Variables, Memory, and Methods

Let's review some key concepts surrounding variables, how they are managed in memory, and how they are handled when passed as arguments to methods. Here are some broad concepts that you should be able to discuss before starting these practice questions.

- Variables come in two types in Java - primitive and reference.
- Java variables are always copied to the method parameters when passed into methods as arguments.
- Using == in conditions with variables is always comparing the value of the variable/s - but whether the value is a value or a reference value is important to know.
- String are immutable.

```
public class Tester {                    //some methods in this class may not work as intended!

    public void increase(int x) {
        x=x+1;
    }

   public double square(double x) {
        return(x*x);
   }

   public void addDoctor(String name) {
        name = "Dr. " + name;
   }

   public void reduce(int[] A) {
        for(int k=0; k<A.length; k++)
              A[k]--;
   }

  public void removeLast(int[] A) {
        int[] temp=new int[A.length-1];
        for(int k=0; k<A.length-1; k++)
              temp[k]=A[k];
        A=temp;
  }

   public void assignPerfect(Student S) {
        S.grade=100;
   }

  public Student changeName(Student S, String newName) {
        S.name=newName;
        return(S);
  }

} //end of class Tester
```

Assume that the following code is inside a runner class. What are the outputs? Assume *print* is a method.

```
Tester T= new Tester();

int num=5;
T.increase(num);
print(num);

double a=3;
double b=4;
double c = T.square(a) + T.square(b);
print(a);  print(b);  print(c);

String s=”Bob”;
T.addDoctor(s);
print(s);

int[] nums = {1,2,3};
T.reduce(nums);
for(int k=0; k<nums.length; k++)
  print( nums[k] );

nums = {1,2,3};
T.removeLast(nums);
for(int k=0; k<nums.length; k++)
  print( nums[k] );

Student Kimmy = new Student();
Kimmy.grade=50;  Kimmy.name=”Kimmy”;
T.assignPerfect(Kimmy);
print(Kimmy.grade);
```

Look at the *changeName* method of the Tester class. Is it coded well? Are there any changes you would suggest? List your reasons for the changes.

Which methods DID NOT WORK as intended?

For each method that didn’t work as intended

1. modify the method so that it could be used from class Tester and work as intended.
2. change the code in the runner class so that it will work with your modified methods.

Consider the following code in a runner class. For the following code segments you should be able to *sketch out* where and what the variables are pointing to in memory. Your sketch should show that you understand what is taking place with primitive and reference variables, and immutable Strings. Assume the methods seen in the code all work as intended.

```
int x=1;
int y=2;
x=y;
print(x==y);
print(x);
print(y);
```

Are x and y pointing to the same memory address?

```
Student Bob=new Student(“Robert”, 75);
Student Bobby=new Student(“Robert”, 75);
Student Jane=new Student(“Jane”, 90);
print(Bob==Bobby);
print(Bob.equals(Bobby));
print(Bob==Jane);
Bob=Jane;
print(Bob==Jane);
Bob.grade=Bob.grade+10;
print(Bob.grade);
print(Bobby.grade);
print(Jane.grade);
Jane=Bob;
Jane.grade=Jane.grade-20;
print(Jane.grade);
print(Bob.grade);
```

```
int[] A=new int[3];
A[0]=0; A[1]=1; A[2]=2;
int[] B=A;
B[0]=99;
A[2]=B[0];
print entire array A
print entire array B
print(A==B);
```

```
String s="hi";
String w="hi";
System.out.println(s==w);
System.out.println(s.equals(w));
System.out.println("*");

String s2=new String("hi");
System.out.println(s==s2);
System.out.println(s.equals(s2));
System.out.println("**");

s2="hi";
System.out.println(s==s2);
System.out.println(s.equals(s2));
System.out.println(s2==w);
System.out.println("***");

s="bye";
System.out.println(s==s2);
System.out.println(s==w);
System.out.println(s2==w);
System.out.println(s.equals(s2));
System.out.println(s.equals(w));
System.out.println(s2.equals(w));
System.out.println("****");

s2=s;
System.out.println(s==s2);
System.out.println(s==w);
System.out.println(s2==w);
System.out.println(s.equals(s2));
System.out.println(s.equals(w));
System.out.println(s2.equals(w));
System.out.println("*****");
```