

CLASES Y OBJETOS

Clase 4

OBJETOS

- Bloque de construcción de un programa.
- Un componente con una funcionalidad deseada.
- Se le puede decir **QUE** hagan algo sin necesidad de saber **COMO** lo hace.
- Ejemplos:
- Que atributos tienen en común?

• Pedro	Juan	Julia
---------	------	-------
- Que atributos tienen en común?

• 03/05/1986	27/10/1986	30/11/2011
--------------	------------	------------

CLASES

- Objetos con un comportamiento común (todos pueden hacer acciones iguales)
- Las clases sirven para generalizar y englobar a varios objetos del mundo real.

CLASES

- ¿Atributos?
 - Definen lo que son los objetos de la clase.
- ¿Métodos?
 - Definen lo que podemos hacer con los objetos de esa clase.

EJEMPLO CLASE PERSONA

- ¿Atributos?
- ¿Métodos?

EJEMPLO CLASE FECHA

- ¿Atributos?
- ¿Métodos?

INTERFACE PÚBLICA DE UNA CLASE

- String (<https://docs.oracle.com/javase/7/docs/api/index.html?java/lang/String.html>)
 - Atributos
 - Constructores
 - Métodos
- Ejercicio: Trabajar con la cadena " La palabra está en la posición 3"
 - Convertir todo a MAYÚSCULAS.
 - Convertir todo a minúsculas.
 - Reemplazar las "a" por "x"
 - Reemplazar solo la primera "a" por "x".
 - Verificar si existe la cadena "alab" y en que posición de la cadena está
 - Extraer la subcadena "en la posición" de la cadena principal.

CONSTRUCTOR

- Cuando inicializamos un objeto, estamos creando una instancia de la clase. Utilizando ella palabra reservada **new**.
- Al crear una instancia de una clase estamos llamando al constructor de la clase.
- Todas las clases tienen un constructor implícito.

```
public class Persona {  
  
    private String nombre;  
    private String apellido;  
    private Date fechaNacimiento;  
  
    public static void main(String[] args) {  
        | Persona p = new Persona();  
    }  
}
```


CONSTRUCTOR

- Todas las clases tienen un constructor implícito.
- `Persona()` {}
- Si la clase tendrá un solo constructor, entonces no es necesario especificar el constructor.

```
public class Persona {  
  
    private String nombre;  
    private String apellido;  
    private Date fechaNacimiento;  
  
    Persona() {}  
  
    public static void main(String[] args) {  
        Persona p = new Persona();  
    }  
  
}
```

CONSTRUCTOR

- Los objetos de una clase pueden ser inicializados (construidos) de diferentes maneras
- A esto se le llama **sobrecarga de constructores**
- Para hacer referencia a la misma clase donde se está trabajando, se utiliza la palabra reservada **this**.
- Generalmente los constructores deben ser públicos.

```
public class Persona {  
  
    private String nombre;  
    private String apellido;  
    private Date fechaNacimiento;  
  
    Persona(){}  
  
    Persona(String n){  
        this.nombre = n;  
    }  
  
    Persona(String n, String a){  
        this.nombre = n;  
        this.apellido = a;  
    }  
  
    Persona(String n, String a, Date fn){  
        this.nombre = n;  
        this.apellido = a;  
        this.fechaNacimiento = fn;  
    }  
  
    public static void main(String[] args) {  
        Persona p = new Persona();  
  
        Persona p2 = new Persona("Chalo", "Salvador");  
    }  
}
```

UTILIZACIÓN DE LOS MÉTODOS

- getters (Accesores)
 - Devuelven el valor del atributo que corresponda
- setter (Mutadores)
 - Asignan o cambian el valor del atributo que corresponda
- Los getters y setter generalmente son públicos ya que son la "interfaz" de interacción que debe ofrecer la clase a los programadores.

```
public class Persona {  
  
    private String nombre;  
    private String apellido;  
    private Date fechaNacimiento;  
  
    Persona(){}  
  
    Persona(String n){  
        this.nombre = n;  
    }  
  
    Persona(String n, String a){  
        this.nombre = n;  
        this.apellido = a;  
    }  
  
    Persona(String n, String a, Date fn){  
        this.nombre = n;  
        this.apellido = a;  
        this.fechaNacimiento = fn;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public String getApellido() {  
        return apellido;  
    }  
  
    public void setApellido(String apellido) {  
        this.apellido = apellido;  
    }  
  
    public Date getFechaNacimiento() {  
        return fechaNacimiento;  
    }  
  
    public void setFechaNacimiento(Date fechaNacimiento) {  
        this.fechaNacimiento = fechaNacimiento;  
    }  
}
```

EJEMPLO DE OCULTAMIENTO DE DATOS

- Añadir un atributo `cedula` a la clase `persona`
- Añadir getters y setters
- El setter debe siempre validar que el valor que se desea asignar al atributo `cedula` sea una cédula válida.
- Algoritmo de validación de cédula:
<http://telesjimenez.blogspot.com/2011/05/algoritmo-de-verificacion-de-cedula.html>
- El método `validarCedula(String cedula)` ; debe ser privado.

¿QUÉ OBTENEMOS DEL SIGUIENTE CÓDIGO?

```
public class Principal {  
    public static void main(String[] args) {  
  
        Persona p2 = new Persona();  
        Persona p3 = p2;  
  
        p2.setNombre("Chalo");  
        p3.setApellido("Salvador");  
  
        System.out.println("P2 " + p2.getNombre());  
        System.out.println("P3 " + p3.getNombre());  
  
        System.out.println("P2 " + p2.getApellido());  
        System.out.println("P3 " + p3.getApellido());  
    }  
}
```

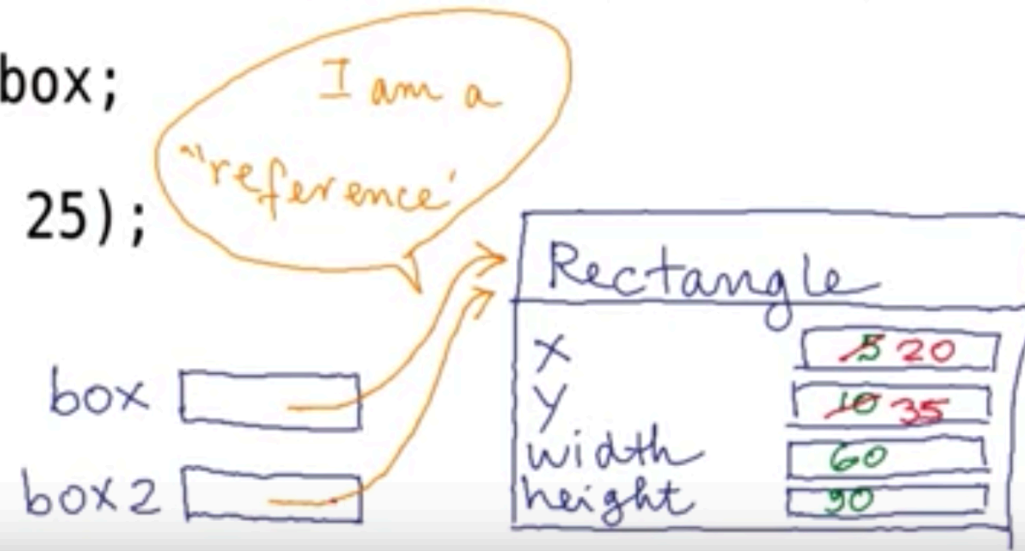
REFERENCIAS A MEMORIA

```
Persona p2 = new Persona();  
Persona p3 = p2;
```

- Esto le dice al programa que p3 apunta a la misma posición de memoria que p2.
- Por lo tanto cuando el contenido de esa posición de memoria cambia, esto afecta a las 2 variables.

OBJETOS COMPARTIDOS

```
Rectangle box = new Rectangle(5, 10, 60, 90);  
Rectangle box2 = box;  
box.translate(15, 25);
```



```
public class CopyingRectangles
{
    public static void main(String[] args)
    {
        Rectangle box = new Rectangle(5, 10, 60, 90);
        Rectangle box2 = box;
        box.setColor(Color.RED);
        box2.setColor(Color.BLUE);
        box.fill();
    }
}
```

¿QUÉ NOS DA EL
SIGUIENTE
CÓDIGO?



¿QUÉ NOS DA EL SIGUIENTE CÓDIGO?

¿Qué valor tiene `greeting` y `greeting2`?

```
String greeting = "Hello, World!";  
String greeting2 = greeting;  
greeting2.toUpperCase();
```

LOS STRING SON INMUTABLES

- No existe métodos que puedan cabiar el valor de un String
- El método toUpperCase(), simplemente retorna otro String con las letras en mayúsculas, pero no afecta al String original.
- Los mismo pasa con los otros métodos de la clase String.

AHORA CON NÚMEROS

¿Cuál es el valor de luckyNumber y luckyNumber2?

```
int luckyNumber = 13;  
int luckyNumber2 = luckyNumber;  
luckyNumber2 = 12;
```


INT NO ES UN OBJETO

- Solo los objetos almacenan las referencias en memoria.
- Los tipos primitivos almacenan el valor real que se le asigna.
- Por lo tanto luckyNumber y luckyNumber2 son independientes uno del otro.

EJERCICIO

- Qué imprimen los siguientes Strings?

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("42 + 7");
        System.out.println(42 + 7);
        System.out.print("Hello");
        System.out.println("World");
    }
}
```



EJERCICIO I

- Crear una clase llamada *Archivo* que permita representar a archivos del computador.
- Definir los atributos y métodos que crea necesarios en la clase.
- Los objetos de esta clase pueden ser inicializados con al menos con el nombre y fecha de creación.
- Declarar 3 objetos de esta clase y setear los valores para nombre, tipo, fecha de creación, contenido.

EJERCICIO 2

- Escribir una programa en Java que permita manejar la información básica de los estudiantes de una universidad. El programa debe brindar información personal del estudiante y además la lista de materias que está está tomando.
- El programa debe crear un arreglo de objetos Estudiante. Cada estudiante en la lista debe ser inicializado con nombre, apellido, cedula, fecha de nacimiento y una lista de materias que se encuentra tomando. Para poder almacenar la lista de materias de cada estudiante, se debe incluir un atributo de tipo `String[]` en la clase Estudiante.
- El programa finalmente debe imprimir en pantalla cuantos estudiantes se encuentran tomando cada materia.