CS2030 Programming Methodology
Semester 2 2022/2023

15 & 16 February 2023
Problem Set #4 Suggested Guidance
Java Generics

1. Consider the following JShell program fragment.

```
jshell> ImList<Integer> list = new ImList<Integer>()
list ==> []

jshell> int one = 1
one ==> 1

jshell> Integer two = 2
two ==> 2

jshell> list = list.add(one).add(two).add(3)
list ==> [1, 2, 3]
```

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

   (a) `for (Integer num : list) { System.out.print(num + " "); }`

   (b) `for (int num : list) { System.out.print(num + " "); }`

   (c) `for (Double num : list) { System.out.print(num + " "); }`

   (d) `for (double num : list) { System.out.print(num + " "); }`

```
jshell> for (Integer num : list) { System.out.print(num + " "); }
1 2 3
jshell> for (int num : list) { System.out.print(num + " "); }
1 2 3
jshell> for (Double num : list) { System.out.print(num + " "); }
|  Error:
|  incompatible types: java.lang.Integer cannot be converted to java.lang.Double
|  for (Double num : list) { System.out.print(num + " "); }
|                    ^--^

jshell> for (double num : list) { System.out.print(num + " "); }
1.0 2.0 3.0
```

2. For each of the code fragments below, indicate and explain the source of the error(s).

   (a) `List<? extends Object> list = new ArrayList<Object>()`

   > `list` *assignment is valid since (read* `<:` *as "is substitutable for")...*
   > `ArrayList<Object> <: ArrayList<? extends Object> <: List<? extends Object>`

`list.add(new Object())`

> `list.add(new Object())` *is invalid since* `list` *could refer to* `ArrayList<Integer>`

(b) `List<? extends Object> list = List.of("abc");`

> `list` *assignment is valid since*
> `List<String> <: List<? extends String> <: List<? extends Object>`

```
list.add("def");
String s = list.get(0);
```

> *Both* `list.add("abc")` *and* `String s = list.get(0)` *aree invalid since* `list` *could refer to* `ArrayList<Integer>`. *However,* `Object o = list.get(0)` *is fine.*

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

> `list` *assignment is invalid since* `List` *is an interface. It wil be fine if we change* `List<Object>` *to* `ArrayList<Object>` *since*
> `ArrayList<Object> <: List<Object> <: List<? super Object> <: List<? super Integer>`

`list.add(new Object())`

> `list.add(new Object())` *is invalid since* `list` *could refer to* `ArrayList<Integer>`. *However,* `list.add(1)` *is fine;* `Integer i = list.get(0)` *is invalid.*

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

> *Error. A generic type cannot be primitive type.*

(e) `List<? super Integer> list = new ArrayList();`

> *Compiles, but with a unchecked conversion warning. Use of raw type should be avoided.*

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

> `List<?>` *can refer to all lists!* `list` *assignment is valid since*
> `ArrayList<String> <: List<String> <: List<?>`.

`list.add("abc");`

> `list.add("abc")` *is invalid since* `list` *could refer to* `ArrayList<Integer>`.