Lab 10

1. Consider the following code fragments. For each, if there is a compiler error, identify where it occurs.

   a. First fragment:

```
List<Integer> ints = new ArrayList<Integer>();
ints.add(1);
ints.add(2);
List<Number> nums = ints;
nums.add(3.14);
```

   b. Second fragment:

```
List<Integer> ints = new ArrayList<Integer>();
ints.add(1);
ints.add(2);
List<? extends Number> nums = ints;
nums.add(3.14);
```

2. Draw a class diagram showing the inheritance relationships among the following types:

   `List<Integer>, List<Number>, List<? extends Integer>, List<? extends Number>, List<? super Integer>, List<? super Number>, List<?>, List<Object>`

3. Recall the definition of `sum` given in the slides:

```
public static double sum(Collection<? extends Number> nums) {
  double s = 0.0;
  for (Number num : nums) s += num.doubleValue();
  return s;
}
```

   a. Is there a compiler error in the following lines of code? If so, where?

```
List<Integer> ints = new ArrayList<Integer>();
ints.add(1);
ints.add(2);
List<? extends Number> nums = ints;
double dbl = sum(nums);
nums.add(3.14);
```

b. Is there a compiler error in the following lines of code? If so, where?

```
List<Object> objs = new ArrayList<Object>();
objs.add(1);
objs.add("two");
List<? super Integer> ints = objs;
ints.add(3);
double dbl = sum(ints);
```

4. Create a generic programming solution to the problem of finding the second smallest element in a list. In other words, devise a public static method secondSmallest so that it can handle the biggest possible range of types.