El signo ? en los genéricos

- El signo **"?"**, cuando se utiliza en un genérico, indica que es u*n tipo desconocido*. Se conocen en inglés como **wildcards**.
- En muchas ocasiones utilizar una variable de tipo **T** o un wildcard **"?"** Es lo mismo. Por ejemplo es igual usar **List<T>** que usar **List<?>**, ya que en ambos casos significa que la lista se puede instanciar con cualquier tipo (que extienda a **Object**).

- Sin embargo existen diferencias de uso que son importantes:
  - 1.- El uso de **"?"** Es menos restrictivo que usar un parámetro de tipo. Por ejemplo escribir **<?,?>** no es lo mismo que **<T,T>**. El primero no exige que ambos elementos sean del mismo tipo. El segundo si.
  - 2.- **"?"** admite ser usado con cotas superiores e inferiores. Los parámetros de tipo sólo con cotas superiores.

El signo ? en los genéricos

```
public static <T extends Number> void copy1(GenericList<T> l1, GenericList<T> l2){
 ...
}
```

Se obliga a que ambos parámetros sean exactamente del mismo tipo.

No se puede hacer:

```
GenericList<Integer> l1 = new GenericList<Integer>();
GenericList<Double> l2 = new GenericList<Double>();
copy1(l1,l2)
```

```
public static void copy2(GenericList<? extends Number> l1,
                         GenericList<? extends Number> l2){
 ...
}
```

Ambos parámetros deben extender a **Number**, pero pueden ser distintos.
Por lo tanto es correcto

```
copy2(l1,l2)
```

El signo ? en los genéricos

```
public static <T super Integer> void print(GenericList<T> l){
 …
}
```

ESTÁ PROHIBIDO – No compila –

```
public static void print(GenericList<? super Integer> l){
 …
}
```

CORRECTO !!!

Restricciones sobre genéricos

- No se pueden instanciar con tipos primitivos:

`Pair<int,char> p = new Pair<int,char>(9,'a');` no compila.

- Hay que hacerlo utilizando **boxing/unboxing:**

```
Pair<Integer,Character> p = new Pair<Integer,Character>(9,'a');
```

- No se pueden hacer instancias de los parámetros de tipo. Por ejemplo, es incorrecto hacer

  `E elem = new E()`. **Genera error de compilación.**

- No se pueden declarar campos estáticos de tipo genérico.
- Para más restricciones consultar:

`https://docs.oracle.com/javase/tutorial/java/generics/restrictions.html`

Ejemplo 1: Aritmética

```
public abstract class BinaryArithmetics<T> {
  protected T x, y;

  public BinaryArithmetics(T x, T y){
    this.x = x;
    this.y = y;
  }

  public abstract T add();
  public abstract T sub();
  public abstract T div();
  public abstract T mult();
}
```

```
public class DoubleArithmetics
          extends BinaryArithmetics<Double> {

  public DoubleArithmetics(Double x, Double y) {
    super(x, y);
  }
  @Override
  public Double add() { return this.x + this.y; }
  @Override
  public Double sub() { return this.x – this.y; }
  @Override
  public Double div() {return  this.x / this.y }
  @Override
  public Double mult() { return this.x * this.y; }
}
```

33

Ejemplo 1: Aritmética

```
public abstract class BinaryArithmetics<T> {
  protected T x, y;

  public BinaryArithmetics(T x, T y){
    this.x = x;
    this.y = y;
  }

  public abstract T add();
  public abstract T sub();
  public abstract T div();
  public abstract T mult();
}
```

```
public class Rational {
        private int num,den;

        public Rational(int n, int d) {
         this.num = n;
         this.den = d;
        }

        public int getNum() {
          return this.num;
        }

        public int getDen() {
          return this.den;
        }

        public String toString() {
         return this.num + "/" + this.den;
        }
}
```

34

Ejemplo 1: Aritmética

```
public class RationalArithmetics extends BinaryArithmetics<Rational> {

        public RationalArithmetics(Rational r1, Rational r2) { super(r1,r2); }
        @Override
        public Rational add() {
         int n = (this.x.getNum() * this.y.getDen()) + (this.x.getDen()*this.y.getNum());
         int d = (this.x.getDen() * this.y.getDen());
         return new Rational(n,d);
        }
        @Override
        public Rational sub() {
         int n = (this.x.getNum() * this.y.getDen()) - (this.x.getDen()*this.y.getNum());
         int d = (this.x.getDen() * this.y.getDen());
         return new Rational(n,d);
        }
        @Override
        public Rational div() {
          int n = this.x.getNum() * this.y.getDen();
          int d = this.x.getDen() * this.y.getNum();
          return new Rational(n,d);
        }
        @Override
        public Rational mult() { ... }
}
```

35

Ejemplo 1: Aritmética

```
public static void main(String[] args) {

        Rational r1 = new Rational(2,3);
        Rational r2 = new Rational(7,8);
        RationalArithmetics r = new   RationalArithmetics(r1,r2);
        System.out.println(r.add());


        DoubleArithmetics d1 = new DoubleArithmetics(2.3,1.2);
        System.out.println(d1.div());
}
```

36

Ejemplo 2: Ordenación

```
abstract public class Comparable<T> {

        public abstract boolean less(T e);
        public abstract boolean greater(T e);
        public boolean equal(T e){
                return (!this.less(e) && !this.greater(e));
        }
        public boolean lessOrEquals(T e){
                return (this.less(e) || this.equals(e));
        }
        public boolean graterOrEqualsThan(T e){
                return (this.greater(e) || this.equals(e)); }
}
```


Ejemplo 2: Ordenación

```
public class Person extends Comparable<Person> {

        private String name;
        private int id;

        Person(String name, int id) {
         this.name = name;
         this.id = id;
        }
        public String toString() { … }

        @Override
        public boolean less(Person e) {
          return this.id < e.id;
        }
        @Override
        public boolean greater(Person e) {
          return this.id > e.id;
        }
}
```

Ejemplo 2: Ordenación

```
public class Account extends Comparable<Account>{
        private int numCuenta;
        private int saldo;

        public Account(int nc, int s) {
           this.numCuenta = nc;
           this.saldo = s;
        }
        @Override
        public boolean less(Account e) {
          return this.numCuenta < e.numCuenta;
        }
        @Override
        public boolean greater(Account e) {
          return this.numCuenta > e.numCuenta;
        }

        public String toString() {
         return "Numero Cuenta: " +
           this.numCuenta + " Saldo: " + this.saldo;
   }
  }
```

39

Ejemplo 2: Ordenación

```
public class SortExample {

 public static void sort(int[] a) {
    for (int i = 1; i < a.length; i++) {
        int x = a[i];
        int j = i - 1;
        while (j >= 0 && a[j] > x) {
          a[j + 1] = a[j];
          j--;
        }
        a[j + 1] = x;
    }
 }
```

```
public static <T extends Comparable<T>> void sort(T[] a) {
        for (int i = 1; i < a.length; i++) {
         T x = a[i];
         int j = i - 1;
         while (j >= 0 && a[j].greater(x)) {
           a[j + 1] = a[j];
           j--;
         }
        a[j + 1] = x;
        }
       }
}
```

40

Ejemplo 2: Ordenación

```
public class SortExample {
 public static void sort(int[] a) { // ordena arrays de enteros
  for (int i = 1; i < a.length; i++) {
   int x = a[i];
   int j = i - 1;
   while (j >= 0 && a[j] > x) {
    a[j + 1] = a[j];
    j--;
   }
   a[j + 1] = x;
  }
 }
 public static <T extends Comparable<T>> void sort(T[] a) {
  for (int i = 1; i < a.length; i++) {
   T x = a[i];
   int j = i - 1;
   while (j >= 0 && a[j].greater(x)) {
    a[j + 1] = a[j];
    j--;
   }
   a[j + 1] = x;
  }
 }
}
```


Ejemplo 2: Ordenación

```
public class SortExample {
 public static void sort(int[] a) { // ordena arrays de enteros
  for (int i = 1; i < a.length; i++) {
   int x = a[i];
   int j = i - 1;
   while (j >= 0 && a[j] > x) {
    a[j + 1] = a[j];
    j--;
   }
   a[j + 1] = x;
  }
 }
 public static <T extends Comparable<T>> void sort(T[] a) {
  for (int i = 1; i < a.length; i++) {
   T x = a[i];
   int j = i - 1;
   while (j >= 0 && a[j].greater(x)) {
    a[j + 1] = a[j];
    j--;
   }
   a[j + 1] = x;
  }
 }
}
```

// ordena arrays que extiendan a Comparable
// para garantizar que existe orden

Ejemplo 2: Ordenación

```
public static void test1() {
  Account[] a = new Account[] {
              new Account(4,3000),
              new Account(2,5000),
              new Account(1,1000) };
  sort(a);
  System.out.println( Arrays.toString(a));
}

public static void test2() {
  Person[] a = new Person[] { new Person("John",4),
                              new Person("Mike",1),
                              new Person("Andres",10) };
  sort(a);
  System.out.println( Arrays.toString(a));
}

public static void test3() {
     int[] a = new int[] {5,2,8,5,3,10};
     sort(a);
     System.out.println( Arrays.toString(a));
}
```