# Clases y tipos genéricos en Java

En **Java**, cuando definimos una nueva clase, **debemos conocer el tipo de dato** con el que trabajaremos. Si queremos realizar una operación específica dentro de esta nueva clase, **sea cual sea el tipo de datos** que va a recibir, podemos hacer uso de los **tipos genéricos**. Este tipo genérico asumirá el tipo de dato que realmente le pasaremos a la clase.

Ejemplo:

```
class ClaseGenerica<T> {
  T obj;

  public ClaseGenerica(T o) {
    obj = o;
  }

  public void classType() {
    System.out.println("El tipo de T es " + obj.getClass().getName());
  }
}

public class MainClass {
  public static void main(String args[]) {
    // Creamos una instancia de ClaseGenerica para Integer.
    ClaseGenerica<Integer> intObj = new ClaseGenerica<Integer>(88);
    intObj.classType();

    // Creamos una instancia de ClaseGenerica para String.
    ClaseGenerica<String> strObj = new ClaseGenerica<String>("Test");
    strObj.classType();

  }
}
```

Notas:

- T es el tipo genérico que será reemplazado por un tipo real.
- T es el nombre que damos al parámetro genérico.
- Este nombre se sustituirá por el tipo real que se le pasará a la clase.

El resultado será el siguiente:

```
1 El tipo de T es java.lang.Integer
2 El tipo de T es java.lang.String
```

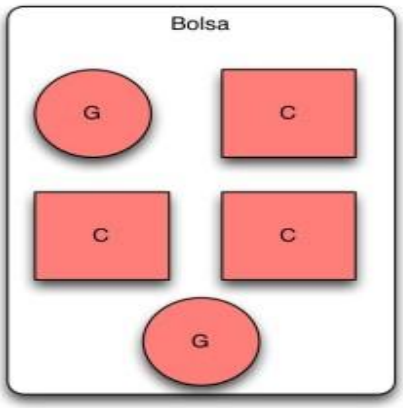
Hay que tener en cuenta que **los generics de java solo funcionan con objetos**. El código siguiente nos mostrará un error:

```
ClaseGenerica<int> myOb = new ClaseGenerica<int>(53); // Error, can't
use primitive type
```

Existen una serie de **convenciones para nombrar a los genéricos**:

- E – Element (usado bastante por Java Collections Framework)
- K – Key (Llave, usado en mapas)
- N – Number (para números)
- T – Type (Representa un tipo, es decir, una clase)
- V – Value (representa el valor, también se usa en mapas)
- S, U, V, etc. – usado para representar otros tipos.

Veamos cómo se usan los Java Generics o llamadas simplemente **clases Genéricas**. Construiremos la clase Bolsa que es una clase sencilla que nos permitirá almacenar objetos de varios tipos.

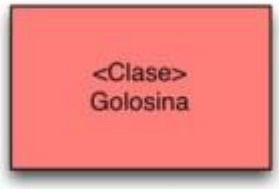


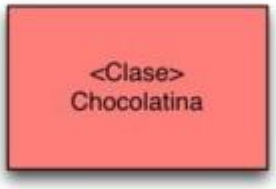

Esta clase tendrá un límite de objetos a almacenar. Alcanzado el límite no se podrán añadir más .Vamos a ver su código fuente:

```
import java.util.ArrayList;
import java.util.Iterator;

public class Bolsa implements Iterable{
	private ArrayList lista;
	private int tope;

	public Bolsa(int tope) {
		lista= new ArrayList();
		this.tope = tope;
	}

	public void add(Object objeto ) {
		if (lista.size()<=tope) {
			lista.add(objeto);
		}else {
			throw new RuntimeException("no caben mas");
		}
	}
```

```
	public Iterator iterator() {
		return lista.iterator();
	}

}
```

En nuestro caso vamos a disponer de dos clases con las cuales rellenar la bolsa. La clase Golosina y la clase Chocolatina.




Vamos a ver su código fuente:

```
public class Golosina {
private String nombre;

public String getNombre() {
	return nombre;
}

public void setNombre(String nombre) {
	this.nombre = nombre;
}

public Golosina(String nombre) {
	super();
	this.nombre = nombre;
}

}


public class Chocolatina {
 private String marca;
 public String getMarca() {
	return marca;
}
 public void setMarca(String marca) {
	this.marca = marca;
}
public Chocolatina(String marca) {
	super();
	this.marca = marca;
```

```
}
}
```

Creamos un programa que llene la Bolsa de Chocolatinas y Golosinas para luego recorrer los elementos que están en la bolsa y sacarlos por pantalla.

```
public class Principal {

 public static void main(String[] args) {
        Bolsa bolsa= new Bolsa(5);
        Chocolatina c= new Chocolatina("milka");
        Chocolatina c1= new Chocolatina("milka");
        Chocolatina c2= new Chocolatina("ferrero");
        Golosina g1= new Golosina("gominola");
        Golosina g2= new Golosina("chicle");

        bolsa.add(c);
        bolsa.add(c1);
        bolsa.add(c2);
        bolsa.add(g1);
        bolsa.add(g2);

        for (Object o: bolsa) {

        if(o instanceof Chocolatina) {
                Chocolatina chocolatina= (Chocolatina)o;
                System.out.println(chocolatina.getMarca());
        }else {
                Golosina golosina= (Golosina)o;
                System.out.println(golosina.getNombre());
        }
        }
 }
}
```

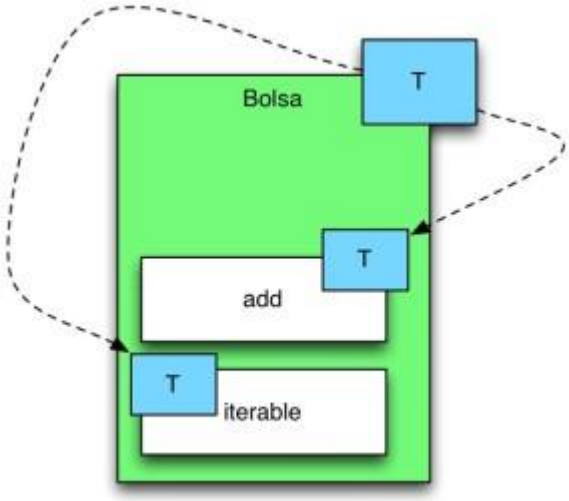
El programa funcionará correctamente, pero nos podremos dar cuenta que **resulta bastante poco amigable la estructura if /else en la cual se chequean cada uno de los tipos a la hora de presentarlo por pantalla.**

# Java Generics

Para solventar este problema podemos construir una clase Genérica. Este tipo de clase nos permitirá definir una Bolsa de un tipo concreto. Puede ser una bolsa de Golosinas o una bolsa de Chocolatinas **pero NO de las dos cosas a la vez**. Esto en un principio puede parecer poco flexible pero si nos ponemos a pensar, cuando programamos solemos imprimir una lista de Facturas o una lista de Compras no una lista mixta. Así pues el enfoque parece razonable. Vamos a ver el código fuente y comentarlo:

```
import java.util.ArrayList;
import java.util.Iterator;

public class Bolsa<T> implements Iterable<T>{
	private ArrayList<T> lista= new ArrayList<T>();
	private int tope;

	public Bolsa(int tope) {
		this.tope = tope;
	}

	public void add(T objeto ) {
		if (lista.size()<=tope) {
			lista.add(objeto);
		}else {
			throw new RuntimeException("no caben mas");
		}
	 }

	public Iterator<T> iterator() {
		return lista.iterator();
	}

}
```

La clase es un poco peculiar ya que al no saber de entrada de qué tipo va a ser la bolsa **debemos declarar un tipo Genérico T a nivel de clase y que será repetido en cada uno de los métodos que lo usen.**




De esta manera, cuando construyamos un objeto de esta clase será el momento de especificar el tipo de Bolsa que deseamos. En el siguiente ejemplo hemos elegido "Chocolatina" como tipo para la Bolsa. De esta manera la bolsa solo admitirá este tipo de objetos.

```
public class Principal {
public static void main(String[] args) {
        Bolsa<Chocolatina> bolsa= new Bolsa<Chocolatina>();

        Chocolatina c= new Chocolatina("milka");
        Chocolatina c1= new Chocolatina("milka");
        Chocolatina c2= new Chocolatina("ferrero");

        bolsa.add(c);
        bolsa.add(c1);
        bolsa.add(c2);

        for (Chocolatina chocolatina : bolsa) {
                System.out.println(chocolatina.getMarca());
        }
}
}
```

http://www.arquitecturajava.com/uso-de-java-generics/