CS2030S Recitation: Problem Set #6 Wk 8, Sem 1, 2022/23

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

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

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

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

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

(a) Problem #A

```java
class B {
  static
   int x = 0;

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

  static class A {
    int y = 0;

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

(b) Problem #B

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

    class A {
      int y = 0;

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

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

(c) Problem #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 on the next page. Try running the following code.

```
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();
```

Change the implementation of `Stack` to make it immutable and create a new class `ImmutableStack`.

```java
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.head == null) {
      throw new RuntimeException("Stack is empty");
    }
    this.head = this.tail.head;
    this.tail = this.tail.tail;
  }

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

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

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