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: {__________________________}

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

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

(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;

        ________________________________________;

        ________________________________________;

        swap(__________, __________);

        a = ____________________________________;

        b = ____________________________________;

    }

    public static void swap(________________, ________________) {

        ________________________________________;

        ________________________________________;

        ________________________________________;
    }

}
```

Solution:

Part (a): Click here for visualizer link

line 23: /* Value of arr: {2, 3, 3, 4} */, because we are changing a copy of each element, not the original elements.
The enhanced **for** loop also has a similar effect to **this**.
line 28: /* Value of arr: {4, 6, 6, 8} */, because B and A point to the same underlying array.
line 34: /* Value of a: 6 Value of b: 7 */, Java is pass by value, so you are only swapping copies of the original integers.

Part (b):

```
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 first = new IntWrapper(a);
        IntWrapper second = new IntWrapper(b);
        swap(first, second);
        a = first.x;
        b = second.x;
    }

    public static void swap(IntWrapper first, IntWrapper second) {
        int temp = first.x;
        first.x = second.x;
        second.x = temp;
    }
}
```

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

Solution:

1. Compile
2. Compile
3. Error, cannot access instance variable books in a static method.
4. Error, cannot access instance variable last in a static method.
5. Compile

(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.

Solution: Click here for visualizer link

```
1  public class Main {
2      public static void main(String[] args) {
3          System.out.println(Library.totalBooks);                 0
4          System.out.println(Book.lastBookTitle());               Error, NullPointerException
5          System.out.println(Book.getTitle());                    Error, does not compile
6
7          Book goneGirl = new Book("Gone Girl");
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     }
30 }
```