CSE 331: Generics

You have *used* generics extensively already in Java: for example, in "`List<Integer>`".
Here, `List` is a generic interface that "takes a type as an argument" (`Integer`).

You can define your own generic interfaces and classes:

```
class MySet<T> {
  private List<T> elements;
  private T lastAccessed;

  public boolean contains(T element) {
    for (T x : elements)
      if (x.equals(element)) return true;
    return false;
  }
}
```

The scope of `T` is the entire class:
types of fields, method parameters, local variables, etc.

A class can also have more than one type parameter:
for example `class MyMap<K, V> { ... }`

Generic type parameters can also have *bounds*:

```
class MySet<T extends Number> {
  void foo(T element) {
     arg.intValue();
  }
}

interface Comparable<T> {
  int compareTo(T other);
}
class MySet<T extends Comparable<T>> {
  private List<T> elements;  // RI: sorted
  void binarySearch(T element) {
    ... elements.get(i).compareTo(element) ...
  }
}
```

`arg.intValue()` is allowed because: `arg` has type `T`, and
`T` is known to be a subtype of `Number`, and
`Number` has an `intValue` method.

Remove `extends Number`, and `arg.intValue()` is disallowed,
because then `T` is only known to be a subtype of `Object`,
and `Object` does not have an `intValue` method.

`T` can be used in its own bound!

You can also have generic *methods*:

```
<T> copy(List<T> src, List<T> dest) {
  for (T x : src)
    dst.add(x);
}
```

The scope for a generic method's type parameter is:
the type parameter bounds, the method parameter's types,
and the body of the method

`Integer` is a Java subtype of `Number`. Is `List<Integer>` a Java subtype of `List<Number>`? _____

A *wildcard* is an anonymous type variable, written "`?`".

```
Object o;                   l = new ArrayList<Object>();             l.add(o);      o = l.get(0);
Number n;                   l = new ArrayList<Number>();             l.add(n);      n = l.get(0);
Integer i;                  l = new ArrayList<Integer>();            l.add(i);      i = l.get(0);
PositiveInteger p;          l = new ArrayList<PositiveInteger>();    l.add(p);      p = l.get(0);
                            l = new ArrayList<NegativeInteger>();    l.add(null);
List<? extends Integer> l;
List<? super Integer> l;
```

`Integer` is a Java subtype of `Number`. Is `Integer[]` a Java subtype of `Number[]`? _____