# ADSI

## Clases y Objetos Java-2 - POO


Instructor: Gustavo Adolfo Rodríguez Q.

garodriguez335@misena.edu.co

ADSI

# CLASES Y OBJETOS CON JAVA SEGUNDA PARTE

## 1. OBJETOS

Un programa típico en Java crea una gran variedad de objetos durante su ejecución, los cuales como sabemos, interactúan entre ellos mediante la invocación de métodos. A través de estas interacciones entre los objetos un programa puede realizar tareas tales como implementar una interfaz gráfica de usuario (GUI por sus siglas en inglés), recibir o enviar información en la red.

A continuación se muestra un ejemplo de una clase ejecutable llamada DemoObjetos, dentro del método ***main*** se realiza la declaración, construcción e invocación de métodos de un objeto de la clase Point y de dos objetos de la clase Rectangle.

A través de la ejecución de método ***main*** se manipulan los objetos cambiando los valores de sus atributos e invocando algunos de sus métodos.

```
public class DemoObjetos {
    public static void main(String[] args) {
        //Declarar un objeto de la clase Point y dos de la clase Rectangle
        Point punto;
        Rectangle rectangulo1;
        Rectangle rectangulo2;

        //Construir el objeto de la clase Point utilizando el constructor
        //que requiere dos argumentos enteros
        punto = new Point(20, 46);

        //Construir los objetos rectangulo1 y rectangulo2 utilizando el
        //constructor que requier el ancho y el alto como argumentos
        rectangulo1 = new Rectangle(12, 89);
        rectangulo2 = new Rectangle(10, 20);

        //fijar la ubicacion del rectangulo1 utilizando el objeto punto
        rectangulo1.setLocation(punto);

        //modificar las coordenadas del objeto punto.
        punto.setLocation(55,6);

        //fijar la ubicacion del rectangulo2 utilizando el objeto punto
        rectangulo2.setLocation(punto);

        //imprimir los atributos del los rectangulos
        System.out.println("Rect1 (X) " + rectangulo1.getLocation().getX());
        System.out.println("Rect1 (Y) " + rectangulo1.getLocation().getY());
        System.out.println("Centro Rec2  (X)" + rectangulo2.getCenterX());
        System.out.println("Centro Rec2  (Y)" + rectangulo2.getCenterY());
    }
}
```

A continuación se muestra la salida obtenido tras la ejecución del programa.

```
Rect1 (X) 20.0
Rect1 (Y) 46.0
Centro Rec2  (X)60.0
Centro Rec2  (Y)16.0
```

Las siguientes secciones utilizan el ejemplo anterior para describir el ciclo de vida de un objeto dentro de un programa. A partir de él, podrá aprender cómo crear y utilizar objetos en sus propios programas.

## 2. CREACION DE OBJETOS

Como se ha mencionado anteriormente, una clase es una plantilla para la creación de objetos, un objeto es creado a partir de una clase. Cada una de las siguientes líneas crea un objeto de una clase particular y asigna ese valor a una variable.

```
punto = new Point(20, 46);
rectangulo1 = new Rectangle(12, 89);
rectangulo2 = new Rectangle(10, 20);
```

Para poder utilizar un objeto de una clase, o lo que es lo mismo decir, una instancia de una clase se debe realizar tres pasos:

- **Declaración del objeto.**
- **Instanciación del objeto.**
- **Inicialización del objeto.**

### 2.1.Declaración del Objeto

Para realizar la declaración de un objeto en Java se debe seguir esta estructura:

```
ClasePerteneciente   nombreDelObjeto
```

Sentencias de este tipo le especifican al compilador que se va a utilizar el nombre *nombreDelObjeto* para referirnos a un objeto de la clase *ClasePerteneciente.* Ejemplos de esto se pueden apreciar en las siguientes líneas de código donde se realiza la declaración de un objeto de la clase Point cuyo nombre es punto y la declaración de dos objetos de la clase Rectangle cuyos nombres son rectangulo1 y rectangulo2.

```
Point punto;
Rectangle rectangulo1;
Rectangle rectangulo2;
```

Hasta este momento los objetos punto, rectangulo1 y rectangulo2 han sido simplemente declarados para su posterior utilización, pero no poseen un valor definido. En estos momentos los objetos tienen un valor nulo (null en inglés). Para poder utilizar los objetos que hemos declarados es necesario realizar una instanciación de la clase como se explica en la siguiente sección.

## 2.2.Instanciación del Objeto

La instanciación de las clases se realiza mediante la utilización del operador **new**, este operador reserva la memoria necesaria para que el objeto pueda existir y retorno una referencia hacia ese espacio de memoria. Mediante el operador **new** se realiza la invocación de un método constructor de la clase que estamos instanciando.

Seguido del operador **new** debemos especificar el nombre del método constructor que vamos a utilizar. El nombre del constructor provee el nombre de la clase a instanciar.

El operador **new** retorna una referencia hacia el objeto que se acaba de crear, esta referencia es asignada a una variable del tipo apropiado. Por ejemplo, el objeto retornado tras la instanciación de la clase Point se asigna a la variable punto.

```
punto = new Point(20, 46);
```

## 2.3.Inicialización del Objeto

A continuación se muestra el código de la clase Point

```
public class Point {
    public int x = 0;
    public int y = 0;
    //constructor
    public Point(int a, int b) {
        x = a;
        y = b;
    }
}
```

Como se puede observar esta clase tiene un único método constructor, se lo puede reconocer puesto que es un método que tiene el mismo nombre que la clase y no tiene tipo de retorno. El constructor de la clase Point necesita dos argumentos de tipo entero (int a, int b) los valores pasados como argumentos en el método constructor se convertirán en los valores de los atributos **x** y **y** de la clase Point. Esto es lo que se conoce

como inicialización del objeto puesto que los valores de los argumentos pasados en el método constructor se convierten en los valores de los atributos de la clase.

```
punto = new Point(20, 46);
```

Las anteriores líneas de código hacen que se inicialice el objeto punto. Así, los atributos X y Y del objeto punto tendrán los valores 20 y 46 respectivamente.

## 3. UTILIZACION DE OBJETOS

Una vez haya sido creado el objeto mediante la invocación de **new** y el método constructor de la clase, lo más probable es que se quiera utilizar este objeto bien sea modificando los valores de sus atributos o invocando los métodos disponibles.

Para acceder a los atributos de un objeto instancia, debemos utilizar el nombre del objeto seguido de un punto y luego el nombre del atributo. Debemos recordar que estos atributos pueden ser accedidos de esta forma siempre y cuando hayan sido declarados públicos (public en inglés).

La invocación de métodos de un objeto se realiza de forma similar; se debe utilizar el nombre del objeto seguido de un punto y luego el nombre del método a continuación se escribe los argumentos del método encerrados entre paréntesis y separados por comas como se muestra en el siguiente ejemplo.

```
punto.setLocation(55,6);

rectangulo2.setLocation(punto);

rectangulo1.getLocation().getX()
```

## 4. OTROS CONCEPTOS RELACIONADOS A OBJETOS

En esta sección se cubre aspectos relacionados con la utilización de referencias a objetos y la utilización del operador punto (.)

Aprenderemos a:

- Retornar valores desde un método.
- Utilización de **this**
- Variables de clase y variables de instancia.

## 4.1.Retornar valores desde un método

Un método retorna hacia la línea de código desde la cual fue invocado cuando:

- Se completan todas las sentencias dentro del método.
- Se alcanza la sentencia ***return*** o
- Se lanza una excepción (esto se estudiará más adelante)

El tipo de retorno de un método es especificado en la declaración del método. Si un método declara algún tipo de retorno, es necesario agregar dentro de la implementación del método una sentencia ***return*** para retornar un valor adecuado. Si el método ha sido declarado con tipo de retorno **void** no es necesario escribir una sentencia ***return*** dentro del cuerpo del método.

A continuación se muestra un ejemplo:

```
// a method for computing the area of the rectangle
public int getArea() {
    return width * height;
}
```

A continuación se muestra otro ejemplo donde el tipo de retorno no es un tipo de dato primitivo sino un tipo de dato complejo, en este caso, un objeto de la clase Box.

```
public Box obtenerCirculo(){
    Box caja = new Box(10);
    return caja;
}
```

## 4.2.Utilización de *this*.

La palabra reservada ***this*** hace referencia al *objeto actual,* o sea, el objeto desde el cuál fue llamado el constructor o método. Podemos referirnos a un atributo de un objeto instancia utilizando la palabra reservada ***this*.**

Por ejemplo, la clase Point fue escrita así:

```
public class Point {
    public int x = 0;
    public int y = 0;

    //constructor
    public Point(int a, int b) {
        x = a;
        y = b;
    }
}
```

Pero podría haber sido escrita de la siguiente forma utilizando ***this*.**

```
public class Point {
    public int x = 0;
    public int y = 0;

    //constructor
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

Nótese la diferencia en los nombres de los argumentos del método constructor y la inicialización del objeto.

La clase Rectangle ofrece otro ejemplo de la utilización del ***this***

```
public class Rectangle {
    private int x, y;
    private int width, height;

    public Rectangle() {
        this(0, 0, 0, 0);
    }
    public Rectangle(int width, int height) {
        this(0, 0, width, height);
    }
    public Rectangle(int x, int y, int width, int height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }
    ...
}
```

Esta clase tiene una serie de constructores. Cada uno inicializa algunos o todos los atributos de la clase Rectangle. El constructor provee de un valor por defecto para cada argumento cuyo valor no haya sido ingresado como un parámetro del

constructor. Por ejemplo, el constructor sin argumentos hace una invocación a otro constructor que necesita de cuatro parámetros los cuales son fijados en 0. Es importante recordar que el compilador decide cuál método utilizar dependiendo del tipo y cantidad de argumentos especificados en la invocación.

## 4.3.Diferencias entre variables de instancia y variables de Clase.

En esta sección se describe la utilización de la palabra reservada ***static*** para crear métodos y campo o variables que pertenecen a una clase en vez de objeto instancia de la clase.

### 4.3.1. Variables de clase o variables estáticas.

Cuando una serie de objetos son creados a partir de la misma clase, cada uno de ellos tiene una copia con distinta de la misma *variable de instancia*. En el caso de una clase Usuario como la mostrada a continuación cada uno de los objetos creados a partir de esta clase tiene sus propios valores para sus *variables de instancia* nombre, apellido, identificación, y cada una de estas *variables de instancia* son almacenadas en posiciones de memoria RAM distintas.

Algunas veces es necesario tener variables que sean comunes a todos los objetos instancia de la misma clase. Para poder tener una variable que sea común a todos los objetos de la misma clase se debe utilizar la palabra reservada ***static*** . Los atributos marcados con la palabra reservada ***static*** en su declaración son llamados *campos estáticos, variables estáticas, atributos estáticos* o también ***variables de clase.***

Las ***variables de clase*** están asociadas a una clase más que a un objeto particular. Cada objeto instancia de una clase comparte las mismas ***variables de clase***. Un objeto puede manipular la variable de clase, pero cualquier modificación en su valor se refleja en todos los objetos instancia de la clase.

Observe el código mostrado a continuación:

```java
public class Bicicleta {

    private int velocidad;
    private int cambio;
    private int pasajeros;

    public static int bicicletasCreadas = 0;

    public Bicicleta(){
        velocidad = 0;
        cambio = 1;
        pasajeros = 1;

        bicicletasCreadas++;
    }

    public Bicicleta(int velocidad){
        this.velocidad = velocidad;
        this.cambio = 1;
        bicicletasCreadas++;
    }

    public Bicicleta(int velocidad, int cambio, int pasajeros){
        this.velocidad = velocidad;
        this.cambio = cambio;
        this.pasajeros = pasajeros;
        bicicletasCreadas++;
    }

}
```

Como puede observarse, la clase Bicicleta declara un atributo entero llamado *bicicletasCreadas* el cual ha sido marcado con la palabra reservada ***static*** esto convierte a *bicicletasCreadas* en un variable de clase. Más adelante en el código podemos observar que existen tres métodos constructores para la clase Bicicleta. Cada uno de los constructores hace un incremento en el valor de *bicicletasCreadas* , este cambio en el valor de la variable se hace visible para todos los objetos instancia.

Supongamos ahora que a la clase Bicicleta le agregamos un método ***main*** para hacer de esta clase una clase ejecutable como se muestra a continuación.

```java
public static void main(String[] args){
    Bicicleta myBicy = new Bicicleta();
    Bicicleta tuBicy = new Bicicleta(10);
    Bicicleta otraBicy = new Bicicleta(5, 2, 1);

    System.out.println("Bicicletas creadas hasta ahora " + Bicicleta.bicicletasCreadas);
}
```

En este método ***main*** se crean tres instancias diferentes de la clase Bicicleta. Puesto que en todos los constructores de la clase Bicicleta se hace un incremento en el valor de la variable de clase o variable estática *bicicletasCreadas.*

Al final del método ***main*** se imprime en pantalla el valor que tiene la variable estática *bicicletasCreadas.* Nótese que para poder acceder a este valor es necesario hacerlo así:

```
Bicicleta.bicicletasCreadas
```

Es decir, para acceder al valor de una variable de clase es necesario escribir el nombre de la clase, luego un punto y seguido el nombre de la variable de clase.

### 4.3.2. Métodos de clase o métodos estáticos.

El lenguaje de programación Java soporta tanto variables estáticas como métodos estáticos. Los métodos estáticos son aquellos que en su declaración tienen la palabra reservada **static** . Para invocar un método estático se hace mediante el nombre de la clase seguido de un punto y a continuación el nombre del método estático y entre paréntesis y separados por comas se escriben los atributos para el método.

Supongamos que a la clase Bicicleta le agregamos el siguiente método:

```
public static int getBicicletasCreadas(){
    return bicicletasCreadas;
}
```

Este método se convierte en un método estático puesto que ha sido marcado con la palabra reservada ***static*** . Para invocar este método tenemos que hacerlo de la siguiente forma:

```
int creadas = Bicicleta.getBicicletasCreadas();
```

### 4.3.3. Declaración de constantes.

En algunos programas es necesario tener algunos valores constantes, es decir, que no sea posible modificar el valor que se les ha asignado inicialmente.

Para realizar la declaración de constantes en Java se debe utiliza tanto la palabra ***static*** como la palabra ***final***. La palabra ***final*** hace que una variable nunca pueda cambiar su valor inicialmente asignado.

Por ejemplo, en la siguiente línea de código se hace la declaración de una variable de tipo float llamada PI, cuyo valor es una aproximación del número real π de la trigonometría.

```
public static final float PI = 3.14159265358793f;
```

Por convención, las constantes en Java se deben escribir todo con letras mayúsculas separando las palabras con un guion bajo. Las siguientes son declaraciones válidas para constantes en java.

```
private static final int PASAJEROS = 0;
private static final int BUSES_PARQUEADOS = 4;
private static final  int PASAJES_VENDIDOS = 1250;
```

## 5. Diagrama de clase apoyo