Lab 06 – Methods

Enrique Saracho Felix
100406980
CPSC 1150
26/06/2023

Exercise 1

A. Running ScopeOfVariables.java…

```
$ java ScopeOfVariables.java
In update after updating - x = 15.0
```

After removing comments:

```
$ java ScopeOfVariables.java
In main before updating - x = 10.0
In update before updating - x = 10.0
In update after updating - x = 15.0
In main after updating - x = 10.0
```

B. The value of *x* after calling update method while in the main method is still 10. It only changes to 15 inside the update method block. This is because *x* in main is a local variable.

C. Modified update method to return new value of *x* to main method.

```
$ java ScopeOfVariables.java
In main before updating - x = 10.0
In update before updating - x = 10.0
In update after updating - x = 15.0
In main after updating - x = 15.0
```

Now the update method returns a new value to main where is set in *x*.

D. Commented out line 14 and removed comment from line 4.

```
$ javac ScopeOfVariables.java
ScopeOfVariables.java:17: error: cannot find symbol
        x += ADD;
             ^
  symbol:   variable ADD
  location: class ScopeOfVariables
1 error
```

This change removed the *ADD* constant from the update method and created an *ADD* constant in the main method. The error is because the constant is local to the main method. So, when the update method tries to use it, it doesn't recognize it and creates a compile error.

E. Commented out line 4 and removed comment from line 2.

```
$ java ScopeOfVariables.java
In main before updating - x = 10.0
In update before updating - x = 10.0
In update after updating - x = 25.0
In main after updating - x = 25.0
```

This change eliminated the *ADD* constant from the update method, and created a global constant *ADD* outside the methods (in the class block). So now the update method can access its value and add it to *x*. Changing the value from 10 to 25.

F. Removed comment from line 14.

```
$ java ScopeOfVariables.java
In main before updating - x = 10.0
In update before updating - x = 10.0
In update after updating - x = 15.0
In main after updating - x = 15.0
```

This change brought back the local *ADD* constant to the update method. The update method now has access to two different *ADD* constants, the local one and the global one, but gives preference to the local one and adds its value to *x*. Making it 15.

G. The modifiers in line 2 are *final* and *static*. In line 14 is *final*.

H. Removed *static* modifier in line 2 and commented out line 14.

```
$ javac ScopeOfVariables.java
ScopeOfVariables.java:18: error: non-static variable ADD cannot be referenced from a static context
        x += ADD;
             ^
1 error
```

This change made the *ADD* constant an instance variable and can't be accessed by static methods. Also, it eliminated the *ADD* local constant from the update method, so when this method tries to use the constant's value it creates a compile error.

Exercise 2

Program SumDigits.java

File name: lab06\SumDigits.java

Purpose: To take two positive integers as input and check whether the sum of even digits of both numbers are equal.

Packages: (list of imported packages)

Limitations: (input it can't handle, list of possible error messages, round-off error)

Bugs: (list of unfixed bugs)

Input: Two positive integer numbers (*a* and *b*).

Output: A boolean value, displaying true if both sum of even digits of *a* and *b* are equal. Displaying false if otherwise.

Pseudocode:

```
Algorithm SumDigits
START

(main)
Set a and b as integer variables
a = getData()
b = getData()
Print isSumEqual(a, b)

(getData)
Set flag = false
Set num as integer variable
Do
{
        If (flag)
        {
                Display error message
        }
        Display prompt message
        Read num
        flag = true
} While (num < 0)
Return num

(isSumEqual(a, b))
Return (sumOfEven(a) = sumOfEven(b))

(sumOfEven(num))
Set sum = 0
Set digit as an integer variable
{
        digit = num % 10
        If digit % 2 = 0
        {
                sum += digit
        }
        num /= 10
```

}
Return *sum*

END SumDigits

Test run(s):

```
$ java SumDigits.java
Enter a positive integer: 1467
Enter a positive integer: 182
true
```

```
$ java SumDigits.java
Enter a positive integer: -89
Error: invalid input. Input must be a positive integer
Enter a positive integer: 8
Enter a positive integer: 344
true
```

```
$ java SumDigits.java
Enter a positive integer: -78
Error: invalid input. Input must be a positive integer
Enter a positive integer: 78
Enter a positive integer: -10
Error: invalid input. Input must be a positive integer
Enter a positive integer: 10
false
```

```
$ java SumDigits.java
Enter a positive integer: 456
Enter a positive integer: 1285
true
```

```
$ java SumDigits.java
Enter a positive integer: 123
Enter a positive integer: 122
false
```