CS2030 Programming Methodology
Semester 2 2021/2022

9 & 10 March 2022
Problem Set #6

1. For each of the questions 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 B {
       int x = 1;

       void f() {
           int y = 2;

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

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

2. Consider the following Stack implementation.

```
public class Stack<T> {
    private T head;
    private Stack<T> tail;
    private static Stack<?> EMPTYSTACK = new Stack<>(null,null);

    private Stack(T head, Stack<T> tail){
        this.head = head;
        this.tail = tail;
    }

    public void push(T t){
        this.tail = new Stack<T>(this.head, this.tail);
        this.head = t;
    }

    public void pop(){
        if (this.tail == null) {
            throw new RuntimeException("Stack is empty");
        }
        this.head = this.tail.head();
        this.tail = this.tail.tail;
    }

    public T head(){
        if (this.tail == null) {
            throw new RuntimeException("Stack is empty");
        }
        return head;
    }

    public boolean isEmpty(){
        if (this.tail == null) {
            return true;
        } else {
            return false;
        }
    }

    public static <T> Stack<T> getEmptyStack(){
        @SuppressWarnings("unchecked")
        Stack<T> emptyStack = (Stack<T>) EMPTYSTACK;
        return emptyStack;
    }
}
```

```
Stack<Integer> s = Stack.getEmptyStack();
s.push(1);
s.push(2);
s.push(3);
s.head()
s.pop()
s.head();
s.pop()
s.head();
s.pop()
```

Lets change the implementation of Stack to make it immutable. Create a new class ImmutableStack.