Java

Tema 5

Creando Clases y Objetos

Contenidos

1. Estructura de una clase
2. Constructores
3. Métodos
4. Creando objetos
5. Parámetros
6. Valores de retorno
7. La palabra clave this
8. Método toString()

Estructura de una clase

```
public class NombreDeClase {
//declaración de los ATRIBUTOS (propiedades) o variables de la clase
  private tipoDato nombreDeAtributo;
  private tipoDato nombreDeAtributo2;

//método/s Constructor/es
  modificador nombreDeClase(listaDeParámetros) {
     ……..
  }

//declaración de MÉTODOS
  modificador tipoDatoDevuelto nombreDeMétodo(listaDeParámetros){
    //declaración de las variables locales del método
     tipoDato variableLocal;
     . . .
    //sentencias de control del método
     . . .

    [return valorDevuelto;] //solo si se devuelve un valor
  }

}
```


Estructura de una clase

- El acceso a los atributos y métodos de una clase se logra mediante los *modificadores de acceso*.

  public: Cualquiera los puede usar

  private: Solo se pueden usar desde la propia clase

  Nosotros haremos:

  - Las clases mejor siempre public
  - Los atributos recomendable siempre private, accesibles mediante métodos:
    - getAtributo() → permitimos ver su valor
    - setAtributo(valor)→ permitimos modificar su valor
  - Los métodos pueden ser public ó private

Estructura de una clase: Ejemplo

```
public class Circulo{
   private double  radio;

    public Circulo(double rad){  // constructor
        radio = rad;
    }
    public double getRadio(){   // método obtiene el radio
      return  radio;
    }
    public void setRadio(double valor){ //método modifica el radio
      radio = valor;
    }
    public double area(){      // método calcula el área
      return  Math.PI * radio * radio;
    }
}
```


Atributos

- El tipo de datos de un **atributo** puede ser:
  - Simples o **primitivos**: *int, long, double, boolean, char….*
  - **Clases** de la api de java: *String, Character, Boolean, Integer, …*
  - **Estructurados**: *vectores, matrices, collection (List, HashSet, HashMap…), …*
  - **Clases definidas** por vosotros: *Persona, Circulo, Cliente, Alumno,…..*

```
public class CuentaBancaria{
   long              numCuenta;
   String            tipoCuenta;
   Persona           titular;
   Set<Persona>      listaAutorizados = new HashSet<>();

}
```

Constructores

El **Constructor** de una clase es un método:

- sin valor de retorno,
- tiene el mismo nombre que su clase
- sirve para crear objetos (instancias) de esa clase y para asignar valores iníciales (pueden venir de los parámetros) a los atributos de la clase de ese objeto creado.

```
public class Circulo{
   private double  radio;

   public Circulo(double rad){
       radio = rad;
   }

}
```


Métodos

- Un método es una función/procedimiento de una clase que puede ser llamado/ invocado/utilizado/ejecutado, y realiza una acción bien definida.

- Puede o no recibir parámetros de entrada *(son datos que no tenemos dentro de la clase)* , ejecuta un bloque de sentencias y devuelve o no un valor.

Métodos

- Cada atributo puede tener ninguno, uno o dos métodos asociados, según las acciones que permitamos sobre ese atributo:
  - *getAtributo()* devuelve su valor:

```
double getRadio() { return radio;}
```

  - *setAtributo(nuevovalor)* que modifica el atributo con el valor proporcionado, puede no devolver nada (void) ó devolver algo (por ejemplo boolean para indicar si ha ido bien o no la modificación):

```
void    setRadio(double valor){ radio=valor}

boolean setRadio(double valor){
     boolean respuesta=false;
     if( valor > 2) { //ejemplo de condición a comprobar
        radio=valor;
        Respuesta = true;
     }
     return respuesta;
}
```


Creando objetos

- Para crear un objeto/instancia de una clase, hace falta declarar una variable e invocar al operador **new** y un **constructor** de la clase.

NombreDeClase variableReferencia = **new** NombreDeClase (parámetros);

  - El operador new crea un nuevo objeto de la clase a partir del constructor NombreDeClase y guarda una referencia del objeto en variableObjeto

Valor del radio del objeto círculo que se quiere crear.

```
Ejemplo:
      Circulo c1 = new Circulo(5.0);
      Circulo c2;    //c2 tendría el valor null
      c2 = new Circulo(3.5);
```

  - El valor **null** significa que el objeto NO ha sido creado. Si intentamos usar esa variable para acceder a un atributo o método saltaría una excepción **NullPointerException**

Parámetros

- Para usar métodos lo hacemos a través de la variable referencia que apunta al obejto:

  variableReferencia.nombreDeMétodo(listaDeArgumentos)

  - listaDeArgumentos: lista de valores/variables, separados por comas, que se asociarán a cada parámetro del método. Los valores deben coincidir en orden y tipo de datos con los parámetros del método. Si el método no tiene parámetros la lista estará vacía.

Ejemplo:

Al llamar (ejecutar/ invocar/usar)

argumento

```
c1.setRadio(5.5);
```

argumento

```
c1.setRadio(radioNuevo);
```

Al definirlo, declararlo

parámetro

```
setRadio(double valor){
   radio = valor;
}
```


Parámetros

- Podemos declarar métodos con varios parámetros

```
public class Matematicas {

   public double sumar(double a, double b){
     double suma;

     suma = a + b;

     return suma;
   }
}
```

Parámetros

- Métodos con parámetros que sean objetos, de su misma clase o de otra.

```
public class Circulo {
    private double radio;
    public Circulo(double rad) {
        radio = rad;
    }
    public double getRadio(){return radio;}
    public boolean igual(Circulo c){  //parámetro objeto círculo
        return (radio == c.getRadio();
    }
}
```

```
public class parametroObjeto {
   public static void main ( String args[ ] ) {
     Circulo c1 = new Circulo( 5.0 );
     Circulo c2 = new Circulo( 5.0 );
     if ( c1.igual(c2)){
       System.out.println ("Iguales"};
     }
   }
}
```


Valor de retorno

- El método indica el tipo de datos del valor que retorna/devuelve
- Si no retorna nada se indica con la palabra void

```
public class Circulo{
    private double radio;

    public void setRadio(double valor){
        radio = valor;
    }

    public double area(){
       return  (Math.PI * radio * radio);
    }
}
```

NO retorna nada

Tipo de valor retornado

La palabra reservada this

- La palabra this, representa al propio objeto, se utiliza para hacer referencia a sus atributos y métodos.

- Cuando un parámetro de un método tiene el mismo nombre que un atributo de la clase, el parámetro oculta a la variable de la clase.

```
// Colisión de nombres. Esto es INCOCORRECTO
  Circulo(double radio){
    radio = radio;
  }
```

- Si queremos usar parámetros con nombres iguales a las variables de la clase debemos usar **this**.

```
// Uso de this para resolver colisiones. CORRECTO
        Circulo(double radio){
          this.radio = radio;
        }
```


Método toString()

- Podemos redefinir el método ***toString()*** heredado de la clase Object, añadiendo antes de su definición *@Override*

```
public class Circulo{
    private double radio;
……
    @Overrride
    public String toString(){
        return "Círculo con radio=" + radio;
    }
}
```

- Al pasar un objeto a un println, este invoca automáticamente al método toString().

```
public class parametroObjeto {
   public static void main(String args[ ]){
     Circulo c1 = new Circulo( 5.0 );
     System.out.println(c1);
   }
}
```

Al ejecutar

```
Circulo con radio=5.0
BUILD SUCCESSFUL (total time: 1 second)
```