CS2030 Programming Methodology

Semester 2 2022/2023

1 & 2 March 2023

Problem Set #5

More Java Generics

1. In the lecture, we have seen the use of the `Comparator<T>` interface with the method specification `int compare(T t1, T t2)` that returns zero if `t1` and `t2` are equal, a negative integer if `t1` is less than `t2`, or a positive integer if `t2` is less than `t1`.

```
public interface Comparator<T> { // <T> declared with class scope
    int compare(T o1, T o2);
    ...
}
```

A generic method `T max3(T a, T b, T c, Comparator<T> comp)` can be defined in JShell as shown below. The method takes in three values of type `T` as well as a `Comparator<T>`, and returns the maximum among the values.

```
jshell> <T> T max3(T a, T b, T c, Comparator<T> comp) { // <T> declared with
   ...>     T max = a;                                  // method scope
   ...>     if (comp.compare(b, max) > 0) {
   ...>         max = b;
   ...>     }
   ...>     if (comp.compare(c, max) > 0) {
   ...>         max = c;
   ...>     }
   ...>     return max;
   ...> }
|  created method max3(T,T,T,Comparator<T>)
```

(a) Demonstrate how the `max3` method can be called so as to return the maximum of three integers −1, 2 and −3. max3(1,2,3,(x, y) -> x - y)

(b) Other than `Comparator<T>`, there is a similar `Comparable<T>` interface with the method specification `int compareTo(T o)`. This allows one `Comparable` object to compare itself against another `Comparable` object.

```
public interface Comparable<T> {
    int compareTo(T o);
}
```

As an example, since `Integer` class implements `Comparable<Integer>`,

```
jshell> Integer i = 1 // 1 autoboxed to an Integer and assigned to i
i ==> 1

jshell> i.compareTo(2) // 2 autoboxed to an Integer and passed to compareTo
$.. ==> -1
```

Let's redefine the `max3` method to make use of the `Comparable` interface instead.

```
<T> T max3(T a, T b, T c) {
    T max = a;
    if (b.compareTo(max) > 0) {
        max = b;
    }
    if (c.compareTo(max) > 0) {
        max = c;
    }
    return max;
}
```

T must implement Comparable<T> interface for it to be able to call the compareTo() method
Since the compiler does not know at the time of compilation if T implements the comparable interface, it would lead to a compilation error

Does the above method work? What is the compilation error?

(c) Does the following declaration of `max3` work?

```
<T> T max3 (Comparable<T> a, Comparable<T> b, Comparable<T> c)
```

no

(d) To restrict `T` to have the `compareTo` method, i.e. any class that binds to `T` must implement `Comparable`, we redefine the type parameter `<T>` to be `<T extends Comparable<T>>`.

```
<T extends Comparable<T>> T max3(T a, T b, T c) {
    T max = a;
    if (b.compareTo(max) > 0) {
        max = b;
    }
    if (c.compareTo(max) > 0) {
        max = c;
    }
    return max;
}
```

As each element T extends the Comparable<T> interface, any element that binds to T will have to compareTo(T) method. In max3(a,b,c), each element a, b & c will be able to use the compareTo() method

Demonstrate how the method `max3` can be used to find the maximum of three values -1, 2 and -3. Explain how it works now.

2. Suppose a `Fruit` class implements the `Comparable` interface, and `Orange` is a sub-class of `Fruit`.

```
class Fruit implements Comparable<Fruit> {
    @Override
    public int compareTo(Fruit f) { ... }
}

class Orange extends Fruit { }
```

We would like to redefine the `max3` method such that the parameter type of `max3` is `List<T>` instead (more specifically a list of three elements). Does the following declaration of the method work?

yes, it works but it takes in a list of any size and not specifically 3 elements

```
<T extends Comparable<T>> T max3(List<T> list)
```

subclass?

Try it out by finding the maximum of a list of three fruits or a list of three oranges. How do you declare the method so that it works for both types of list? You should aim to make the method as flexible as you can.

```
<T extends Comparable<T>> T max3(List<? extends T> list)
```

By using the upper-bounded wildcard, any type parameter that is a subtype of T will be able to be taken in the list.

Ans:

```
<T extends Comparable<? super T>> T max3(List<T> list)
```

```
<T extends Comparable<? super T>> T max3(List<? extends T> list)
```