CS 61B Spring 2024

Scope, Pass-by-Value, Static

Exam-Level 01: January 22, 2024

1 Quik Maths

(a) Fill in the blanks in the main method below. (Fall '16, MT1)

```
public class QuikMaths {
    public static void multiplyBy3(int[] A) {
        for (int i = 0; i < A.length; i += 1) {
            int x = A[i];
            x = x * 3;
        }
    }

    public static void multiplyBy2(int[] A) {
        int[] B = A;
        for (int i = 0; i < B.length; i+= 1) {
            B[i] *= 2;
        }
    }

    public static void swap(int A, int B) {
        int temp = B;
        B = A;
        A = temp;
    }

    public static void main(String[] args) {
        int[] arr = new int[]{2, 3, 3, 4};
        multiplyBy3(arr); // Value of arr: {__2__,__3__,__3__,__4_____}

        arr = new int[]{2, 3, 3, 4};
        multiplyBy2(arr); // Value of arr: {__4__,__6__,__6__,__8_____}

        int a = 6;
        int b = 7;
        swap(a, b); // Value of a: ___6___  Value of b: ___7___
    }
}
```

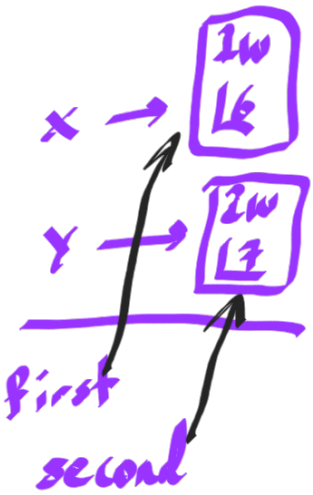
Isnt directly changing elements of A, just assigning it to variable x and tripling it

accessing each element at a given index and multiplying by 2

doesnt impact the scope of main a/b stay the same

(b) Now take a look at the code below. How could we write 'swap' to perform swapping primitive variables in a function? Be sure to use the `IntWrapper` class below.

```
class IntWrapper {
    int x;
    public IntWrapper(int value) {
        x = value;
    }
}

public class SwapPrimitives {
    public static void main(String[] args) {
        int a = 6;
        int b = 7;

        IntWrapper x = new IW(a);

        IntWrapper y = new IW(b);

        swap(___x_____, ___y_____);

        a = __x.x_____________________;

        b = __y.x_____________________;

    }

    public static void swap(__IW first__, __IW second__) {

        Int temp = first.x ;

        first.x = second.x ;

        second.x = first.x ;
    }

}
```



2 Static Books

Suppose we have the following `Book` and `Library` classes.

```
class Book {
    public String title;
    public Library library;
    public static Book last = null;

    public Book(String name) {
        title = name;
        last = this;
        library = null;
    }

    public static String lastBookTitle() {
        return last.title;
    }
    public String getTitle() {
        return title;
    }
}
```

```
class Library {
    public Book[] books;
    public int index;
    public static int totalBooks = 0;

    public Library(int size) {
        books = new Book[size];
        index = 0;
    }

    public void addBook(Book book) {
        books[index] = book;
        index++;
        totalBooks++;
        book.library = this;
    }
}
```

(a) For each modification below, determine whether the code of the `Library` and `Book` classes will compile or error if we **only** made that modification, i.e. treat each modification independently.

1. Change the `totalBooks` variable to **non** `static` compile
2. Change the `lastBookTitle` method to **non** `static` compile
3. Change the `addBook` method to `static` wont compile
4. Change the `last` variable to **non** `static` wont compile
5. Change the `library` variable to `static` compile

we're not allowed to access instance variables with static methods. If we're to invoke a static method with a classname. Library.addBook, this wouldnt make any sense, for Library class to have access to instance variables Instance variable is specific to an instance.

(b) Using the original `Book` and `Library` classes (i.e., without the modifications from part a), write the output of the `main` method below. If a line errors, put the precise reason it errors and continue execution.

```
1  public class Main {
2      public static void main(String[] args) {
3          System.out.println(Library.totalBooks);                    ____0____
4          System.out.println(Book.lastBookTitle());                  ____null Pointer exception____
5          System.out.println(Book.getTitle()); error                 no instance of class
6                                                                       non static method
7          Book goneGirl = new Book("Gone Girl");                       not allowed to access instance method
8          Book fightClub = new Book("Fight Club");
9
10         System.out.println(goneGirl.title);                        ____"Gone Girl"____
11         System.out.println(Book.lastBookTitle());                  ____"Fight club"____
12         System.out.println(fightClub.lastBookTitle());             ____"Fight club"____
13         System.out.println(goneGirl.last.title);                   ____"Fight club"____
14
15         Library libraryA = new Library(1);
16         Library libraryB = new Library(2);
17         libraryA.addBook(goneGirl);
18
19         System.out.println(libraryA.index);                        ____1____
20         System.out.println(libraryA.totalBooks);                   ____1____
21
22         libraryA.totalBooks = 0;
23         libraryB.addBook(fightClub);
24         libraryB.addBook(goneGirl);
25
26         System.out.println(libraryB.index);                        ____2____
27         System.out.println(Library.totalBooks);                    ____2____
28         System.out.println(goneGirl.library.books[0].title);       "Fight club"
29     }                                  libraryB
30 }
```