Set 20: Passing by Reference and by Value

Skill 20.01: Differentiate between primitive and reference data types
Skill 20.02: Interpret the outcome of passing primitive and reference data types

Skill 20.01: Differentiate between primitive and reference data types

Skill 20.01 Concepts

Primitive data types are the most basic data types available within the Java language. There are 8: Boolean, byte, char, short, int, long, float, and double. These types serve as the building blocks of data manipulation in Java. Such types serve only one purpose — containing pure, simple values of a kind.

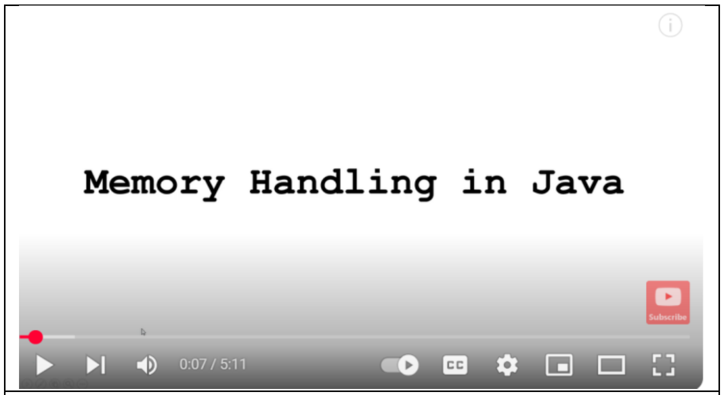
We have already explored and used int, double, Boolean, and char data types. Below are a few more,

- long: an integer which gives more digits than an int
- short: an integer which gives fewer digits than an int
- float: a floating-point number (a double is also a floating point number) that gives fewer significant figures than an double

Primitive data types always store a value

Reference data types are objects and do not store values, but instead point to a *reference* (or location) in the memory. Objects include arrays and Strings; they also include classes.

The video below provides further explanation,



https://www.youtube.com/watch?v=LTnp79Ke8FI&t=6s

Skill 20.01: Exercise 1

Skill 20.02: Interpret the outcome of passing primitive and reference data types

Skill 20.02 Concepts

The above example illustrates how different data types are handled in memory. When passing different data types to a method in a program, you should be mindful of how they are stored in the memory to avoid unexpected results.

The below example is illustrative,

BankAccount

```
public class BankAccount{
    public double balance;

    public BankAccount(double b){
        balance = b;
    }
}
```

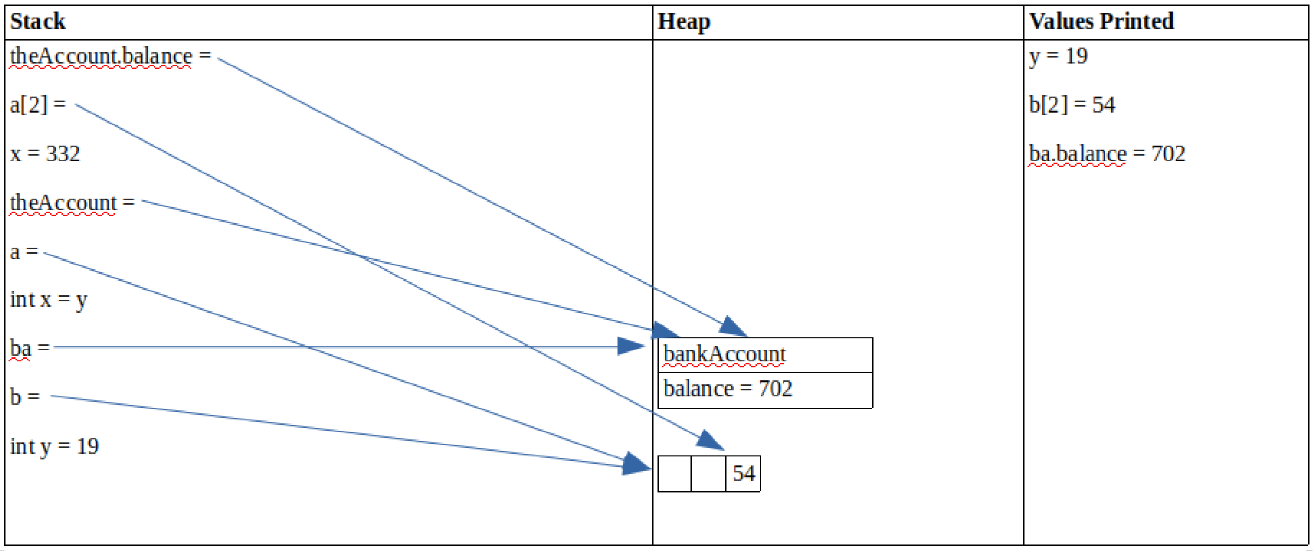
Main

```
public class Main {
    public static void main(String args[]){
        int y = 19;
        double b[] = new double[3];
        b[2] = 19;
        BankAccount ba = new BankAccount(10.0);
        method1(y, b, ba);
        System.out.println(y + " " + b[2] + " " +
ba.balance);
    }

    public static void method1(int x, double a[],
BankAccount theAccount){
        x = 332;
        a[2] = 54;
        theAccount.balance = 702;
    }
}
```

Output

```
19 54.0 702.0
```



Stack	Heap	Values Printed

theAccount.balance =
a[2] =
x = 332
theAccount =
a =
int x = y
ba =
b =
int y = 19
	bankAccount
	balance = 702
	54
		y = 19
		b[2] = 54
		ba.balance = 702

Skill 20.02: Exercises 1 thru 3