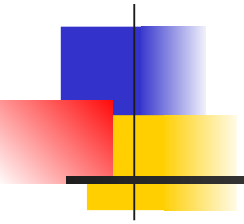


Generics and Type Parameters

References for the Details

Generics.doc - my write-up on generics and type param

Core Java for the Impatient - has a chapter on generics, with many examples

Big Java, Chapter 18 (Generics)

Simple Generic Class

Example you have already written:

```
public class ArrayIterator<T>
             implements Iterator<T> {
    private T[] array;

    public ArrayIterator(T[] a) {
        this.array = a;
    }
```

Type Param allowed only on instance members

- Java processes type parameters using "type erasure".
- As a result, a class type param cannot be used on static members.

```
public class MyClass<T> {
    private T attribute;        // OK
    private static T  x;        // ERROR

    public T getAttribute() { // OK
         return attribute;
    }
    public static T newInstance() // ERROR
```

Type Parameter with Bound

- You can limit type parameters using "`extends`" and "`super`".
- T is any type that implements Runnable:

```
public class TaskManager<T extends Runnable>
{
    private List<T> tasks;

    public void runAll() {
        for(Task t: tasks) t.run();
    }
    public void addTask(T task) {
        tasks.add(task);
    }
```

Wildcard Character ?

- Means "anything".
- Can use "?" on non-generic methods.
- Can be used with bounds, like "? extends Valuable".

```
public void printAll(List<?> list) {

    // forEach is same as "for-each" loop
    // requires Java 8
    list.forEach( (x) ->
        System.out.println(x) );
}
```

Demo: CoinUtil.filterByCurrency

- The signature of the method is:

```
public static void sortByCurrency(
                        List<Valuable> list) {
    // sort items by currency
}
```

- But this code won't compile. Why? How to fix?

```
List<Coin> coins = Arrays.asList(
   new Coin(5,"Cents"),new Coin(1,"Baht"));

sortByCurrency( coins );    // ERROR
```

Generic Methods

- A static method can have a type parameter.
- It can be method in an ordinary (non-generic class).
- java.util.Arrays and java.util.Collections are examples

```
public static <E> reverse(E[ ] array) {

    int size = array.length - 1;
    for(int k=0; k<size/2; k++) {
        E temp = a[k];
        a[k] = a[size-k];
        a[size-k] = temp;
    }
}
```

Calling Generic Methods

- The caller does not mention the type parameter.
- Java compiler *infers* the type from parameters.

```
Number[] array = new Number[]{1, 2, 3};

reverse( array );  // result: [3, 2, 1]

String[] words = {"a", "b", "c"};

reverse( words );  // result: ["c","b","a"]
```

Example

- Write a generic "max" method.
- Find "max" of two objects that implement Comparable.

```
public static <E extends Comparable<E>>
      E max(E a, E b) {

    //TODO return the greater of a and b
}
// This method has a problem:
String m = max("Cat", "Dog");    // OK
Coin m2 = max(new Coin(5), new Coin(10));
      // Compile error
      // Coin implements Comparable<Valuable>
```

"? super E"

- max should accept 2 objects that implement Comparable<any superclass of E>
- Coin implements Comparable<Valuable> implies Coin is also comparable to Coin

```
static <E extends Comparable<? super E>>
      max(E a, E b) {

    //TODO return the greater of a and b
}

Coin m2 = max(new Coin(5), new Coin(10));
      // OK!
```

Demo: Java API

- Look at `Collections.fill` - replace all elements in list with copies of an object
- `Collections.sort( )` - 2 methods

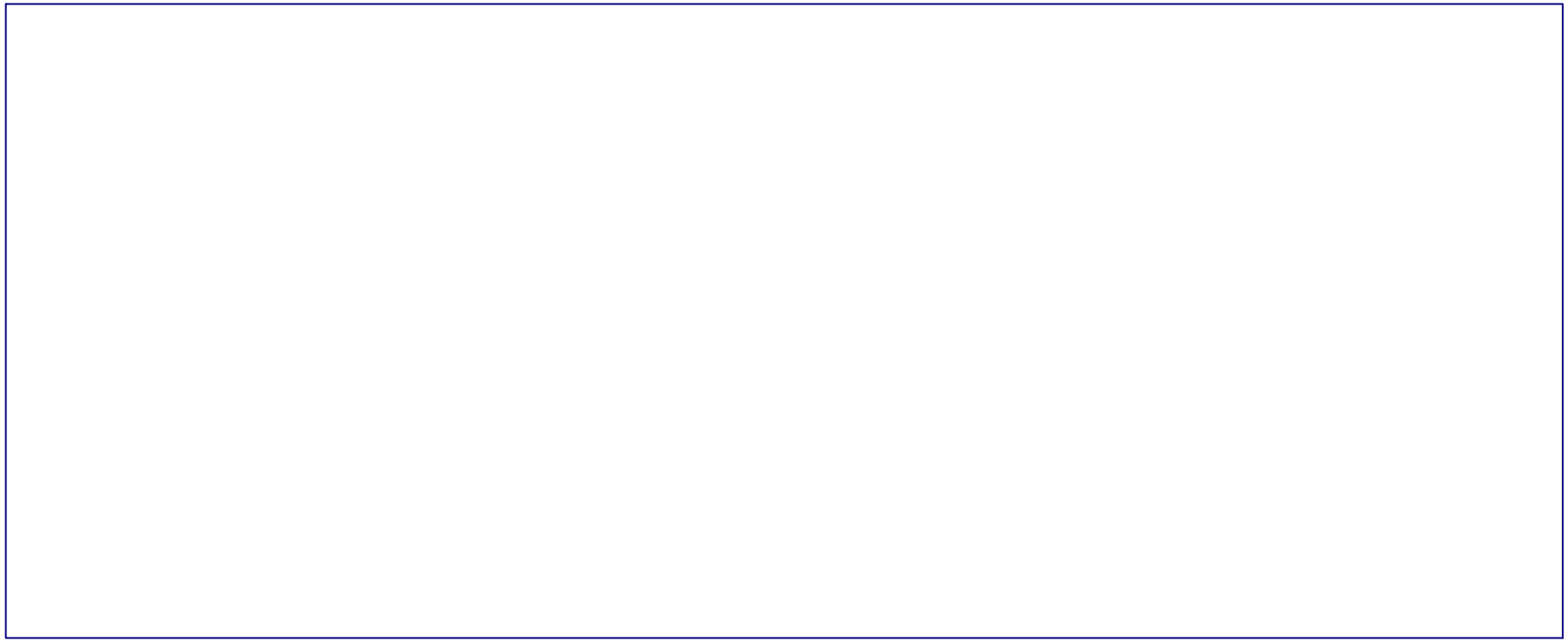
```
public static <T> void
            fill(List<? super T>, T obj)

public static <T> sort(List<T> list,
      Comparator<? super T> comp)

// See Also: sort(List<T> list)
```

Demo: CoinUtil.filterByCurrency

- Always returns `List<Valuable>`



Summary

Class wildcards apply to instance members only.

Generic methods have their own type param.

Bounds and wildcards:

? = match anything (can be used on *any* method)

<? super E> = match superclass of E

<E extends Foo> = bound on type param

Most important is to *understand generics in API docs*.

If you need to write generic method, check a book.