


| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 132/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Guía práctica de estudio 10: Polimorfismo (1ra parte)




Elaborado por:

M.C. M. Angélica Nakayama C.
Ing. Jorge A. Solano Gálvez

Autorizado por:

M.C. Alejandro Velázquez Mena

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 133/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Guía práctica de estudio 10: Polimorfismo (1ra parte)

Objetivo:

Implementar el concepto de polimorfismo en un lenguaje de programación orientado a objetos.

Actividades:

- Crear una jerarquía de clases.
- Crear una referencia de la clase base más general.
- Crear instancias de diferentes clases derivadas.


Introducción


El término **polimorfismo** es constantemente referido como uno de los pilares de la programación orientada a objetos (junto con la Abstracción, el Encapsulamiento y la Herencia).

El término **polimorfismo** es una palabra de origen griego que significa **muchas formas**. En la programación orientada a objetos se refiere a la **propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos**.

El **polimorfismo** consiste en conseguir que un objeto de una clase se comporte como un objeto de cualquiera de sus subclases. Se puede aplicar tanto a métodos como a tipos de datos. Los métodos pueden evaluar y ser aplicados a diferentes tipos de datos de manera indistinta. Los tipos polimórficos son tipos de datos que contienen al menos un elemento cuyo tipo no está especificado.

NOTA: En esta guía se tomará como caso de estudio el lenguaje de programación JAVA, sin embargo, queda a criterio del profesor el uso de éste u otro lenguaje orientado a objetos.

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 134/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 135/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Polimorfismo

Polimorfismo se refiere a la habilidad de **tener diferentes formas**. El término **IS-A** se refiere a la pertenencia de un objeto con un tipo, es decir, si se crea una instancia de tipo A, se dice que el objeto creado es un A.


El polimorfismo se puede clasificar en dos grandes grupos:


- Polimorfismo dinámico (o paramétrico): es aquel en el que **no se especifica el tipo de datos sobre el que se trabaja** y, por ende, se puede utilizar todo tipo de datos compatible. Este tipo de polimorfismo también se conoce como programación genérica.
- Polimorfismo estático (o ad hoc): es aquel en el que **los tipos de datos que se pueden utilizar deben ser especificados** de manera explícita antes de ser utilizados.

En Java, cualquier objeto que pueda comportarse como más de un **IS-A** (es un) puede ser considerado **polimórfico**. Por lo tanto, todos los objetos en Java pueden ser considerados polimórficos porque todos se pueden comportar como objetos de su propio tipo y como objetos de la clase *Object*.

Por otro lado, debido a que la única manera de acceder a un objeto durante su tiempo de vida es a través de su referencia, existen algunos puntos clave que se deben recordar sobre las mismas:

- Una referencia puede ser solo de un tipo y, una vez declarado, el tipo no puede ser cambiado.
- Una referencia es una variable, por lo tanto, ésta puede ser reasignada a otros objetos (a menos que la referencia sea declarada como *final*).
- El tipo de una referencia determina los métodos que pueden ser invocados del objeto al que referencia, es decir, solo se pueden ejecutar los métodos definidos en el tipo de la referencia.
- A una referencia se le puede asignar cualquier objeto que sea del mismo tipo con el que fue declarada la referencia o de algún subtipo (**Polimorfismo**).

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 136/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 137/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Dada la siguiente jerarquía de clases:

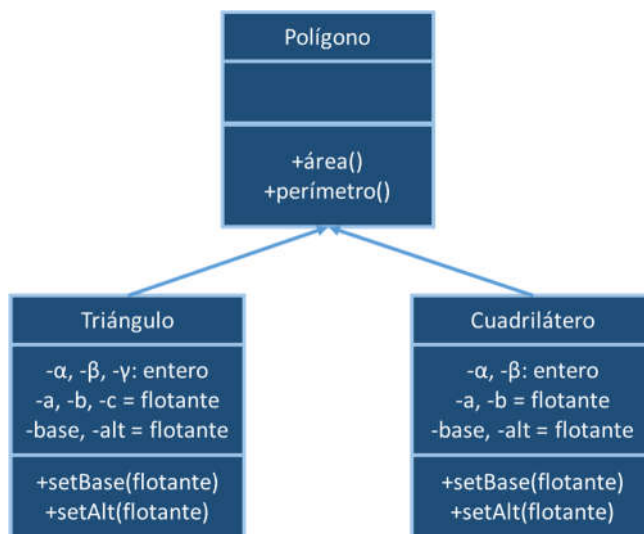


Figura 1. Jerarquía de clases


Su codificación sería la siguiente:

```

public class Poligono {
    public double area(){
        return 0d;
    }

    public double perimetro(){
        return 0d;
    }

    public String toString(){
        return "Poligono";
    }
}
  
```

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 138/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

```
public class Cuadrilatero extends Poligono {
    private int alfa, beta;
    private float a, b;
    private float base, altura;

    @Override
    public String toString(){
        return "Cuadrilátero";
    }

    public Boolean tieneLadosParalelos(){
        return true;
    }
}
```

```
public class Triangulo extends Poligono {
    float sideA, sideB, sideC;

    @Override
    public String toString(){
        return "Triángulo";
    }

    public Boolean tieneLadosParalelos(){
        return false;
    }
}
```


La jerarquía de clases anterior se puede probar a través de la siguiente clase:

```
public class PruebaFigurasGeometricas {
    public static void main (String [] args){
        Poligono poligono = new Poligono();
        // Polígono puede comportarse como Objeto
        Object objeto = poligono;
        System.out.println(objeto);

        // Una referencia puede ser reasignada a otros objetos
        poligono = new Triangulo();
        // Solo se pueden ejecutar los métodos que están definidos
        // en la referencia, sin embargo, se ejecutarán como están
        // implementados en la instancia.
        System.out.println(poligono);

        poligono = new Cuadrilatero();
        // El método toString se puede ejecutar porque está definido
        // en Polígono, sin embargo, se va a ejecutar como está
        // implementado en la instancia (Cuadrilatero o Triangulo).
        System.out.println(poligono);

        // El método tieneLadosParalelos no está definido en Polígono,
        // por lo que la siguiente instrucción marcaría un error:
        //System.out.println(poligono.tieneLadosParalelos());
    }
}
```

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 139/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Cast o moldeo de objetos

Es posible convertir un objeto de un tipo más general a uno más específico mientras exista una relación de herencia a través de un cast o moldeo. La única restricción es que el objeto sea de un tipo más específico.

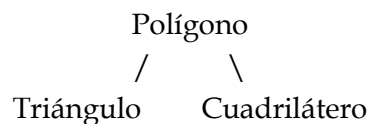
Para validar si el objeto al que apunta la referencia es de un tipo en específico se utiliza la palabra reservada `instanceof` de la siguiente manera:

```
ref instanceof Triangulo
```

En este caso se verifica si la referencia `ref` está apuntando en el heap a un objeto del tipo `Triangulo`. La validación anterior devuelve verdadero o falso.

Existen dos tipos de cast, el cast hacia arriba (up-casting) y el cast hacia abajo (down-casting). El up-casting es explícito, es decir, no es obligatorio indicar a qué tipo de dato se quiere convertir el objeto, se infiere a partir de la referencia. Por otra parte, el down-casting debe ser implícito, es decir, se debe indicar a qué tipo se quiere moldear el objeto.

Por ejemplo, para la jerarquía de clases de `Polígono`, `Triángulo` y `Cuadrilátero` se tiene:




A una referencia tipo `Polígono` se puede asignar una instancia de `Polígono`, de `Triángulo` o de `Cuadrilátero` sin necesidad de hacer un casting explícito, es decir:

```

Poligono p = new Poligono();
Poligono fig = new Triangulo();
Poligono pol = new Cuadrilatero();

```


| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 140/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Sin embargo, a partir de estas referencias creadas, para recuperar los objetos se debe asignar la referencia tipo polígono a una referencia más específica, realizando un cast de manera explícita, es decir:

Triangulo t = (Triangulo) fig;
Cuadrilatero c = (Cuadrilatero) pol;


```
public class PruebaFigurasGeometricas {
    public static void main (String [] args){
        System.out.println("\nPoligono p = new Triangulo();\n");
        Poligono p = new Triangulo();
        System.out.println(p.toString());
        if (p instanceof Triangulo){
            // Cast de Polígono a Triángulo
            System.out.println("\nTriangulo t = (Triangulo)p\n");
            Triangulo t = (Triangulo)p;
            System.out.println(t.tieneLadosParalelos());
        }

        System.out.println("\np = new Cuadrilatero();\n");

        p = new Cuadrilatero();
        System.out.println(p.toString());
        if (p instanceof Cuadrilatero){
            // Cast de Polígono a Cuadrilátero
            System.out.println("\nCuadrilatero q = (Cuadrilatero)p\n");
            Cuadrilatero q = (Cuadrilatero)p;
            System.out.println(q.tieneLadosParalelos());
        }
    }
}
```

Polimorfismo en métodos

Un método puede recibir cualquier tipo y número de datos como parámetros y puede devolver cualquier tipo de dato. A este concepto se le conoce también como **polimorfismo en métodos**.

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 141/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Cuando el parámetro definido es una referencia a una clase, el método es capaz de recibir un objeto de ese tipo o de **cualquier subtipo** de esa clase. Cuando el valor de retorno definido es una referencia a una clase, el método es capaz de devolver un objeto de ese tipo o de **cualquier subtipo** de esa clase.


```

public class PolimorfismoMetodos {
    public static void main (String [] args){
        Poligono fig1 = crearFigura(1);
        Poligono fig2 = crearFigura(2);
        Poligono fig3 = crearFigura(3);
        imprimirTipoPoligono(fig3);
        imprimirTipoPoligono(fig2);
        imprimirTipoPoligono(fig1);
    }

    // Crear figura recibe como parámetro un entero para
    // indicar el tipo de objeto que se va a crear:
    // 1 polígono, 2 triángulo, 3 cuadrilátero
    public static Poligono crearFigura(int id){
        Poligono pol = null;
        switch(id) {
            case 1:
                pol = new Poligono();
                break;
            case 2:
                pol = new Triangulo();
                break;
            case 3:
                pol = new Cuadrilatero();
                break;
        }
        return pol;
    }

    public static void imprimirTipoPoligono(Poligono p){
        if (p instanceof Triangulo){
            System.out.println("p es una instancia de Triángulo");
        } else {
            if (p instanceof Cuadrilatero){
                System.out.println("p es una instancia de Cuadrilatero");
            } else {
                System.out.println("p es una instancia de Polígono");
            }
        }
    }
}

```

| | | | |
|--|---|--|---------------------|
|  | Manual de prácticas del Laboratorio de Modelos de programación orientada a objetos | Código: | MADO-21 |
| | | Versión: | 02 |
| | | Página | 142/166 |
| | | Sección ISO | 8.3 |
| | | Fecha de emisión | 14 de junio de 2018 |
| Facultad de Ingeniería | | Área/Departamento: Laboratorio de computación salas A y B | |
| La impresión de este documento es una copia no controlada | | | |

Bibliografía

Barnes David, Kölling Michael

Programación Orientada a Objetos con Java.

Tercera Edición.

Madrid

Pearson Educación, 2007

Deitel Paul, Deitel Harvey.

Como programar en Java

Septima Edición.

México

Pearson Educación, 2008

Martín, Antonio

Programador Certificado Java 2.

Segunda Edición.

México

Alfaomega Grupo Editor, 2008

Dean John, Dean Raymond.

Introducción a la programación con Java

Primera Edición.

México

Mc Graw Hill, 2009