Assignment 1
Comparing value types and reference types

Passing by value is assigning value into an object by its value. It is used for primitive type only, not an object.

Passing by reference is assigning the reference so it wil have same value like the reference.

Example:
1. Passing by value for primitive type

```
int x = 1;
int y = x;

System.out.println("x: " + x); // x: 1
System.out.println("y: " + y); // y: 1

y = 2;

System.out.println("y: " + y); // y: 2
```

2. Passing by reference for an object, the n1 will also change because n2 use n1 reference

```
MyNumber n1 = new MyNumber();
n1.val = 1;

MyNumber n2 = n1;
System.out.println("n2: " + n2.val); // 1
n2.val = 2;
System.out.println("n1: " + n1.val); // 2
System.out.println("n2: " + n2.val); // 2
```

3. Passing by value for an object, the n1 will not change because n2 use the value of n1

```
MyNumber n1 = new MyNumber();
n1.val = 1;

MyNumber n3 = new MyNumber();
n3.val = n1.val;
System.out.println("n3: " + n3.val); // 1
n3.val = 2;
System.out.println("n1: " + n1.val); // 1
System.out.println("n3: " + n3.val); // 2
```

Assignment 2

OOP principle in Java:

1. Encapsulation: object will only expose the selected information

Assume we have MyClass with attributes valNum and valString.

```
class MyClass{
    2 usages
    int valNum;
    2 usages
    String valString;
}
```

If we access an attribute, it will only give the selected attribute.

```
MyClass c1 = new MyClass();
c1.valNum = 1;
c1.valString = "hello";

System.out.println(c1.valNum); // 1
System.out.println(c1.valString); // hello
```

2. Abstraction: hides complex details to reduce complexity

Assume we have class that can give us the sum of 2 numbers.

```
class MyClass{
    1 usage
    private int valNum1;
    1 usage
    private int valNum2;
    2 usages
    private int sum;

    1 usage
    MyClass(int val1, int val2){
        this.valNum1 = val1;
        this.valNum2 = val2;
        this.sum = val1 + val2;
    }

    1 usage
    public int getSum() {
        return sum;
    }
}
```

If we want to get the sum, we can just use the getSum method after create the object so we don't need to expose the algorithm to get the sum.

```
MyClass c1 = new MyClass(1,3);
System.out.println(c1.getSum()); // 4
```

3. Inheritance: entities can inherit attributes from its parent class
Assume we have class Desk that is the child of Furniture.

```
class Furniture{
    2 usages
    int length;
    2 usages
    int height;
}

2 usages
class Desk extends Furniture{
    2 usages
    String name;
}
```

Desk will also have length and height attributes.

```
Desk d = new Desk();
d.height = 2;
d.length = 10;
d.name = "Desk 1";

System.out.println("height: " + d.height); // 2
System.out.println("length: " + d.length); // 10
System.out.println("name: " + d.name); // Desk 1
```

4. Polymorphism: entities can have more than one form
Using the Furniture and Desk class, the Desk will have its own attribute (name) and the Furniture attributes (height and length)

```java
Desk d = new Desk();
d.height = 2;
d.length = 10;
d.name = "Desk 1";

System.out.println("height: " + d.height); // 2
System.out.println("length: " + d.length); // 10
System.out.println("name: " + d.name); // Desk 1
```