7.F. Polimorfismo.

1. Polimorfismo.

1.2. Ligadura dinámica.

La conexión que tiene lugar durante una llamada a un método suele ser llamada **ligadura**, **vinculación** o **enlace** (en inglés **binding**). Si esta **vinculación** se lleva a cabo durante el proceso de compilación, se le suele llamar **ligadura estática** (también conocido como **vinculación temprana**). En los lenguajes tradicionales, no orientados a objetos, ésta es la única forma de poder resolver la **ligadura** (en **tiempo de compilación**). Sin embargo, en los **lenguajes orientados a objetos** existe otra posibilidad: la **ligadura dinámica** (también conocida como **vinculación tardía, enlace tardío** o **late binding**).

La **ligadura dinámica** hace posible que sea el **tipo de objeto** instanciado (obtenido mediante el **constructor** finalmente utilizado para crear el objeto) y no el **tipo de la referencia** (el tipo indicado en la declaración de la variable que apuntará al objeto) lo que determine qué versión del método va a ser invocada. El **tipo de objeto** al que apunta la variable de tipo referencia sólo podrá ser conocido durante la **ejecución** del programa y por eso el **polimorfismo** necesita la **ligadura dinámica**.

En el ejemplo anterior de la clase **X** y sus **subclases A** y **B**, la llamada al método **m** sólo puede resolverse mediante ligadura dinámica, pues es imposible saber en tiempo de compilación si el método **m** que debe ser invocado será el definido en la subclase **A** o el definido en la subclase B:

```
// Llamada al método m (sin saber si será el método m de A o de B).
```

```
obj.m ()  // Esta llamada será resuelta en tiempo de ejecución (ligadura dinámica)
```

Ejercicio resuelto

Imagínate una clase que represente a **instrumento musical** genérico (**Instrumento**) y dos subclases que representen tipos de instrumentos específicos (por ejemplo **Flauta** y **Piano**). Todas las clases tendrán un método **tocarNota**, que será específico para cada subclase.

Haz un pequeño programa de ejemplo en Java que utilice el **polimorfismo** (referencias a la **superclase** que se convierten en instancias específicas de **subclases**) y la **ligadura dinámica** (llamadas a un método que aún no están resueltas en **tiempo de compilación**) con estas clases que representan instrumentos musicales. Puedes implementar el método **tocarNota** mediante la escritura de un mensaje en pantalla.

Solución:

La clase **Instrumento** podría tener un único método (**tocarNota**):

```
public abstract class Instrumento {
```

```
    public void tocarNota (String nota) {
```

```
        System.out.printf ("Instrumento: tocar nota %s.\n", nota);
```

```
    }
```

```
}
```

En el caso de las clases **Piano** y **Flauta** puede ser similar, heredando de **Instrumento** y redefiniendo el método **tocarNota**:

```
public class Flauta extends Instrumento {
```

```
    @Override
```

```
    public void tocarNota (String nota) {
```

```
        System.out.printf ("Flauta: tocar nota %s.\n", nota);
```

```
    }
```

```
}
```

```

```

```

```

```
public class Piano extends Instrumento {
```

```
    @Override
```

```
    public void tocarNota (String nota) {
        System.out.printf ("Piano: tocar nota %s.\n", nota);
    }
}
```

A la hora de declarar una **referencia** a un objeto de tipo instrumento, utilizamos la **superclase** (`Instrumento`):

```
Instrumento instrumento1; // Ejemplo de objeto polimórfico (podrá ser Piano o Flauta)
```

Sin embargo, a la hora de instanciar el objeto, utilizamos el **constructor** de alguna de sus **subclases** (`Piano, Flauta`, etc.):

```
  if (<condición>) {
            // Ejemplo de objeto polimórfico (en este caso va adquirir forma de Piano)
            instrumento1= new Piano ();
        }
        else if (<condición>) {
            // Ejemplo de objeto polimórfico (en este caso va adquirir forma de Flauta)
            instrumento1=  new Flauta ();
        } else {
            ...
        }
```

Finalmente, a la hora de invocar el método `tocarNota`, no sabremos a qué versión (de qué **subclase**) de `tocarNota` se estará llamando, pues dependerá del tipo de objeto (**subclase**) que se haya instanciado. Se estará utilizando por tanto la **ligadura dinámica**:

```
   // Interpretamos una nota con el objeto instrumento1
   // No sabemos si se ejecutará el método tocarNota de Piano o de Flauta
  // (dependerá de la ejecución)
        instrumento1.tocarNota ("do");   // Ejemplo de ligadura dinámica (
```

Solución completa (ficheros):

Instrumento.java

```
package ejemplopolimorfismoinstrumentos;
/**
 *
 * Clase Instrumento
 */
public abstract class Instrumento {
   public void tocarNota (String nota) {
        System.out.printf ("Instrumento: tocar nota %s.\n", nota);
    }
}
```

Flauta.java

```
package ejemplopolimorfismoinstrumentos;

/**
 *
 * Clase Flauta
 */
public class Flauta extends Instrumento {

    @Override
    public void tocarNota (String nota) {
        System.out.printf ("Flauta: tocar nota %s.\n", nota);
    }
}
```

Piano.java

```
package ejemplopolimorfismoinstrumentos;

/**
 *
 * Clase Piano
 */
public class Piano extends Instrumento {

    @Override
    public void tocarNota (String nota) {
        System.out.printf ("Piano: tocar nota %s.\n", nota);
    }
}
```

EjemploPolimorfismoInstrumentos.java

```
/*
 * Ejemplo de polimorfismo y ligadura dinámica
 */
package ejemplopolimorfismoinstrumentos;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class EjemploPolimorfismoInstrumentos {

    /**
     * @param args the command line arguments
     */
```

```
public static void main(String[] args) {
     String tipo= null;
     String nota= null;
     Instrumento instrumento1= null;
    // ¿Flauta o Piano?
    do {
        System.out.println("Elija instrumento: flauta(F) o piano(P): ");
        try {
            tipo= lecturaTeclado();
        }
        catch (Exception e) {
            System.err.println(e.getMessage());
        }
        if (tipo.equals("P") || tipo.equals("p")) tipo="piano";
        else if (tipo.equals("F") || tipo.equals("f")) tipo="flauta";
        else tipo="X";

    } while (tipo.equals("X"));

    // Nota musical
    System.out.println("Escriba nota musical: ");
    try {
        nota= lecturaTeclado();
    }
    catch (Exception e) {
        System.err.println(e.getMessage());
    }

    // Creación del objeto instrumento1 (desconocido en tiempo de compilación)
    // Sabemos que será subclase de Instrumento, pero no sabemos si será Flauta o Piano
    // (dependerá de la ejecución)
    if (tipo.equals("piano")) {
        instrumento1= new Piano (); // Ejemplo de objeto polimórfico (puede ser Piano o Flauta)
    }
    else if (tipo.equals("flauta")) {
        instrumento1=  new Flauta ();  // Ejemplo de objeto polimórfico (puede ser Piano o Flauta)
    } else {
    }
```

```
        // Interpretamos una nota con el objeto instrumento1
        // No sabemos si se ejecutará el método tocarNota de Piano o de Faluta
        // (dependerá de la ejecución)
        instrumento1.tocarNota(nota);   // Ejemplo de ligadura dinámica (tiempo de ejecución)
     }

    //---------------------------------------------------------------
    // MÉTODO lecturaTeclado: Captura de una cadena de teclado
    //---------------------------------------------------------------
    private static String lecturaTeclado () throws Exception {
        try {
            InputStreamReader inputStreamReader = new InputStreamReader(System.in);
            BufferedReader reader = new BufferedReader(inputStreamReader);
            String line = reader.readLine();
            return line;
        }
        catch (Exception e) {
            throw e;
        }
    }
}
```