1. Consider the class below:

```java
public class Mystery{
    private int x;
    private int [] data;

    public Mystery(){
        x = 3;
        data = new int [ ] { 1, 2, 3};
    }

    public void run() {
        f();
        k(data);
        x = data[0];
        System.out.println("x is " + x);
    }

    public void f() {
        x = x * 100;
        g(x);
        x = x + h(x);
        System.out.println("x is " + x);
    }

    public void g(int x) {
        x = x * 10;
        System.out.println("x is " + x);
    }

    public int h(int x) {
        x = x * 2;
        System.out.println("x is " + x);
        return this.x * 2;
    }

    public void k(int[] info) {
        info[0] = info[1] * 2;
        int x = info[0];
        System.out.println("x is " + x);
    }
}
```

What is printed when the following statements are executed? Show enough work to receive full credit.

Mystery m = new Mystery();
m.run();

x is 3000
x is 600
x is 900
x is 4
x is 4
2. Given the following code:

```java
public class Driver{
    private String name;
    
    public Driver(String name){
        this.name = name;
    }
}

public class Car{
    private Driver d;
    
    public Car(Driver d) {
        this.d = d;
    }
    
    public void setDriver(Driver d) {
        this.d = d;
    }
}
```

Consider the following statements in the main method:
Driver d4 = new Driver ("Anna");
Car c = new Car(d4);
Driver d5 = new Driver ("Tom");
c.setDriver(d5);

Does the driver for the Car c change from above , if we execute the following statement?

```java
d5 = d4;
```

No. The driver does for the car object referred by c does not change.
3. Consider the following code:

```java
public class WeatherRecord {
    private double inchesOfRain;
    private double hiTemp;

    public WeatherRecord(double inchesOfRain, double hiTemp){
        this.inchesOfRain = inchesOfRain;
        this.hiTemp = hiTemp;
    }

    public double getRainfall() { return inchesOfRain; }
    public double getHiTemp() { return hiTemp; }
}

Sample Dr Java Interactions:
// Analyze the weather for a hypothetical 3-day month
WeatherRecord day1 = new WeatherRecord(0, 60); // 0 inches of rain, hi temp = 60
WeatherRecord day2 = new WeatherRecord(3, 65);
WeatherRecord day3 = new WeatherRecord(1, 68); // for simplicity, this month has 3 days
WeatherRecord[] record = new WeatherRecord[]{day1, day2, day3};
WeatherReporter reporter = new WeatherReporter();
WeatherRecord monthRecord = reporter.computeStats(record);
System.out.println(monthRecord.getRainfall()); // average rainfall for the month
Output: 1.3333333333333333
System.out.println(monthRecord.getHiTemp()); // highest temperature of the month
Output: 68.0

A WeatherRecord (code supplied above) is flexible in that it can hold weather information for a
day, a month, even a year or a millenium.
A WeatherReporter has a single method called computeStats which takes an array of
WeatherRecords as input, analyzes it, and returns a WeatherRecord which contains the results of
the analysis. Complete the computeStats method. You can assume that input array is valid (i.e.
not null or length is greater than 0) and that it is full (contains no nulls).

```
4. Assume that statement below compile and runs without errors:

> A data = new C();

   a) What can we infer about relationship between A and C?

   A is supertype and C is subtype (C is a A). This is due to type rules – a reference variable of type t may refer to an object of its own type or any subtype.

   b) Assume B is a class that inherit from A. Class B has a method b() but C does not. Class C has a methods c() but B does not. Consider the following:

> A data = new C();

For the following interaction answer **yes or no** if the code compiles with error. If it compiles does it have run-time/semantic errors. Explain.

> ((C)data).c();

Compiles. Without the cast it would be compile error as c() is not in A i.e. superclass and variable data is of type A. It also executes without errors as data is referring to an instance of C.

> ((B)data).b();

Compiles. Without the cast it would be compile error as b() is not in A i.e. superclass and variable data is of type A. However, it does not execute as data is referring to an instance of C. We cannot convert an instance of type C to a B.