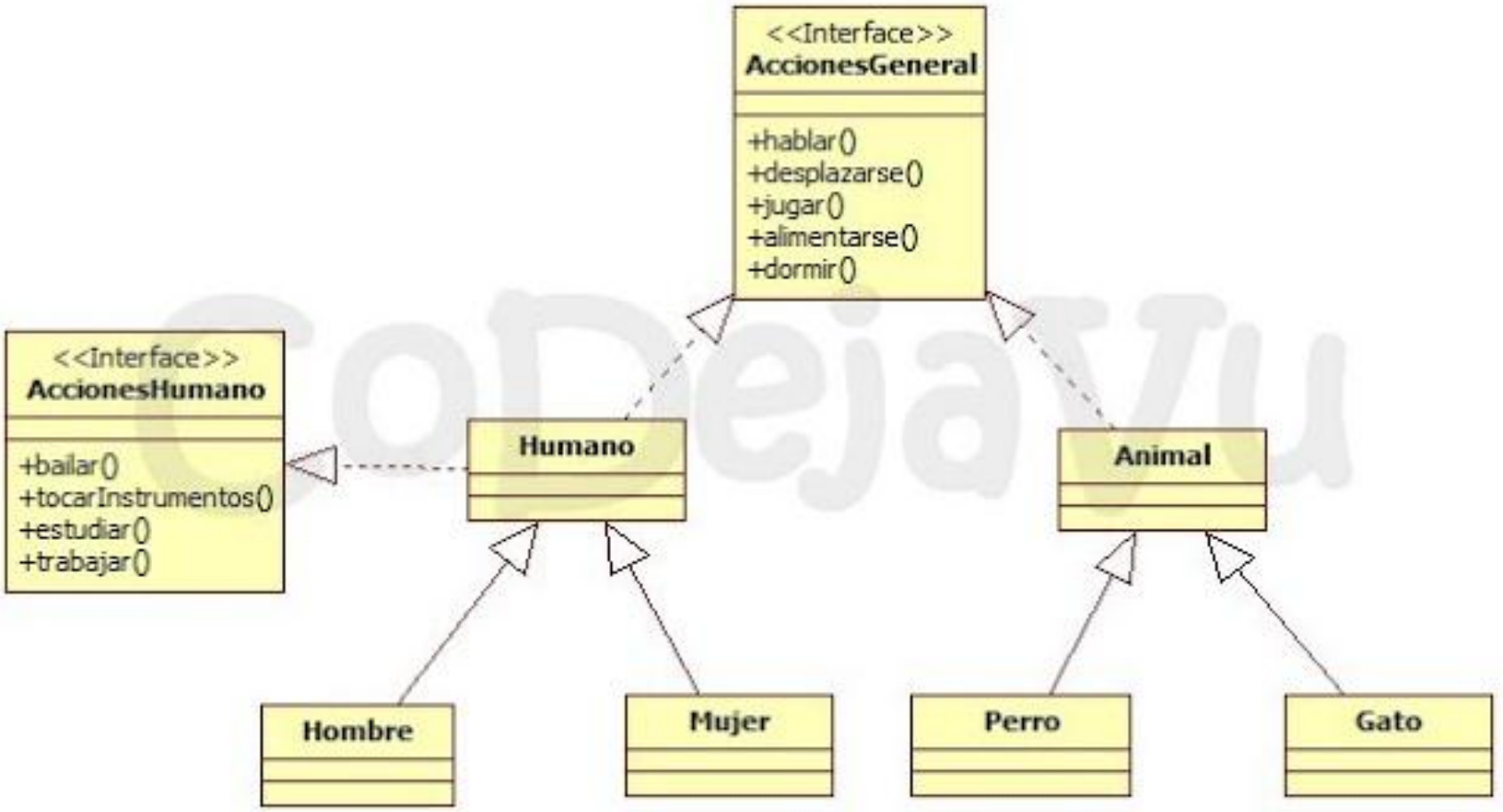
Para clases Abstractas como para Interfaces la **herencia** es permitida, pero por ejemplo para este tipo componentes, si una **interface hereda** de otra, esta no está obligada a implementar los métodos que posee la **Interface padre**, ya que la implementación tanto de los métodos de la **clase padre** como de la **interface** que los **hereda** depende de la **clase concreta** que **implemente** dicha **interface**.

Este principio también aplica a las **clases Abstractas**, si **una clase abstracta implementa una interface**, los métodos de esta no necesariamente se deben implementar en la **clase Abstracta**, pero si se tienen que implementar en la **clase concreta que herede de la clase abstracta**

Interfaces

Su uso esta muy ligado al concepto de herencia y cumple el mismo principio que aplicamos al usar clases abstractas, lo que buscamos es establecer un mecanismo donde podamos compartir características comunes entre clases diferentes, además al igual que con clases abstractas nos aseguramos que los métodos y atributos solo están disponibles para las clases que las implementen.

Interfaces


<<Interface>>
AccionesGeneral
+hablar()
+desplazarse()
+jugar()
+alimentarse()
+dormir()
<<Interface>>
AccionesHumano
+bailar()
+tocarInstrumentos()
+estudiar()
+trabajar()
Humano
Animal
Hombre
Mujer
Perro
Gato

FUENTES

- Fundamentos de programación. Problemas resueltos de programación en lenguaje Java. José María Pérez Menor, Jesús Carretero Pérez, Félix García Carballeira, and José Manuel Pérez Lobato. Madrid: Paraninfo, 2003. p[1]-22

- http://codejavu.blogspot.com/2013/05/conceptos-de-programacion-orientada.html

- http://codejavu.blogspot.com/2013/05/interfaces-en-java.html

- JavaWorld – Revista Digital

- SCJP Sun Certified Programmer for Java 6