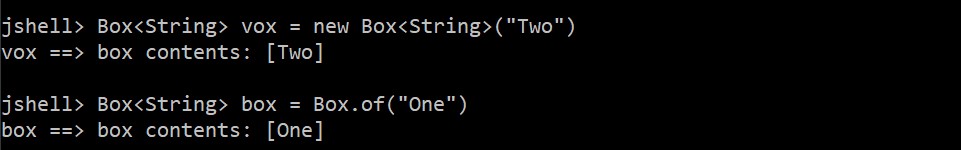
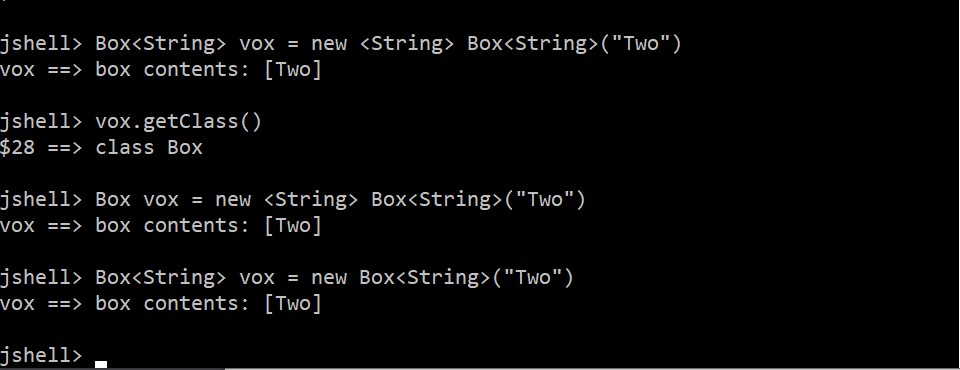
**CS2030S**

**RECITATION 4**

**Q1**

1. Ok
2. Ok
3. No

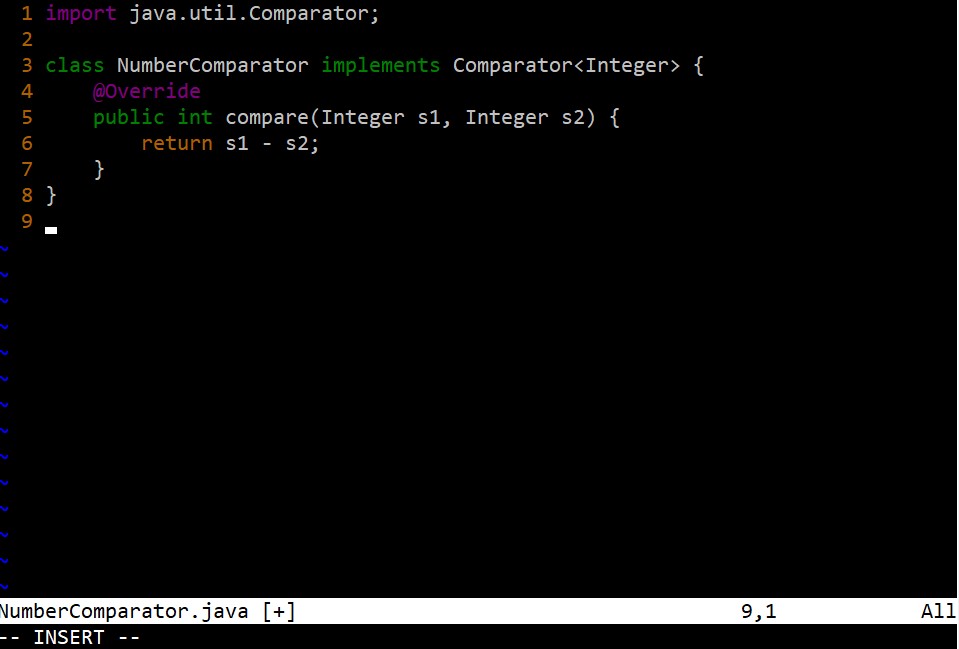




**Q2**

1. Not valid, since the method is invariant (ie only accepts List object, even if ArrayList is a subclass of List)
2. Valid, since the method is a contravariant method.
3. Valid, since the method is a covariant method
4. Not valid. Int is a primitive and does not extend from any object
5. Invalid, since ArrayList is not a superclass of Integer

**Q3**



1. -1, 2, and -3 are all primitive int so the generic <T> is <int> . originally, the primitive -1 is assigned to the variable int max. at the if-clause, the NumberComparator’s compareTo method is invoked with 2 and max passed in as arguments, where if the return value is >0, it means that 2 is greater than max’s value (which it is), and thus the variable max is now assigned the value 2. The next if-clause evaluates false as the compareTo’s arguments of c, which is -3, and max returns a negative value, meaning the argument c is of lower value than the value held by the variable max. so ultimately the max3 method returns 2
2. No it does not work because the generic parameter is of primitive int, which does not have the compareTo method that is present in the Integer class. Error is that compareTo method is not defined
3. With the parameterized wrapper <T extends Comparable<T>>, the values placed as arguments have to be a subclass of Comparable. Java then autowraps the primitive int values into class Integer. Since compareTo method is present in the Integer class, it can be evoked to compare the values against each other. Steps are the same as part (a).
4. i) <T> Comparable<T> max3(Comparable<T> a, Comparable<T> b, Comparable<T> c) cannot. Wrong syntax?

ii) <T> T max3 (Comparable<T> a, Comparable<T> b, Comparable<T> c). wrong syntax.

iii) Comparable max3(Comparable a, Comparable b, Comparable c) is ok.