Type Arguments

- supplied in declarations and "`new`" expressions
- are comma separated and enclosed in <>
- are actual types.
- can be "<>" when the compiler can deduce the missing bits.

Type Argument Examples

- `List<String> x = new ArrayList<String>();`
- `ArrayList<Double> xx = new ArrayList<>();`

Restrictions

- `ArrayList<int>`. Type arguments cannot be primitives.
- `new E();` — Cannot create an object of generic type.
- `new E[5];` — Cannot create an array object whose elements are generic.
- `static List<E> counters` Static fields cannot reference generic class formal parameters.

Formal Type Parameter Examples

- `<E>`
- `<K,V>`
- `<T extends Comparable<? super T>>`

Formal Type Parameters

- are comma separated and enclosed in <>
- are either names or ?
- can have "`extends` *xxx*" or "`super` *xxx*" clauses.

Generic class declaration

```
public class Stack<E>
{   private List<E> data ;
    public Stack()
    {   data = new LinkedList<E>() ;
    }

}
```

Generic method declarations

Formal type parameter come just before return type or constructor name.

```
{   private static <E> min(E[] data)
    { ... }

}
```