Desarrollo Avanzado de Aplicaciones I

Programación Orientada a Objetos 3

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 herencia y polimorfismo, jerarquía de clases y sus modificadores que aplican.”

idat

Introducción

idat

- Revisión Rápida de Temas de Sesión Anterior
- Revisión de Ejercicios de Sesión Anterior
- Inquietudes y/o Preguntas
- Agenda de Sesión
  - Herencia
  - Polimorfismo
  - Jerarquía de Clases
  - Referencia super
  - Modificadores

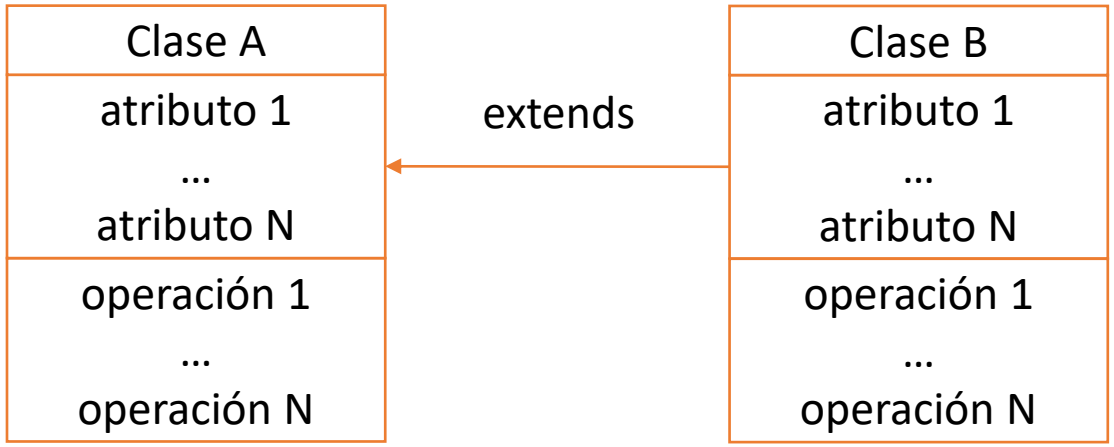
Desarrollo

Desarrollo del Contenido de la Sesión

idat

Herencia

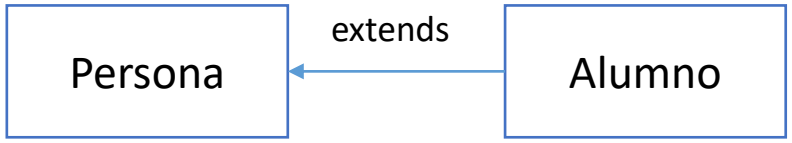
- ✓ La herencia es un mecanismo que permite la declaración de nuevas clases a partir de otras ya existentes.
- ✓ La herencia permite compartir (heredar) automáticamente atributos y operaciones entre superclases, clases, subclases y objetos. Esto implica que una superclase o clase base o clase padre transfiere sus atributos y operaciones a una subclase o clase derivada o clase hija.



La Clase B hereda los atributos y operaciones de la Clase A

Herencia (cont.)

- ✓ Las clases hijas, subclases o clases derivadas añaden características específicas (atributos y/o operaciones) que las diferencian de sus superclases o clases padre. O pueden redefinir (sobrescribir) las operaciones heredadas (Polimorfismo).
- ✓ La herencia permite establecer una jerarquía de generalización / especialización mediante la relación "es-un" o "es-una".
- ✓ La herencia proporciona el beneficio de la reutilización de código ya que permite a una clase más específica incorporar la estructura y comportamiento de una clase más general.



El Alumno es una Persona.

Herencia (cont.)

idat

- ✓ Cuando instancias una clase hija o subclase, se instancia también su clase padre o superclase. Es decir, se ejecuta el constructor de la clase padre o superclase. Y si hubieran mas clases padres o superclases se instancian en cascada.
- ✓ La palabra reservada que nos permite realizar herencia entre clases es "extends".
- ✓ En Java, si una clase no tiene una superclase explícita, implícitamente su superclase es la clase Object.

El Vehículo es un Objeto. La Moto es un Vehículo.

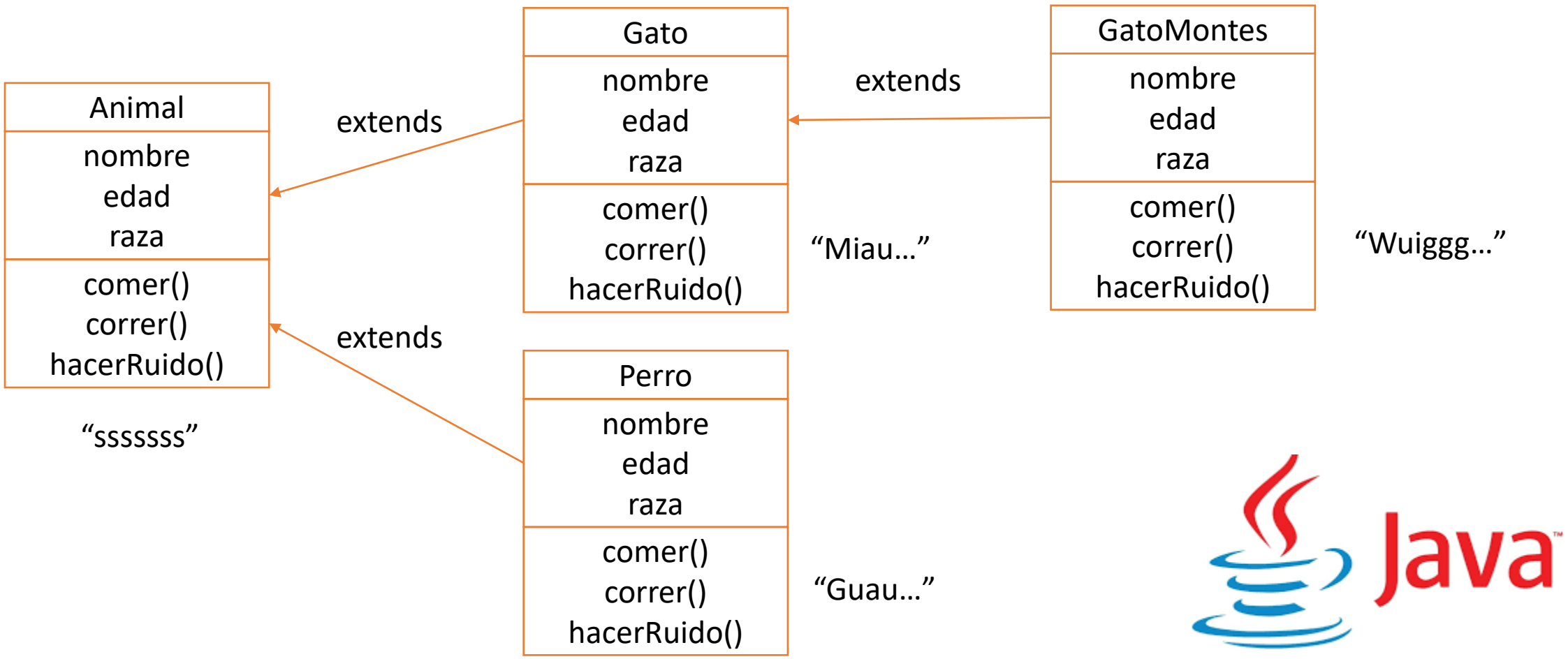
Java

Polimorfismo

idat

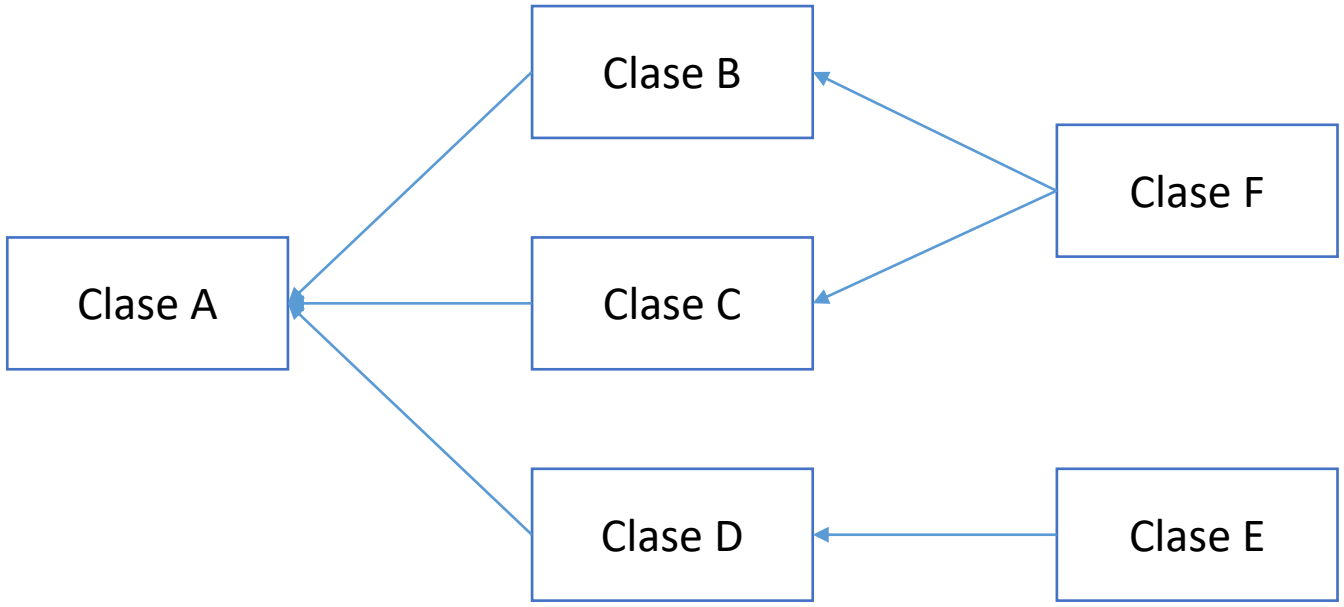
- ✓ El polimorfismo es una característica que permite llamar a métodos con igual nombre pero que pertenecen a clases distintas derivadas de una misma clase padre o superclase.
- ✓ El concepto de polimorfismo está estrechamente ligado con el concepto de herencia y es también llamado polimorfismo dinámico o en tiempo de ejecución.
- ✓ No se debe confundir polimorfismo con sobrecarga de métodos. Aunque algunos autores indican que la sobrecarga es un tipo de polimorfismo.
- ✓ El polimorfismo dinámico se implementa a través de la sobrescritura de métodos heredados. Es decir, una clase hija o subclase proporciona una implementación especializada de un método que ya existe en la clase padre o superclase.
- ✓ La sobreescritura se aplica usando la anotación @Override.

Ejemplo de Herencia y Polimorfismo



Jerarquía de Clases

- ✓ La herencia está representada por una estructura jerárquica o estructura de árbol, donde la clase padre de cualquier clase es conocida como su superclase. La clase hija de una superclase es llamada una subclase.

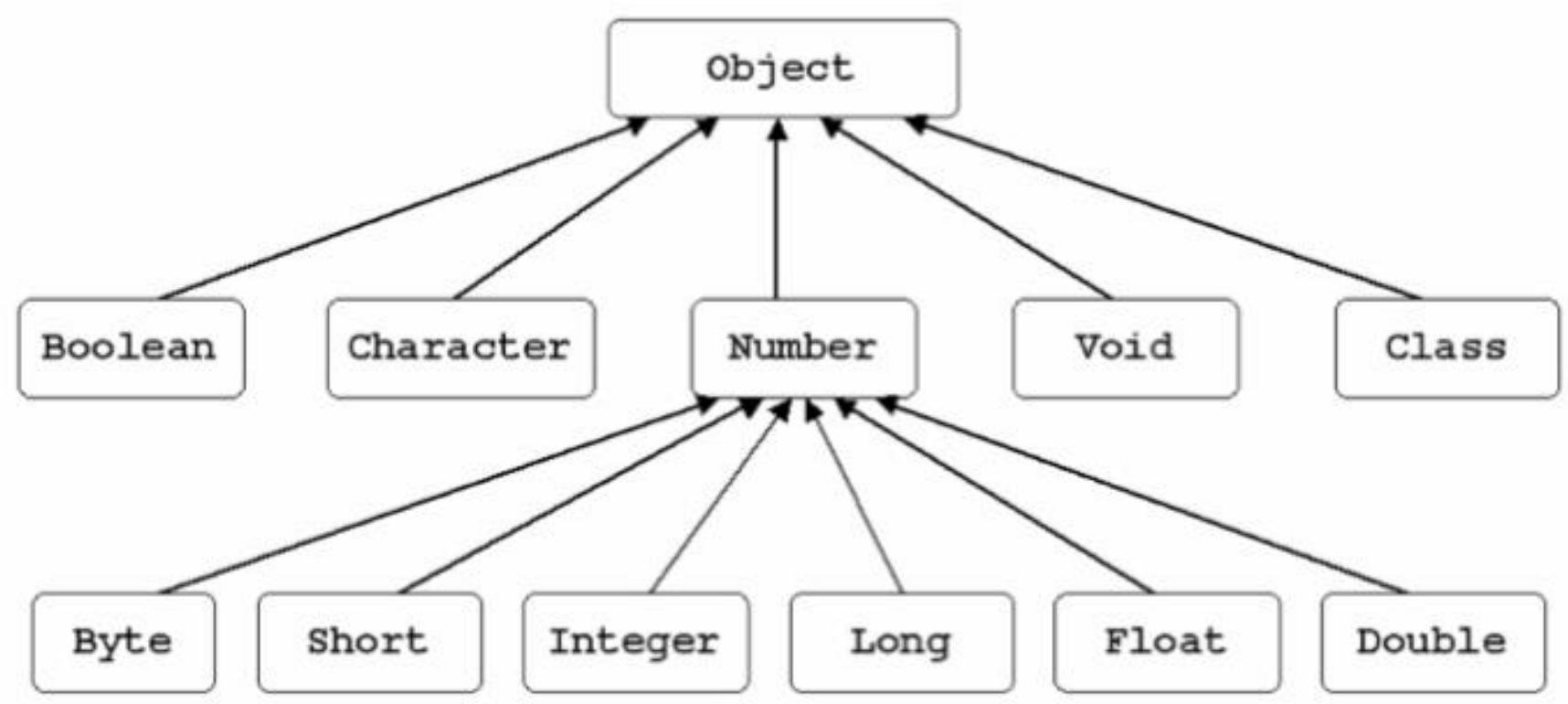


A es la superclase de B, C y D.
B, C y D son subclases de A.
D es la superclase de E.
E es una subclase de D.
B y C son superclases de F.
F es una subclase de B y C.

La Clase Object

idat

- ✓ La Clase Object es la clase raíz de toda la jerarquía de clases de Java.



Java

La Clase Object (cont.)

idat

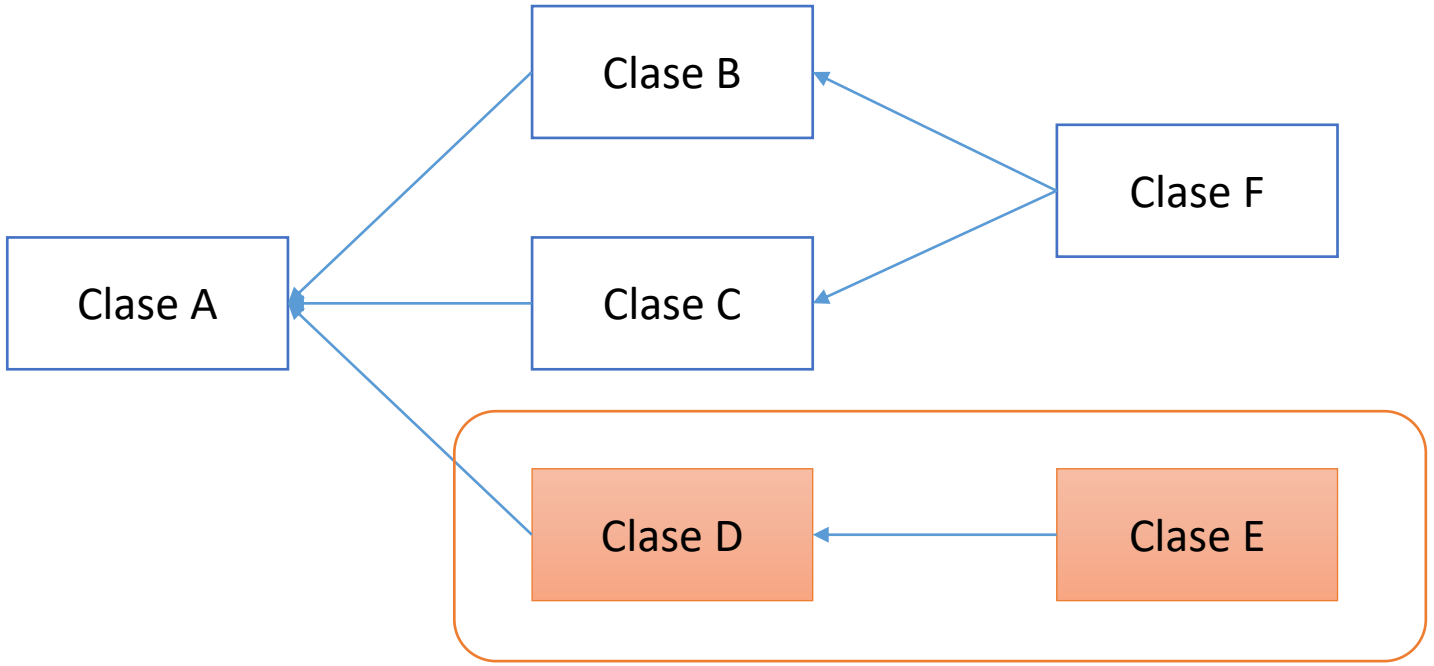
- ✓ Todas las clases tienen algunos métodos heredados de la clase Object:

Método	Función
clone()	Genera una instancia a partir de otra de la misma clase.
equals()	Devuelve un valor lógico que indica si dos instancias de la misma clase son iguales.
toString()	Devuelve un String que contiene una representación como cadena de carateres de una instancia.
finalize()	Finaliza una instancia durante el proceso de recogida de basura del Garbage Collector.
getClass()	Devuelve la clase a la que pertenece una instancia.

Tipos de Herencia

idat

- ✓ Existen cuatro tipos de herencia:
  - ➢ Herencia Simple o Única. Una clase hija hereda de una clase padre.

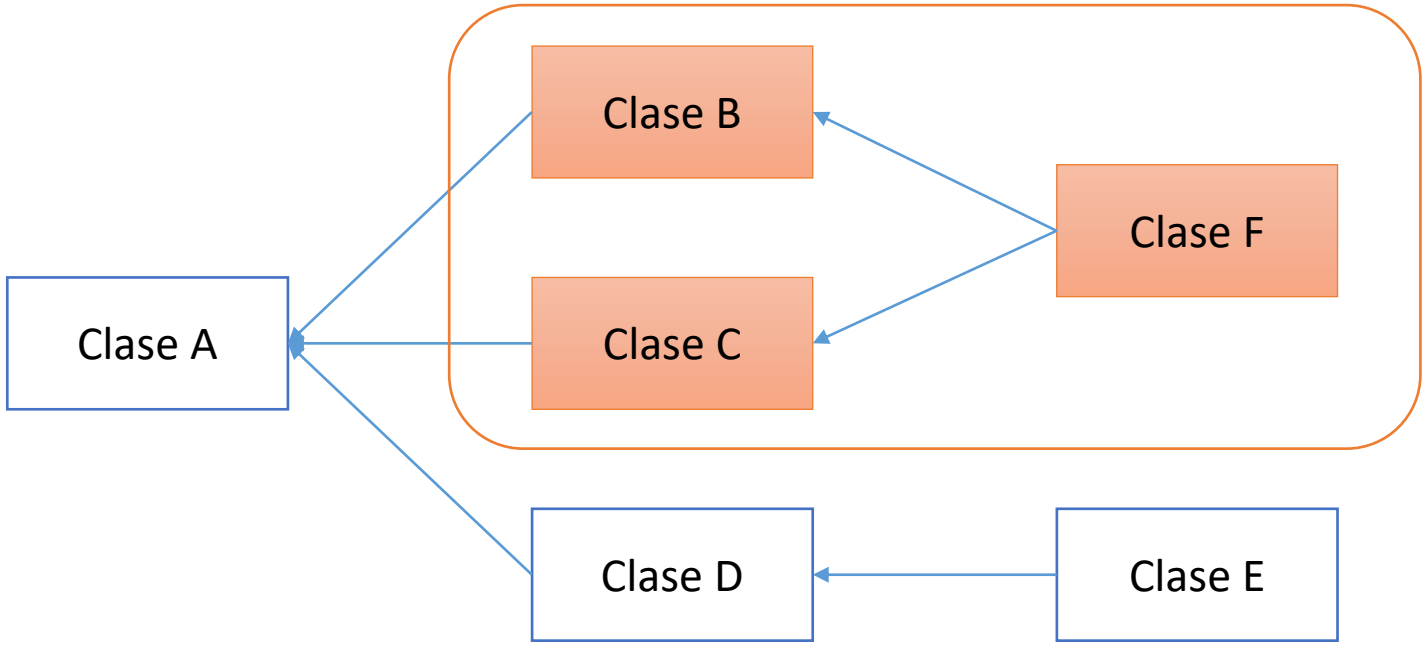


Java

Tipos de Herencia (cont.)

idat

- Herencia Múltiple. Una clase hija hereda de dos o más clases padre. Java no lo soporta. Se puede aplicar usando Interfaces.

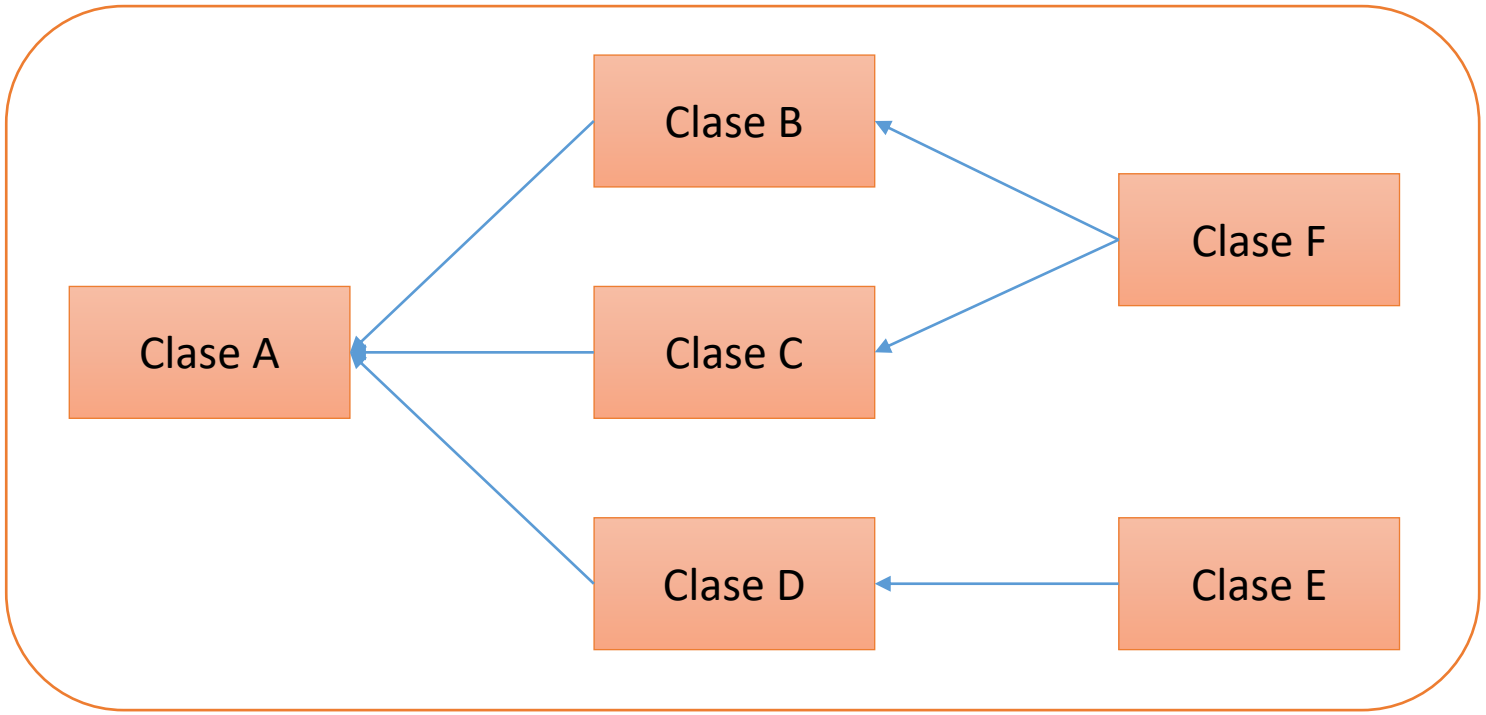


Java

Tipos de Herencia (cont.)

idat

- Herencia Multinivel. Una clase hija hereda de una clase padre, que a su vez es clase hija de otra clase padre.

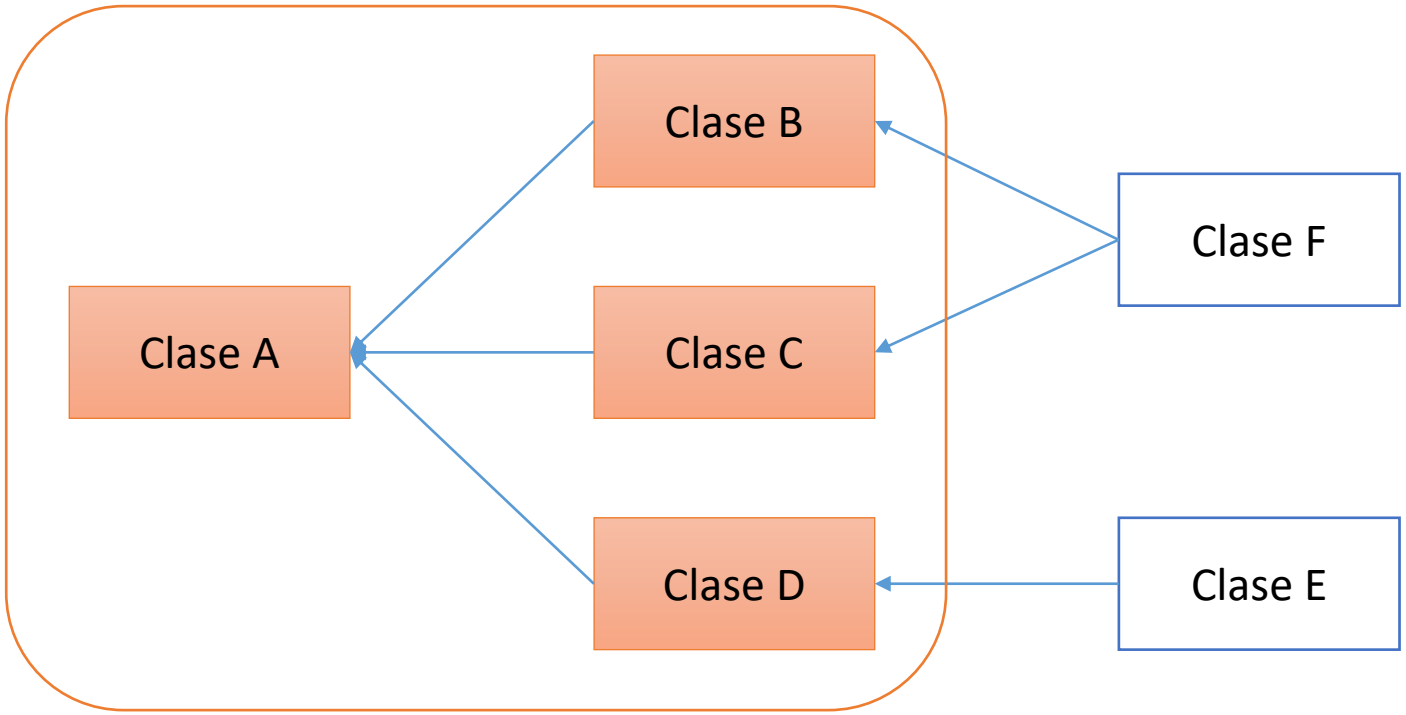


Java

Tipos de Herencia (cont.)

idat

- Herencia Jerárquica. Una clase padre hereda a dos o más clases hijas.



Java

Ejemplo de Herencia Simple y Polimorfismo


```java
class Animal {
  public void hacerRuido() {
    System.out.println( x: "Grr...");
  }
}
class Gato extends Animal {
  @Override
  public void hacerRuido() {
    System.out.println( x: "Miauuuu....");
  }
}
```

Clase padre o clase base o superclase

Método general

Clase derivada o clase hija o subclase

Anotación

Método específico. Sobrescrito.

Ejemplo de Herencia Simple (cont.)


```
public class Herencia {
    public static void main(String[] args){
        Animal ani = new Animal();
        ani.hacerRuido();
        Gato cat = new Gato();
        cat.hacerRuido();
    }
}
```

```
Grr...
Miauuuu....
```

Ejemplo de Herencia Jerárquica

idat

```
class Animal {
    public void hacerRuido() {
        System.out.println( x: "Grr...");
    }
}
class Gato extends Animal {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Miauuuu....");
    }
}
```

```
class Perro extends Animal {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Guau Guau...");
    }
}
```

Dos clases hijas o subclases.

Dos comportamientos para el método “hacerRuido”.

Ejemplo de Herencia Jerárquica (cont.)

idat

```
public class Herencia {
    public static void main(String[] args){
        Animal ani = new Animal();
        ani.hacerRuido();
        Gato cat = new Gato();
        cat.hacerRuido();
        Perro dog = new Perro();
        dog.hacerRuido();
    }
}
```

```
Grr...
Miauuuu....
Guau Guau...
```

Java

Ejemplo de Herencia Multinivel

idat

Gato es la subclase de Animal.

GatoMontes es la subclase de Gato.

```
class Animal {
    public void hacerRuido() {
        System.out.println( x: "Grr...");
    }
}
class Gato extends Animal {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Miauuuu....");
    }
}
```

```
class GatoMontes extends Gato {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Wuigggg....");
    }
}
```

Ejemplo de Herencia Multinivel (cont.)

idat

```
public class Herencia {
    public static void main(String[] args){
        Animal ani = new Animal();
        ani.hacerRuido();
        Gato cat = new Gato();
        cat.hacerRuido();
        Perro dog = new Perro();
        dog.hacerRuido();
        GatoMontes catm = new GatoMontes();
        catm.hacerRuido();
    }
}
```

```
Grr...
Miauuuu....
Guau Guau...
Wuigggg....
```

Java

Ejemplo de Instancias en Cascada de Superclases

idat

```
class Animal {
    public Animal() {
        System.out.println( x: "Superclase");
    }
    public void hacerRuido() {
        System.out.println( x: "Grr...");
    }
}
class Gato extends Animal {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Miauuuu....");
    }
}
```

```
class Perro extends Animal {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Guau Guau...");
    }
}
class GatoMontes extends Gato {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Wuigggg....");
    }
}
```

Ejemplo de Instancias en Cascada de Superclases (cont.)


```
public class Herencia {
    public static void main(String[] args){
        Animal ani = new Animal();
        ani.hacerRuido();
        Gato cat = new Gato();
        cat.hacerRuido();
        Perro dog = new Perro();
        dog.hacerRuido();
        GatoMontes catm = new GatoMontes();
        catm.hacerRuido();
    }
}
```

```
Superclase
Grr...
Superclase
Miauuuu....
Superclase
Guau Guau...
Superclase
Wuigggg....
```

Referencia super

idat

- ✓ Se puede utilizar la referencia “super” para eludir la versión sobrescrita de un método de la clase hija o subclase e invocar a la versión original del método en la clase padre o superclase.
- ✓ Del mismo modo, se puede utilizar “super” para acceder a variables miembro de la clase padre o superclase.

```
class Animal {
    String ruido = "Grrr...";
    public void hacerRuido() {
        System.out.println( x: ruido); } }
class Gato extends Animal {
    String ruido = "Miau...";
    @Override
    public void hacerRuido() {
        super.hacerRuido();
        System.out.println(super.ruido
                + this.ruido);   } }
```

```
public class Herencia {
    public static void main(String[] args){
        Animal ani = new Animal();
        ani.hacerRuido();
        Gato cat = new Gato();
        cat.hacerRuido();
    }
}
```

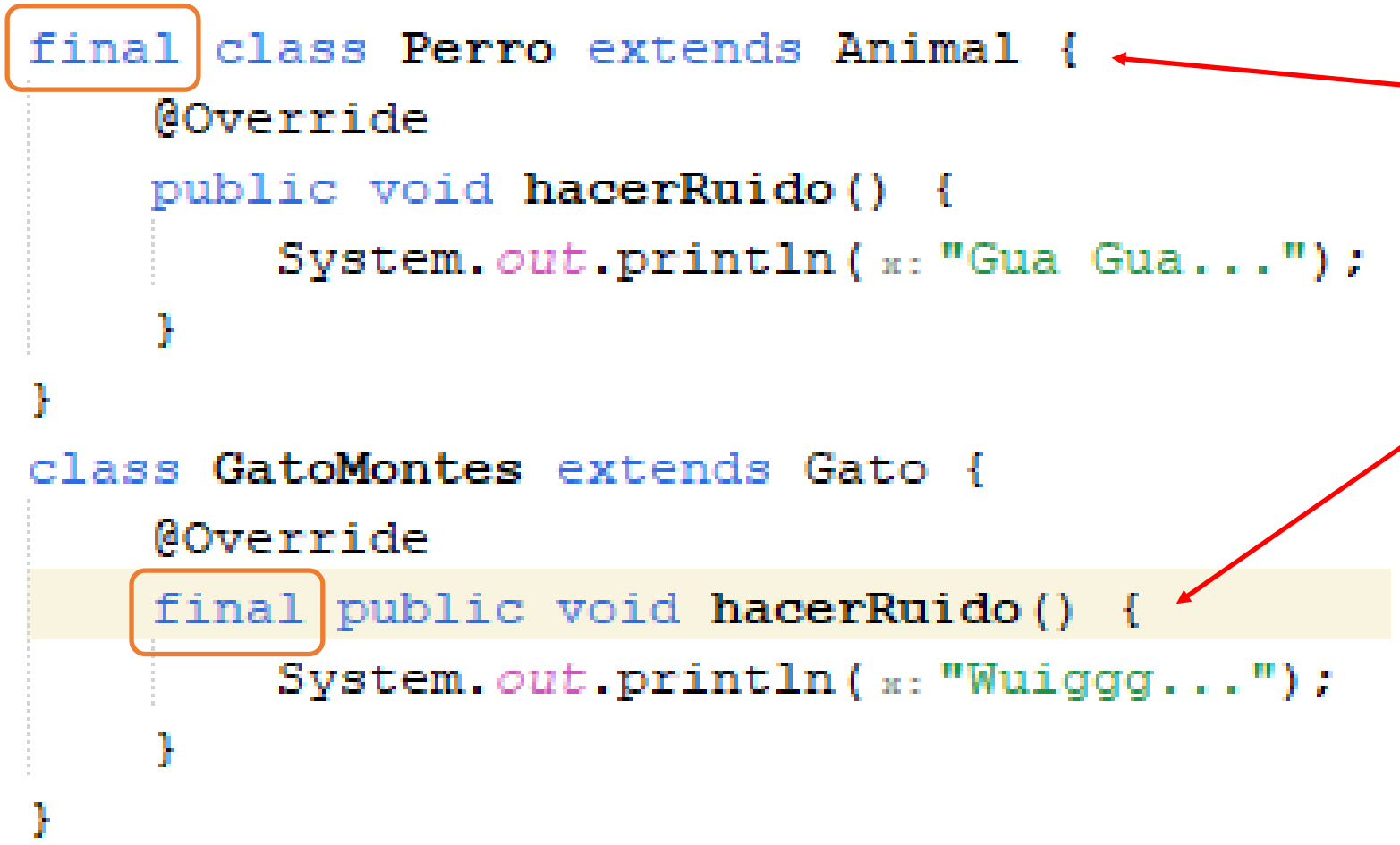
```
Grrr...
Grrr...
Grrr...Miau...
```

Modificadores de Comportamiento

idat

Modificador	Aplica para	Comportamiento
Estático static	Variables y métodos	Pertenece a la clase y no a las instancias de la clase. Se puede utilizar de forma directa sin instanciar un objeto de la clase. Los métodos estáticos solo pueden invocar directamente a otros métodos y variables estáticos de la clase.
Final final	Clases, variables y métodos	Indica que una variable, método o clase no se va a modificar. Si una variable se marca como final, no se podrá asignar un nuevo valor a la variable. Si una clase se marca como final, no se podrá extender la clase. Si un método se declara como final, no se podrá sobrescribir.
Abstracto abstract	Clases y métodos	Indica que no se provee una implementación para un cierto método, sino que la implementación vendrá dada por las clases que extiendan la clase actual. Una clase que tenga uno o más métodos “abstract” debe declararse como “abstract” a su vez.
Sellada sealed	Clases	Las clases selladas nos permiten restringir que otras clases pueden extenderse de ellas. Se debe agregar la cláusula “permits” seguida de las sub clases que permitimos sean extendidas. Las sub clases extendidas deben usar los modificadores “final”, “sealed” o “non-sealed”.
No sellada non-sealed	Sub clases	Una sub clase permitida puede aplicar el modificar “non-sealed” de manera que revierte en su propia jerarquía el “sellado” de la clase padre, y abriendo la extensión a otros clases desconocidas por la clase padre.

Modificador de Comportamiento Final



Clase con modificador final.
No puede crearse una subclase de la clase Perro.

Método con modificador final.
No puede sobrescribirse en una subclase de la clase GatoMontes.

Modificador de Comportamiento Abstracto


```
abstract class Animal {
    abstract public void hacerRuido();
}
sealed class Gato extends Animal permits GatoMontes {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Miau...");
    }
}
```

Clase con modificador abstract. Significa que tiene por lo menos un método con modificador abstract.

Método con modificador abstract. Sin implementación.

Método que implementa el comportamiento del método hacerRuido.

Modificador de Comportamiento Sellado y No sellado

idat

```
sealed class Gato extends Animal permits GatoMontes {
    @Override
    public void hacerRuido() {
        System.out.println( x: "Miau...");
    }
}
non-sealed class GatoMontes extends Gato {
    @Override
    final public void hacerRuido() {
        System.out.println( x: "Wuiggg...");
    }
}
```

Clase con modificador sealed. Permite crear la subclase GatoMontes.

También puede ser final y sealed.

Subclase con modificador non-sealed. Revierte el sellado de la clase padre.

Modificador de Acceso Protegido

idat

✓ **Recordando:**

Modificador de acceso	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
Público - public	X	X	X	X
Protegido - protected	X	X	X	
Sin modificador de acceso (predeterminado)	X	X		
Privado - private	X			

Modificador de Acceso Protegido (cont.)

idat

- ✓ Una superclase o clase padre declara un miembro (atributo y/o método) con modificador de acceso protegido (protected) para permitir el acceso al miembro desde una clase que se encuentra en el mismo paquete y desde el interior de sus subclases o clases hijas.

```
package segundopaquete;

public class AccesoSP {
    protected void SaludarProtegidoSP() {
        System.out.println("Método Protegido de una Clase de otro paquete. "
                + "Paquete: segundopaquete Clase: AccesoSP");
    }
}
```

Modificador de Acceso Protegido (cont.)


```java
package desarrollo.de.aplicaciones;

import segundopaquete.AccesoSP;

public class ModificadoresDeAcceso extends AccesoSP {
    public static void main(String[] args) {
        ModificadoresDeAcceso ma = new ModificadoresDeAcceso();
        ma.SaludarProtegidoSP();
    }
}
```

```
Método Protegido de una Clase de otro paquete. Paquete: segundopaquete Clase: AccesoSP
```

Aplicación

Revisar ejemplos y realizar ejercicios prácticos

idat

Término

Indicaciones generales y/o Resumen de Sesión

idat

Resumen de Sesión

idat

- Herencia
- Polimorfismo
- Jerarquía de Clases
- Referencia super
- Modificadores
- Ejemplos y ejercicios

GRACIAS