**COMP 110/L**
**Fall 2022**

**Midterm #2 Written Practice Exam with Solutions**

1.) Consider the following code, which calls an unseen myMethod method.

```
public class MyClass {
  //
  // myMethod is defined somewhere here
  //

  public static void main(String[] args) {
    int x = myMethod(3.14, "hello");
  }
}
```

Write down the signature of `myMethod` below. You don't need to provide the body, just the signature.

```
public static int myMethod(double x, String y)
```

2.) What is wrong with the following code:

```
public class SomeClass {
  private int x;
  public SomeClass(int param) {
    param = x;
  }
}
```

Instance variable `x` is never initialized; should be `x = param`

3.) The `Blah` class' constructor takes an `int` and a `String`. Create an instance of `Blah` below, using any input of the right type, saving a reference to the `Blah` instance in a variable.

```
Blah x = new Blah(7, "foo");
```

4.) Assume the `main` method in the class below is called. What is the output?

```
public class Something {
  private int x;
  public Something(int x) {
    this.x = x;
  }
  public int something() {
    return x + 1;
  }
  public static void main(String[] args) {
    Something x = new Something(2);
    System.out.println(x.something());
  }
}
```

3

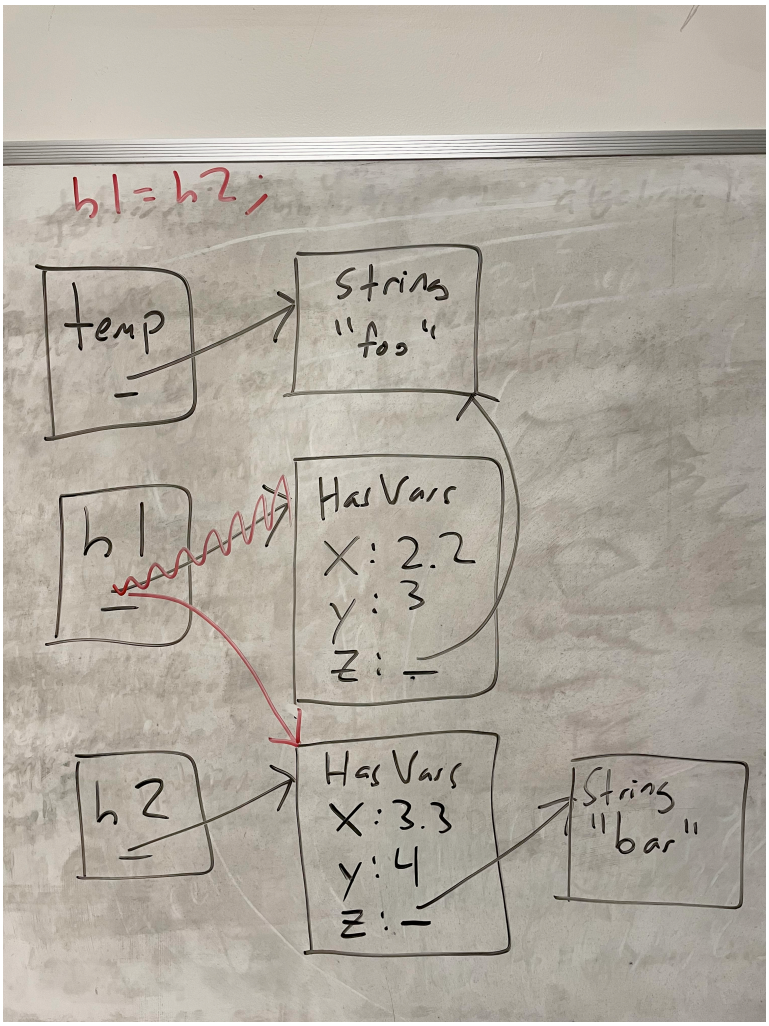
6.) What is wrong with the following code? Assume that `First` and `Second` are defined in `First.java` and `Second.java`, respectively.

```
public class First {
  private int first;
  public First(int first) {
    this.first = first;
  }
  public int getFirst() {
    return first;
  }
}

public class Second {
  public static void main(String[] args) {
    First first = new First(123);
    System.out.println(first.first);
  }
}
```

Instance variable `first` is marked as `private`, so it's not accessible in `Second.java`. You could fix this by using `first.getFirst()` instead of `first.first` in `Second`'s `main` method.

7.) Draw a memory diagram representing how memory looks just before `main` terminates.

```
public class HasVars {
  private double x;
  private int y;
  private String z;
  public HasVars(double x, int y, String z) {
    this.x = x;
    this.y = y;
    this.z = z;
  }

  public static void main(String[] args) {
    String temp = "foo";
    HasVars h1 = new HasVars(2.2, 3, temp);
    HasVars h2 = new HasVars(3.3, 4, "bar");
    h1 = h2;
  }
}
```



8.) What is the output of the following code?

```
int x = 7 / 3;
int y = 9 % 4;
System.out.println(x);
System.out.println(y);
```

```
2
1
```

9.) Someone has written the following code, where the program is intended to randomly print `rock`, `paper`, or `scissors`:

```
import java.util.Random;

public class RockPaperScissors {
  public static void main(String[] args) {
    Random r = new Random(123L);
    switch (r.nextInt(2)) {
      case 0:
        System.out.println("rock");
      case 1:
        System.out.println("paper");
      case 2:
        System.out.println("scissors");
    }
  }
}
```

9.a.) Upon running this program, multiple things get printed, instead of just one. Why are multiple things printed, and how could this be fixed?

There is no break after each case, so it falls through to the next case. `break` statements should be added after each case.

9.b.) After fixing 9.a., it's noticed that the program will always make the same choice, no matter what. Why is this happening, and how could this be fixed?

A fixed seed is being passed to `Random`'s constructor, causing it to always generate the same number. Instead, no seed should be passed.

9.c.) After fixing 9.b., it's noticed that the program will never pick `scissors`. Why is this happening, and how could this be fixed?

`r.nextInt(2)` will only ever return 0 or 1, so `case 2` will never be reached. This should be `r.nextInt(3)` instead.

10.) Consider the following code, which uses if/else:

```
if (x == 5) {
  System.out.println("five");
} else if (x == 6) {
  System.out.println("six");
} else if (x == 7) {
  System.out.println("seven");
} else {
  System.out.println("other");
}
```

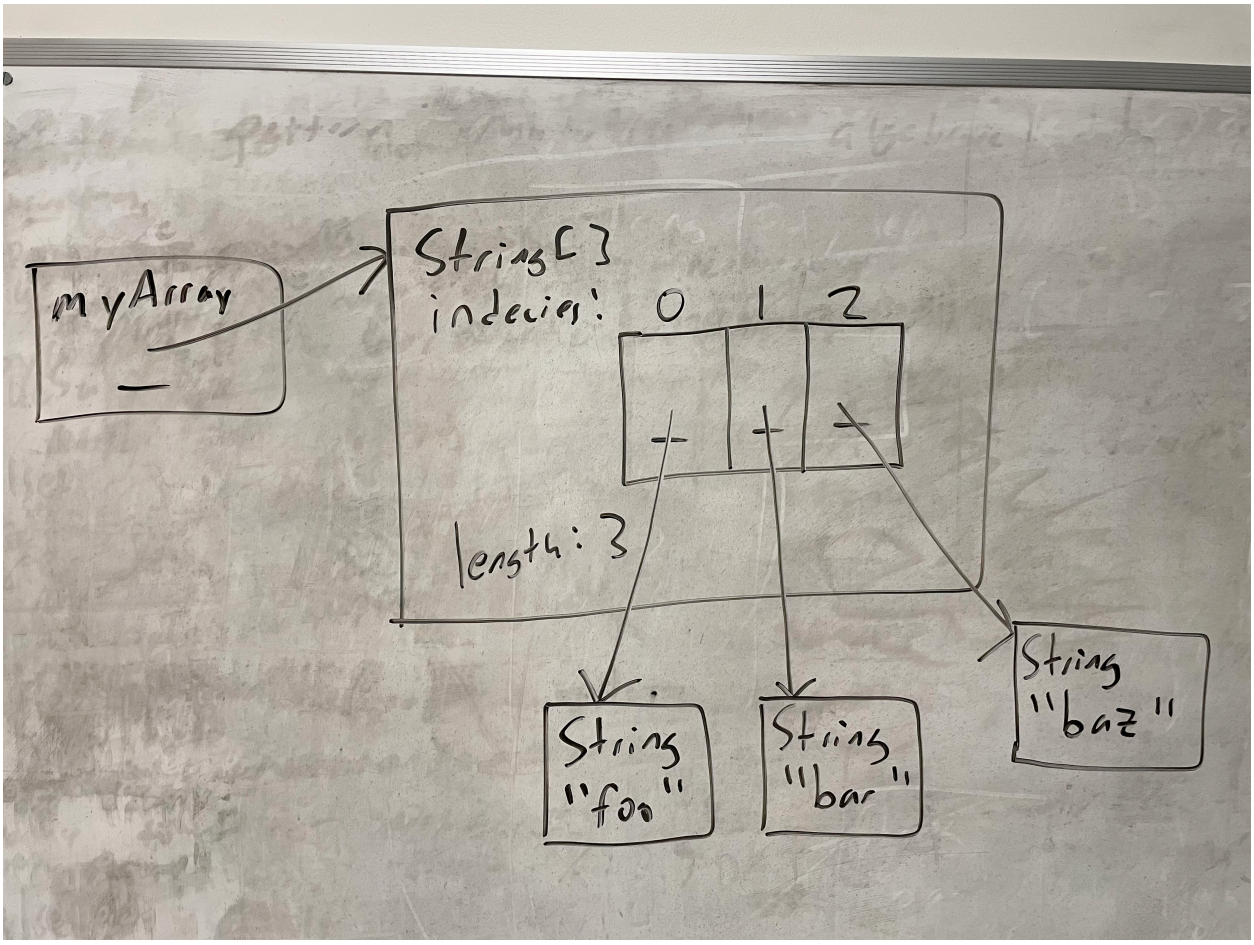
Write an equivalent `switch` below.

```
switch (x) {
  case 5:
    System.out.println("five");
    break;
  case 6:
    System.out.println("six");
    break;
  case 7:
    System.out.println("seven"
    break;
  default:
    System.out.println("other");
    break;
}
```

11.) Consider the following code:

```
String[] myArray = new String[3];
myArray[0] = "foo";
myArray[1] = "bar";
myArray[2] = "baz";
```

Write a memory diagram representing how this looks in memory.



12.) What is wrong with the following code?

```
int[] array = new int[2];
array[0] = 5;
array[1] = 7;
System.out.println(array[2]);
```

`array[2]` does not exist, so this would crash if you try to run it.

13.) What is the output of the following code?

```
double[] arr = new double[3];
arr[0] = 3.3;
arr[1] = arr[0];
arr[2] = 4.4;
arr[0] = 5.5;
System.out.println(arr[0]);
System.out.println(arr[1]);
System.out.println(arr[2]);
System.out.println(arr.length);
```

```
5.5
3.3
4.4
3
```

14.) What is wrong with the following code?

```
int temp = 0;
while (temp < 10) {
  System.out.println(temp);
}
System.out.println("after loop");
```

This is an infinite loop, so `after loop` will never be printed. It looks like `temp` is intended to be incremented somewhere in the body of the loop, as with `temp++;`