CS2030 Programming Methodology
Semester 1 2019/2020

4 October 2019
Problem Set #5
Local classes and Lambda Expressions

1. For each of the questions 1a and 1b below, suppose the following is invoked:

   ```
   B b = new B();
   b.f();
   ```

   Sketch the content of the stack, heap and metaspace *immediately after* the line

   ```
   A a = new A();
   ```

   is executed. Label the values and variables/fields clearly. You can assume `b` is already on the heap and you can ignore all other content of the stack and the heap before `b.f()` is called.

   (a)
   ```
   class B {
       static int x = 0;

       void f() {
           A a = new A();
       }

       static class A {
           int y = 0;

           A() {
               y = x + 1;
           }
       }
   }
   ```

   (b)
   ```
   class B {
       void f() {
           int x = 0;

           class A {
               int y = 0;
               A() {
                   y = x + 1;
               }
           }

           A a = new A();
       }
   }
   ```

   (c)
   ```
   class A {
       int x = 1;

       void f() {
           int y = 2;

           class B {
               void g() {
                   x = y;
               }
           }

           B b = new B();
           b.g();
       }
   }
   ```

2. Java implements lambda expressions as anonymous classes. Suppose we have the following lambda expression `Function<String,Integer>`:

   ```
   Function<String,Integer> findFirstSpace = str -> str.indexOf(' ');
   ```

   Write the equivalent anonymous class for the expression above.

3. Suppose we have a class `A` that implements the following methods:

```
class A {
    int x;
    boolean isPositive;

    static A of(int x) {
        A a = new A();
        a.x = x;
        a.isPositive = (x >= 0);
        return a;
    }

    A foo(Function<Integer, A> map) {
        return map.apply(this.x);
    }

    A bar(Function<Integer, A> map) {
        if (this.isPositive) {
            return map.apply(this.x);
        } else {
            return A.of(this.x);
        }
    }
}
```

Which of the following conditions hold for `A` for all values of `x`? `f` and `g` are both variables of type `Function<Integer,A>`; `a` is an object of type `A`.

(a) `A.of(x).foo(f)` always returns `f.apply(x)`

(b) `a.foo(f).bar(g)` equals to `a.foo(x -> f.apply(x).bar(g))`

(c) `a.bar(f).bar(g)` equals to `a.bar(x -> f.apply(x).bar(g))`

4. Write your own `Optional` class with the following skeleton:

```
class Optional<T> {
    private final T value;

    public static <T> Optional<T> of(T v) {
        :
    }

    public static <T> Optional<T> ofNullable(T v) {
        :
    }

    public static <T> Optional<T> empty(T v) {
        :
    }

    public void ifPresent(Consumer<? super T> consumer) {
        :
    }

    public Optional<T> filter(Predicate<? super T> predicate) {
        :
    }

    public <U> Optional<U> map(Function<? super T, ? extends U> mapper) {
        :
    }

    public<U> Optional<U> flatMap(Function<? super T, Optional<U>> mapper) {
        :
    }

    public T orElseGet(Supplier<? extends T> other) {
        :
    }
}
```