CS2030S
AY22/23 S2 **Problem Set 3** 8-9 February 2023

1. Suppose we have the following types:

   - `SubR` <: `R` <: `SuperR`
   - `SubE` <: `E` <: `SuperE` <: `Exception`

   We have the following class `A`. The implementation of the method `foo` and other details in `A` are omitted.

   ```
   class A {
     R foo() throws E { ... }
   }
   ```

   Now, suppose we have a class `B` that inherits from `A`. `B` overrides the method `foo` in `A`. Consider the following method declaration of `foo` in `B`. Which would violate the substitutability of `A` by `B` and thus should not be allowed? Explain your answer in the context of the code snippet below:

   ```
   void bar(A a) {
     try {
       R r = a.foo();
       // use r
     } catch (E e) {
       // handle exception
     }
   }
   ```

   (a) `SubR foo() throws E { ... }`

   (b) `SuperR foo() throws E { ... }`

   (c) `R foo() throws SubE { ... }`

   (d) `R foo() throws SuperE { ... }`

2. Java provides an abstract class called `Number` that is the superclass of all primitive wrapper classes. `Number` is also the superclass of `BigInteger`, a class that supports arbitrary-precision integers. The primitive wrapper classes and `BigInteger` implement the `Comparable<T>` interface.

   Ah Beng first wrote the following method to convert an array of `BigInteger` to an array of `short` values. The method takes in a parameter `threshold`. Any value larger than the threshold is set to 0.

   ```
   public static short[] toShortArray(BigInteger[] a, BigInteger threshold) {
     short[] out = new short[a.length];
     for (int i = 0; i < a.length; i += 1) {
       if (a[i].compareTo(threshold) <= 0) {
         out[i] = a[i].shortValue();
       }
     }
     return out;
   }
   ```

   As he continued to code, he realized that he also needed to convert an array of `Integer` and an array of `Double` to an array of `short`. He thus duplicated his method above and replaced `BigInteger` with `Integer` and `Double` respectively. He ended up with two more methods:

```java
public static short[] toShortArray(Integer[] a, Integer threshold) {
  short[] out = new short[a.length];
  for (int i = 0; i < a.length; i += 1) {
    if (a[i].compareTo(threshold) <= 0) {
      out[i] = a[i].shortValue();
    }
  }
  return out;
}

public static short[] toShortArray(Double[] a, Double threshold) {
  short[] out = new short[a.length];
  for (int i = 0; i < a.length; i += 1) {
    if (a[i].compareTo(threshold) <= 0) {
      out[i] = a[i].shortValue();
    }
  }
  return out;
}
```

Soon, he realized that he needed to do this for all other wrapper classes. Instead of overloading the method `toShortArray` multiple times, he decided to write a single method that generalizes the above methods.

(a) His first few attempts below, however, did not work correctly. Explain why these attempts are not correct.

(i)

```java
public static short[] toShortArray(Object[] a, Object threshold) {
  short[] out = new short[a.length];
  for (int i = 0; i < a.length; i += 1) {
    if (a[i].compareTo(threshold) <= 0) {
      out[i] = a[i].shortValue();
    }
  }
  return out;
}
```

(ii)

```java
public static short[] toShortArray(Number[] a, Number threshold) {
  short[] out = new short[a.length];
  for (int i = 0; i < a.length; i += 1) {
    if (a[i].compareTo(threshold) <= 0) {
      out[i] = a[i].shortValue();
    }
  }
  return out;
}
```

(iii)

```java
public static short[] toShortArray(Comparable[] a, Comparable threshold) {
  short[] out = new short[a.length];
  for (int i = 0; i < a.length; i += 1) {
    if (a[i].compareTo(threshold) <= 0) {
      out[i] = a[i].shortValue();
    }
  }
  return out;
}
```

(b) Ah Beng discovered that Java supports generics. Particularly, he found that a type parameter can have multiple bounds using the `&` symbol. For instance, `<T extends S1 & S2>` means that the type variable `T` is a subtype of both `S1` and `S2`[1].

Using generics with bounded type parameters, help Ah Beng to re-write all his methods into a single generic method.

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

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

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

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

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

  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);
  }
}
```

Past Year Questions

These questions are provided here for discussion among yourselves (e.g., on Piazza). We will not discuss these during the recitations. All questions are taken from **Midterm 2020/21 Semester 2.**

4. Consider the following generic class:

```java
class Wrapper<U extends Comparable<U>> {
  U value;
}
```

After type erasure, what will the type of `value` be?

A. `Object`

B. `Comparable<U>`

C. `Comparable`

D. `Wrapper`

[1]A class bound must be specified first before an interface bound

5. Consider the following:

```
interface I {
}

abstract class A<T> {
}

class C extends A<Integer> implements I {
}
```

For each statement below, indicates if it compiles without any error or warning. Please provide a rationale for your answer.

(a) `I i = new A<Integer>();`

(b) `I i = new C();`

(c) `A<String> a = new C();`

6. Consider the following classes `Main` and `SSHClient`, where:

`PasswordIncorrectException` <: `AuthenticationException` <: `Exception`

```
class Main {
  void start() {
    try {
      SSHClient client = new SSHClient();
      client.connectPENode();
    } catch (Exception e) {
      System.out.println("Main");
    }
  }
}

class SSHClient {
  void connectPENode() throws Exception {
    try {
      // Line A (Code that could throw an exception)
    } catch (AuthenticationException e) {
      System.out.println("SSHClient");
    }
  }
}
```

After calling:

```
new Main().start()
```

(a) What would be printed if an `Exception` is thrown from Line A of `connectPENode`?

(b) What would be printed if an `AuthenticationException` is thrown from Line A of `connectPENode`?

(c) What would be printed if a `PasswordIncorrectException` is thrown from Line A `connectPENode`?