CS2030 Programming Methodology
Semester 1 2019/2020

13 September 2019
Problem Set #3
Generics and Variance of Types

1. For each of the statements below, indicate if it is a valid statement with no compilation error. Explain why.

   (a) `List<?> list = new ArrayList<String>();`

   (b) `List<? super Integer> list = new List<Object>();`

   (c) `List<? extends Object> list = new LinkedList<Object>();`

   (d) `List<? super Integer> list = new LinkedList<int>();`

   (e) `List<? super Integer> lsit = new LinkedList();`

2. Given the following Java program fragment,

```
class Main {
    public static void main(String[] args) {
        double sum = 0.0;

        for (int i = 0; i < Integer.MAX_VALUE; i++) {
            sum += i;
        }
    }
}
```

   you can determine how long it takes to run the program using the `time` utility

```
$time java Main
```

   Now, replace `double` with the wrapper class `Double` instead. Determine how long it takes to run the program now. What inferences can you make?

3. Recall that the `==` operator compares only references, i.e. whether the two references are pointing to the same object. On the other hand, the `equals` method is more flexible in that it can override the method specified in the `Object` class.

   In particular, for the `Integer` class, the `equals` method has been overridden to compare if the corresponding `int` values are the same or otherwise.

   What do you think is the outcome of the following program fragment?

```
Integer x = 1;
Integer y = 1;
x == y

x = 1000;
y = 1000;
x == y
```

Why do you think this happens? *Hint: check out Integer caching*

4. Compile and run the following program fragments and explain your observations.

(a)
```
import java.util.List;

class A {
    void foo(List<Integer> integerList) {}
    void foo(List<String> StringList) {}
}
```

(b)
```
class B<T> {
    T x;
    static T y;
}
```

(c)
```
class C<T> {
    static int b = 0;
    T y;

    C() {
        this.b++;
    }

    public static void main(String[] args) {
        C<Integer> x = new C<>();
        C<String> y = new C<>();

        System.out.println(x.b);
        System.out.println(y.b);
    }
}
```

5. In the lecture, we have seen the generic method `max3` that takes in an array of generic type `T` that such that `T` implements the `Comparable` interface.

```
public static <T extends Comparable<T>> T max3(T[] arr) {
    T max = arr[0];
    if (arr[1].compareTo(max) > 0) {
        max = arr[1];
    }
    if (arr[2].compareTo(max) > 0) {
        max = arr[2];
    }
    return max;
}
```

Suppose we replace the method header with each of the following:

can t be fast food?

(a) `public static <T> Comparable<T> max3(Comparable<T>[] arr)`

(b) `public static <T> T max3 (Comparable<T>[] arr)`

(c) `public static Comparable max3(Comparable[] arr)`

What if the parameter type of `max3` is `List<T>` instead? How would you change the method header to be as flexible as you can?

6. Which of the following code fragments will compile? If so, what is printed?

(a)
```
List<Integer> list = new ArrayList<>();
int one = 1;
Integer two = 2;

list.add(one);
list.add(two);
list.add(3);

for (Integer num : list) {
    System.out.println(num);
}
```

(b)
```
List<Integer> list = new ArrayList<>();
int one = 1;
Integer two = 2;

list.add(one);
list.add(two);
list.add(3);

for (int num : list) {
    System.out.println(num);
}
```

(c)
```
List<Integer> list = Arrays.asList(1, 2, 3);

for (Double num : list) {
      System.out.println(num);
}
```

(d)
```
List<Integer> list = Arrays.asList(1, 2, 3);

for (double num : list) {
    System.out.println(num);
}
```

(e)
```
List<Integer> list = new LinkedList<>();
list.add(5);
list.add(4);
list.add(3);
list.add(2);
list.add(1);

Iterator<Integer> it = list.iterator();
while (it.hasNext()) {
    System.out.println(it.next());
}
```