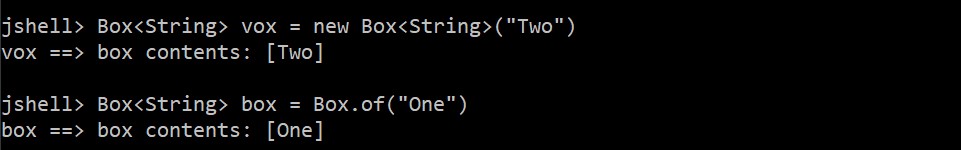
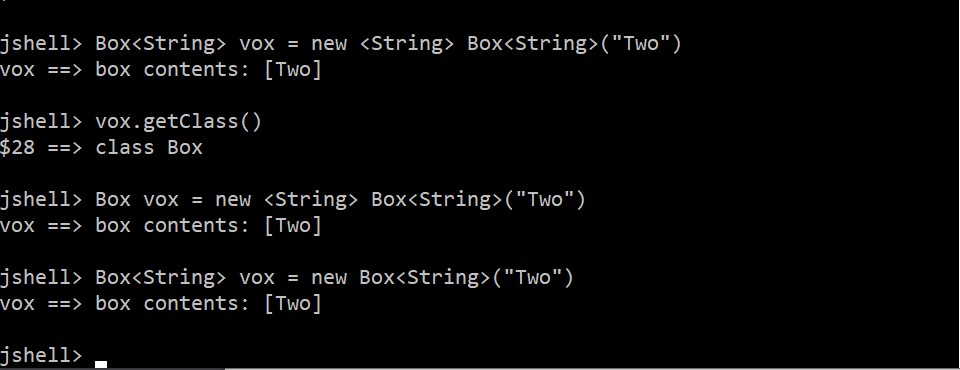
**CS2030S**

**RECITATION 4**

**Q1**

1. Ok
2. Ok
3. No. it actually works, but you will get a warning (since it is a backward compatibility thing. this is a raw-type.





Primitives -> int, boolean, double

Reference -> Double, Integers, etc

Python is latent/implicit typing, java is explicit typing.

<T> is the type parameter (placeholder). Eg normally it is int n; but now it is new A<Integer>. T must be a reference type, not primitive

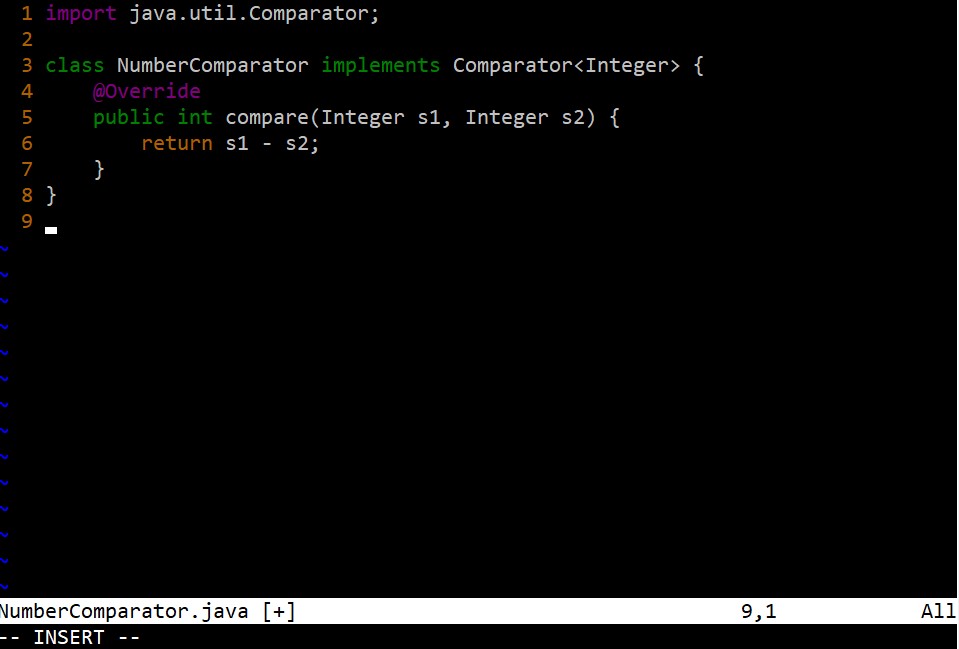
If there is nothing, ie <> is <Object> since Object is the root.

Can also do <T, U, S> etc

**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. Java does the auto boxing of the int to Integer at the NumberComparator method when x-y. it unboxes the Integer to become int, then once the operation is done, it boxes it up again to become Integer class.
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. Wrong. Because at this point Java does not know what T is yet, so it cannot guarantee T has a compareTo method, so the compiler does throws an error (ie cannot fine symbol symbol: method compareTo())
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). wrong explanation. The difference is that in 3c, since T extends (in this place, it is not implements even though it is implements. It is extends. Java things.) Comparable, it can guarantee that there is a compareTo method since Comparable interface has a compareTo method. So Java does not break.
4. i) <T> declare this is a type Comparable<T> return value is datatype 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.

<T> at the start is to declare T is a type parameter. Second T is the return type of the method, T inside for arguments have to follow the <T> at the start

Max3 -> principle of least commitment. Ie comp does the comparison, max3 just needs to take in the function to do the comparison

Java does not allow to cast in function as arguments

Max3 needs to take in different comparators because some types do not have the same comparison methods (eg Integers vs Strings)