Programación orientada a objetos

Capítulo 10

Más técnicas de abstracción

Tema 10: Más técnicas de abstracción. Semana 10	
1- Simulaciones. 2- El modelo de simulación predador-presa. 3- Clases abstractas. 4- Más métodos abstractos. 5- Herencia múltiple. 6- Interfaces. 7- Resumen de herencia.	1- Estudiar el capítulo 10 del libro base para la "Unidad Didáctica II". 2- Realizar los ejercicios correspondientes del libro base. 3- Realizar los ejercicios resueltos en examines de años anteriores en los que se utilice la herencia.

Método abstracto

La definición de un **método abstracto** consiste en la signatura de un método sin su correspondiente cuerpo. Se indica mediante la palabra clave **abstract**.

Un método abstracto se caracteriza por dos detalles:

- Está precedido por la palabra clave `abstract`.
- No tiene cuerpo y su encabezado termina con un punto y coma.

```
abstract public void actuar (Campo campoActual,
        Campo campoActualizado,
        List<Animal> animalesNuevos);
```

```
obj instanceof MiClase
```

Clases abstractas

- No se creará ninguna instancia de clases abstractas
- Sólo las clases abstractas pueden tener métodos abstractos
- Las clases abstractas con métodos abstractos fuerzan a las subclases a sobreescribir una implementación de aquellos métodos declarados abstractos

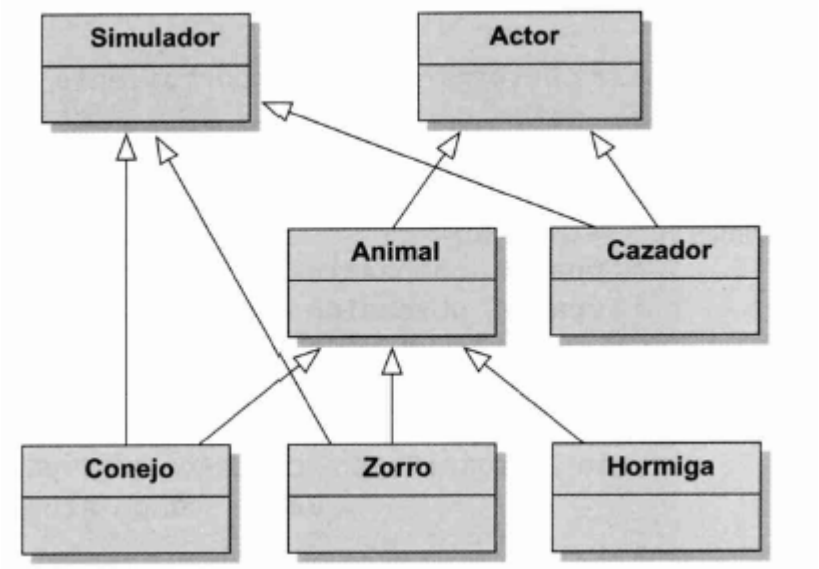
Una **clase abstracta** es una clase de la que no se pretende crear instancias. Su propósito es servir como superclase a otras clases. Las clases abstractas pueden contener métodos abstractos.

Subclases abstractas. Para que una subclase de una clase abstracta se convierta en una subclase concreta, debe proveer las implementaciones de todos los métodos abstractos heredados. De lo contrario, será propiamente abstracta.

```
public abstract class Animal
{
   // Se omiten los campos
   /*
   * Hace que este animal actúe, es decir: hace que haga
lo que quiera
   * o necesita hacer.
   * @param campoActual El campo que ocupa actualmente
   * @param campoActualizado El campo al que se trasladará
   * @param animalesNuevos Una lista para agregar los
animales recién
         *                                    nacidos.
*/
abstract public void actuar (Campo campoActual,
                             Campo campoActualizado,
                             List<Animal> animalesNuevos);
// Se omiten los restantes métodos
}
```

Herencia múltiple

Herencia múltiple. Una situación en la que una clase deriva de más de una superclase se denomina herencia múltiple.



El escenario aquí presentado usa una estructura que se conoce como *herencia múltiple*. La herencia múltiple existe en los casos en los que una clase deriva de más de una superclase. En consecuencia, la subclase tiene todas las características de ambas superclases y aquellas definidas en la subclase propiamente dicha.

Interfaces

Una **interfaz** en Java es la especificación de un tipo (bajo la forma de un nombre de tipo y un conjunto de métodos) que no define ninguna implementación para los métodos.

- En el encabezado de la declaración se usa la palabra clave `interface` en lugar de `class`.
- Todos los métodos de una interfaz son abstractos; no se permiten métodos con cuerpo. No es necesaria la palabra clave `abstract`.
- Las interfaces no contienen ningún constructor.
- Todas las signaturas de los métodos de una interfaz tienen visibilidad pública. No es necesario declarar la visibilidad: por ejemplo, no es necesario que cada método contenga la palabra clave `public`.
- En una interfaz, sólo se permiten los campos constantes (campo público, estático y final). Pueden omitirse las palabras clave `public`, `static` y `final` pero todos los campos, igualmente, serán tratados como públicos, estáticos y finales.

Una clase puede derivar de una interfaz de la misma manera en que deriva de una clase. Sin embargo, Java utiliza una palabra clave diferente, `implements`, para la herencia a partir de interfaces.

Se dice que una clase *implementa* una interfaz si incluye una *cláusula implements* en su encabezado. Por ejemplo:

```
public class Zorro extends Animal implements Dibujable
{
   ...
}
```

Como en este caso, si una clase extiende a una clase e implementa una interfaz, entonces la cláusula `extends` debe escribirse primero en el encabezado de la clase.

Herencia múltiple de interface

- Java permite que cada clase extienda como máximo a otra clase, pero permite que una clase implemente cualquier número de interface (además de la posibilidad de extender una clase)

```
public class Cazador implements Actor, Dibujable
{
   ...
}
```

Términos introducidos en este capítulo

método abstracto, clase abstracta, clase concreta, herencia múltiple, interfaz (construcción Java), implementa

Resumen de conceptos

- **método abstracto** Una definición de un método abstracto consiste en una signatura de método sin un cuerpo. Se marca con la palabra clave `abstract`.
- **clase abstracta** Una clase abstracta es una clase de la que no se tiene intención de crear instancias. Su propósito es servir como una superclase a otras clases. Las clases abstractas pueden contener métodos abstractos.
- **subclases abstractas** Para que una subclase de una clase abstracta se vuelva concreta, debe proveer implementaciones para todos los métodos abstractos heredados; de lo contrario, es propiamente abstracta.
- **herencia múltiple** Una situación en la que una clase deriva de más de una superclase se denomina herencia múltiple.
- **interfaz** Una interfaz en Java es la especificación de un tipo (bajo la forma de un nombre de tipo y un conjunto de métodos) que no define ninguna implementación para ningún método.