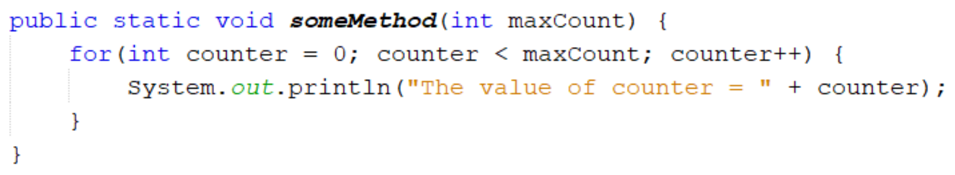
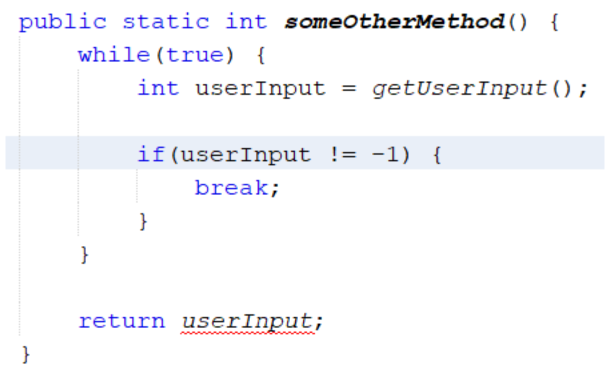
1. Look at the code below, then answer the questions:



```
public static void someMethod(int maxCount) {
    for(int counter = 0; counter < maxCount; counter++) {
        System.out.println("The value of counter = " + counter);
    }
}
```

1. What is the scope of the variable with the *identifier* **`counter`**?
2. What is the scope of the variable with the *identifier* **`maxCount`**?

2. Look at the code below, then answer the question:



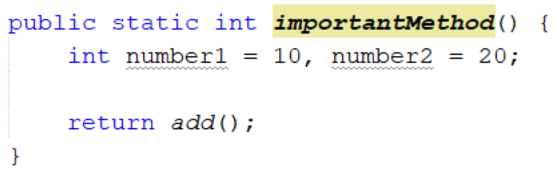
```
public static int someOtherMethod() {
    while(true) {
        int userInput = getUserInput();

        if(userInput != -1) {
            break;
        }
    }

    return userInput;
}
```

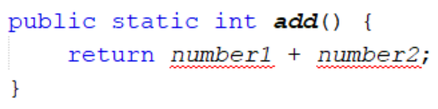
a. What is Netbeans highlighting an error with the variable ***`userInput`***?

c. Look at the code below, then answer the question:



```
public static int importantMethod() {
    int number1 = 10, number2 = 20;

    return add();
}
```



```
public static int add() {
    return number1 + number2;
}
```

a. Netbean is complaining again about variables '**`number1`**' and '**`number2`**'. Why?

d. What is a method's *signature*?

5. What is the difference between ***formal arguments*** and ***actual arguments***?

6. Write a method that accepts an array of integers and returns the sum of all the integers in the array.

7. Write a method call **`arrayJoin`** that accepts two arrays as arguments and returns a new integer array that contains all the elements of the two arrays passed as arguments. Here is its signature:

```
int[] joinArray(int[] firstArray, int[] secondArray)
```

The elements in the new array copied **`firstArray`** first, and then **`secondArray`**.

For example:

```
int[] first = {1, 2, 3};
int[] second = {7, 8, 9};

// ...

int[] newArray = arrayJoin(first, second);

// Contents of newArray: [1, 2, 3, 7, 8, 9]
```

8. Write a method called '**`arrayFlatten`**'. This method accepts as a single argument a 2-dimenional array of floating point numbers. Here is the method's signature:

```
float[] arrayFlatten(float[][] inputArray)
```

The method will return a 1-dimensional array containing all the values from the 2-dimenional array. For example:

```
float[][] arr2D = {{0.1, 0.2, 0.3},
                   {1.1, 1.2, 1.3},
                   {2.1, 2.2, 2.3}};

float[] arr1D = arrayFlatten(arr2D);

// arr1D: [0.1, 0.2, 0.3, 1.1, 1.2, 1.3, 2.1, 2.2, 2.3]
```