Comparar objetos de nuestras clases en Java.
Por defecto, al crear un objeto tenemos un método llamado equals que viene de la clase java.lang.Object (es heredada por defecto por todas las clases que creamos).
Cuando comparamos dos objectos, por este método heredado, **determina si la dirección de los objetos es igual**

```
public class testComparador {

	public static void main(String[] args) {
		Deposito d1 = new Deposito(new Persona("535t", "pepe perez"),
365, 0.05, 1000);
		System.out.println(d1);
		Deposito d2 = new Deposito(new Persona("535j", "mar perez"),
365, 0.55, 2000);
		System.out.println(d2);
		Deposito d3 = d1;
		if (d1.equals(d2)) {
			System.out.println("d1 y d2 tienen direcciones
iguales");
		} else {
			System.out.println("d1 y d2 tienen direcciones
distintas");
		}
		if (d1.equals(d3)) {
			System.out.println("d1 y d3 apuntan al mismo sitio");
		} else {
			System.out.println("d1 y d3 tienen direcciones
distintas");
		}
	}

}
```

¿Podríamos definir nosotros este método indicando que se debe cumplir para que 2 depositos sean iguales? La respuesta es sí, podemos sobrescribir este método, como ya vimos en su momento. La definición del método es:
public boolean equals (objeto_clase a)
Supongamos que para que un deposito sea igual que otro tiene que tener mismo capital, mismo plazo, mismo interes entonces

También podemos indicar si un objeto es mayor o menor que otro, por ejemplo, si un deposito tiene mas capital que otro, indicaremos que es mayor.

Para ello usaremos el método compareTo e implementaremos la interfaz Comparable. Veamos como se hace:

```
public class Persona implements Comparable<Persona>
```

Ahora nos obligara a implementar el método compareTo, esta es su definición:

```
public int compareTo (objeto otro)
```

Menor que cero este objeto menor que el otro objeto
0 este objeto y el otro son iguales
Mayor que cero este objeto mayor que el otro objeto

```
public class serVivo implements Comparable<serVivo> {
private int edad;
public serVivo(int edad) {
	this.edad = edad;
}

public int getEdad() {
	return edad;
}

public void setEdad(int edad) {
	this.edad = edad;
}
public int compareTo(serVivo otro) {
	if (this.edad<otro.edad) return -1;
	else if (this.edad>otro.edad) return 1;
	else return 0;
}
@Override
public String toString() {
	return "serVivo [edad=" + edad + "]";
}

}
/***************/
public class serHumano extends serVivo{
private String nombre;

public serHumano(int edad, String nombre) {
	super(edad);
	this.nombre = nombre;
}

public String getNombre() {
```

```
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}
@Override
public String toString() {
	return "serHumano [nombre=" + nombre +  super.toString();
}

}


public class testVivos {

	public static void main(String[] args) {
		serVivo x,y;
		 x=new serVivo(5);
		 y=new serVivo(3);
		 serHumano h;
	  if (x.compareTo(y)>0) {
	   System.out.println("x es mayor que y");
	  }
	 serHumano  z=new serHumano(34,"Homero");
	  serHumano t=new serHumano(34,"Bart");
		if (z.compareTo(t)==0) {
			System.out.println(z.getNombre()+ " es de la misma edad
que "+t.getNombre());
		}

	}

}
```

Dos cosas más sobre objetos

Operador instanceof o cómo saber si una instancia pertenece a una clase:

El operador ***instanceof*** sirve para conocer si un objeto es de un tipo determinado. Por tipo, nos referimos a clase o interfaz (*interface*), es decir, si el objeto pasaría el test «ES UN» para esa clase o ese interfaz, especificado a la derecha del operador.

```
String s = new String("No leas esto, sólo es un ejemplo");
if (s instanceof String)
     System.out.prinln("Efectivamente s pertenece a la clase String");
```

Puede ocurrir que el objeto que estamos comprobando con

instanceof, no sea una instancia directa de la clase que aparece a la derecha del operador, aun así instanceof devolverá true si el objeto es de un tipo compatible con el objeto que aparece a su derecha. Por ejemplo una instancia de una subclase. Esto se ve mejor con un ejemplo:

```
class Animal {}
class Perro extends Animal {
   public static void main (String[] args){
     Perro toby = new Perro();
     if (toby instanceof Animal)
       System.out.println("toby es un perro y también un animal");
   }
}
```

Lo mismo ocurre si en lugar de tener clases y subclases, implementamos interfaces. Por ejemplo, veamos el siguiente caso:

```
interface Nada {}
class A implements Nada {}
class B extends A {}


A a = new A();
B b = new B();
...
```

Las siguientes sentencias serían verdaderas:

```
a instanceof Nada
b instanceof A
b instanceof Nada  //Ya que b implementa el interfaz Nada, de manera indirecta
```

Error de compilación de instanceof

No podemos usar instanceof para tratar de comprobar dos clases de diferentes jerarquías

clonacion de objetos

cuando queramos que los objetos de una clase puedan ser copiados debemos implementar la interfaz Cloneable y el método clone()
Esto permite que en la susodicha clase se pueda realizar la copia de objetos tal cual, permitiendo hacer por tanto duplicados exactos, aunque eso sí, objetos al fin y al cabo distintos

ejemplo

```
public class Persona {
   public int dni, edad;
      public Persona( int d, int e){
          this.dni = d;
          this.edad = e;
    }
```

```
}
```

debemos reescribir Persona e implementar la interface Cloneable y añadir el método clone

```
public class Persona implements Cloneable{

   public int dni, edad;
      public Persona( int d, int e) {     this.dni = d;       this.edad = e;     }


      public Persona clone()   {
      Persona c = new Persona(this.dni,this.edad);
      return c;
   }
}
```

definimos un método público (ha de ser obligatoriamente público puesto que es la sobreescritura de un método público) que devuelve un objeto de la clase Persona y que se invoca sin parámetros.

¿Qué código incluimos en el método? El código necesario para crear un nuevo objeto Persona con los mismos atributos que tenga el objeto sobre el que se invoque el método, para usarlo:

```
public class Programa {


      public static void main(String arg[]) {
         Persona p = new Persona(74999999,35);
         Persona p2 = p.clone();
          // hemos clonado en el objeto p2 los datos de la Persona p , por tanto p2
         // tiene como dni 74999999 y una edad de 35
         // a continuación vamos a cambiar el dni de p
         p.dni=25454345;
         System.out.println("Datos Persona p: DNI:"+p.dni+" Edad:"+p.edad);
         System.out.println("Datos Persona p2 clon: DNI:"+p2.dni+" Edad:"+p2.edad);
       }
}
```