Software Engineering-Refactoring and Testing Assignment

Made by: Ali bani jaber

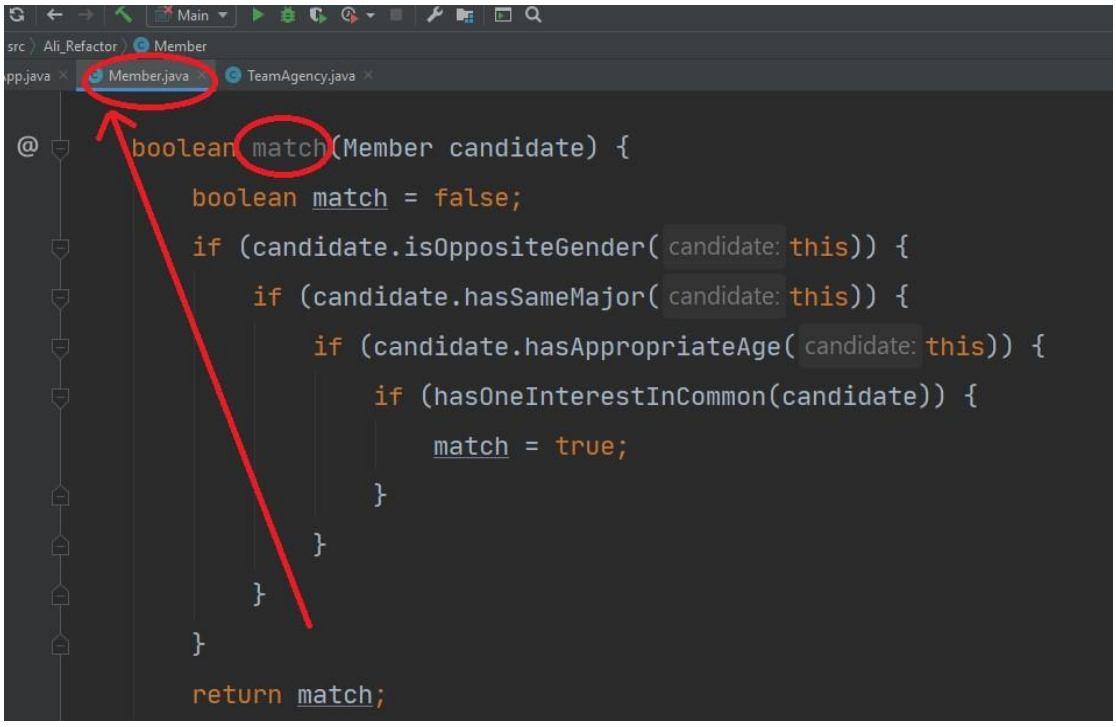
Part 1: Refactoring

1_Move Refactoring :

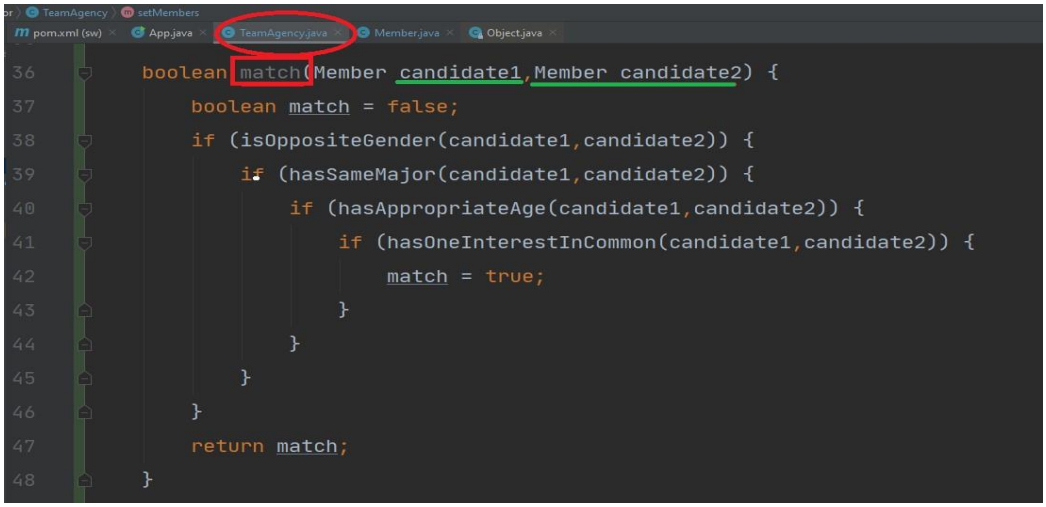
Apply this technique to the following functions: Match,hasSameMajor,hasOneInterestInCommon,hasAppropriateAge and isOppositeGender . And making necessary changes in the code . Moving these methods from Member class to TeamAgency class. In order to reduce the dependency between classes .Also Because these functions focused on the logic but not the structure of date. so I move it from Member class to TeamAgency class.

A) Method match :

Code before apply move



Code after Refactoring :

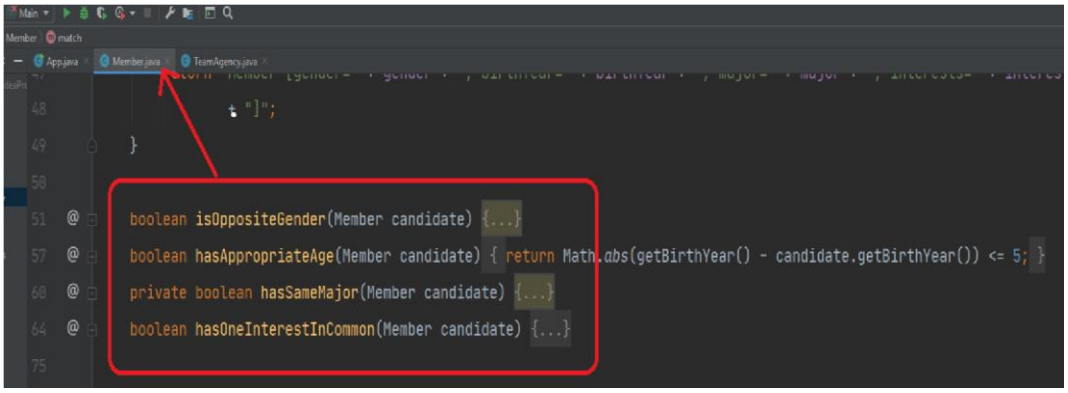


```
boolean match(Member candidate1,Member candidate2) {
    boolean match = false;
    if (isOppositeGender(candidate1,candidate2)) {
        if (hasSameMajor(candidate1,candidate2)) {
            if (hasAppropriateAge(candidate1,candidate2)) {
                if (hasOneInterestInCommon(candidate1,candidate2)) {
                    match = true;
                }
            }
        }
    }
    return match;
}
```

B)for functions

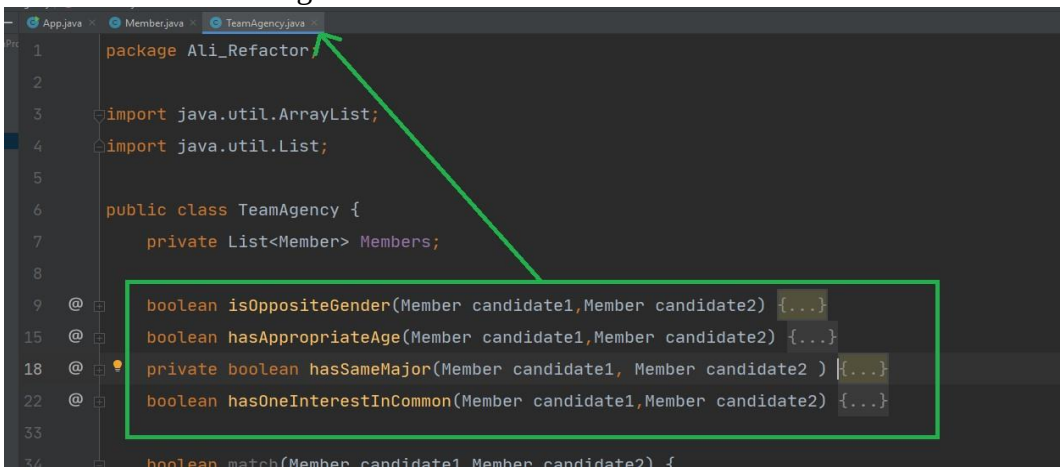
IsOppositeGender ,hasAppropriateAge, hasSameMajor and HasOneInterestInCommon

Code before Refactoring :



```
boolean isOppositeGender(Member candidate) {...}
boolean hasAppropriateAge(Member candidate) { return Math.abs(getBirthYear() - candidate.getBirthYear()) <= 5; }
private boolean hasSameMajor(Member candidate) {...}
boolean hasOneInterestInCommon(Member candidate) {...}
```

Code after Refactoring :



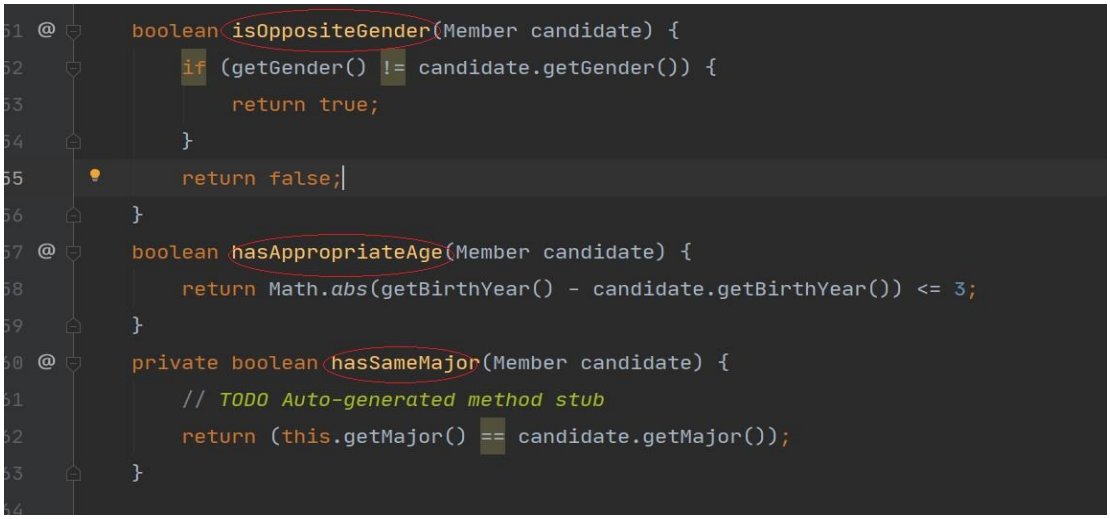
```
package Ali_Refactor;

import java.util.ArrayList;
import java.util.List;

public class TeamAgency {
    private List<Member> Members;

    boolean isOppositeGender(Member candidate1,Member candidate2) {...}
    boolean hasAppropriateAge(Member candidate1,Member candidate2) {...}
    private boolean hasSameMajor(Member candidate1, Member candidate2 ) {...}
    boolean hasOneInterestInCommon(Member candidate1,Member candidate2) {...}
```

Results : reduces dependency between classes

2_inline method:

For the following functions IsOppositeGender ,hasAppropriateAge and hasSameMajor I can apply inline Refactoring on it.

By replace calls to the method with the method's content and delete the method.



```
    boolean isOppositeGender(Member candidate) {
        if (getGender() != candidate.getGender()) {
            return true;
        }
        return false;
    }
    boolean hasAppropriateAge(Member candidate) {
        return Math.abs(getBirthYear() - candidate.getBirthYear()) <= 3;
    }
    private boolean hasSameMajor(Member candidate) {
        // TODO Auto-generated method stub
        return (this.getMajor() == candidate.getMajor());
    }
```

Code after refactoring:

Step1 : Replace all calls to the method with the method's content:

```
boolean match(Member candidate1,Member candidate2) {
    boolean match = false;
    if (candidate1.getGender()!=candidate2.getGender()?true :false) { //isOppositeGender(candidate1,candidate2)
        if (candidate1.getMajor()==candidate2.getMajor()?true:false) {//hasSameMajor(candidate1,candidate2)
            if (Math.abs(candidate1.getBirthYear() - candidate2.getBirthYear()) <= 5) {//hasAppropriateAge(candidate1,candi
                if (hasOneInterestInCommon(candidate1,candidate2)) { //We cant convert this method to inline
                    match = true;
                }
            }
        }
    }
    return match;
}
```

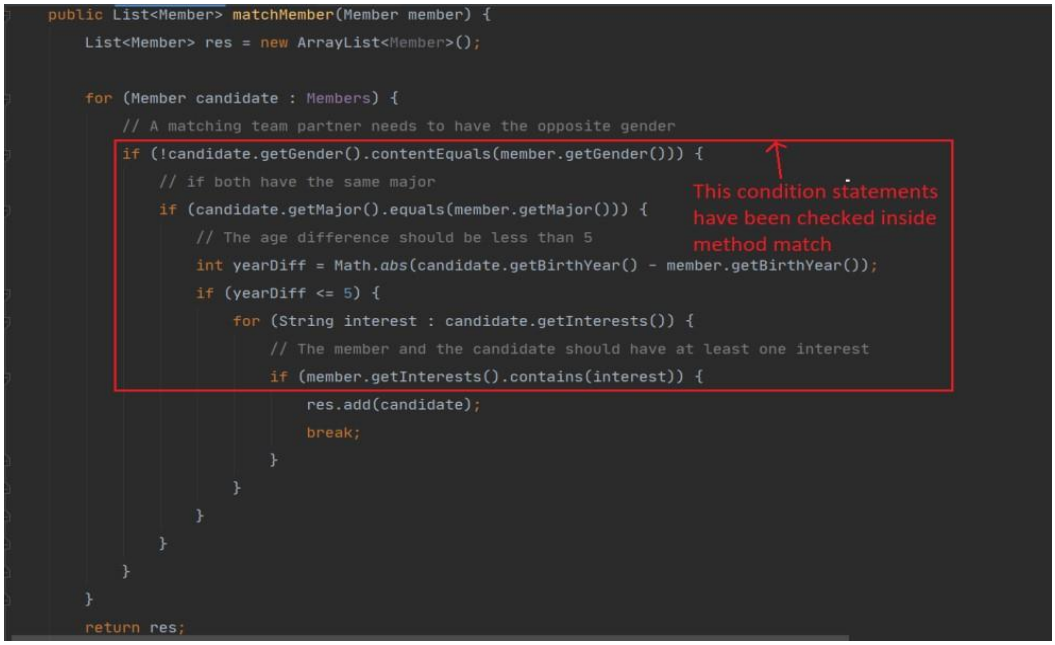
Step2 : delete the method

The Result for this Refactoring :

By minimizing the number of unneeded methods, you make the code more straightforward.

3_ Extract Method :

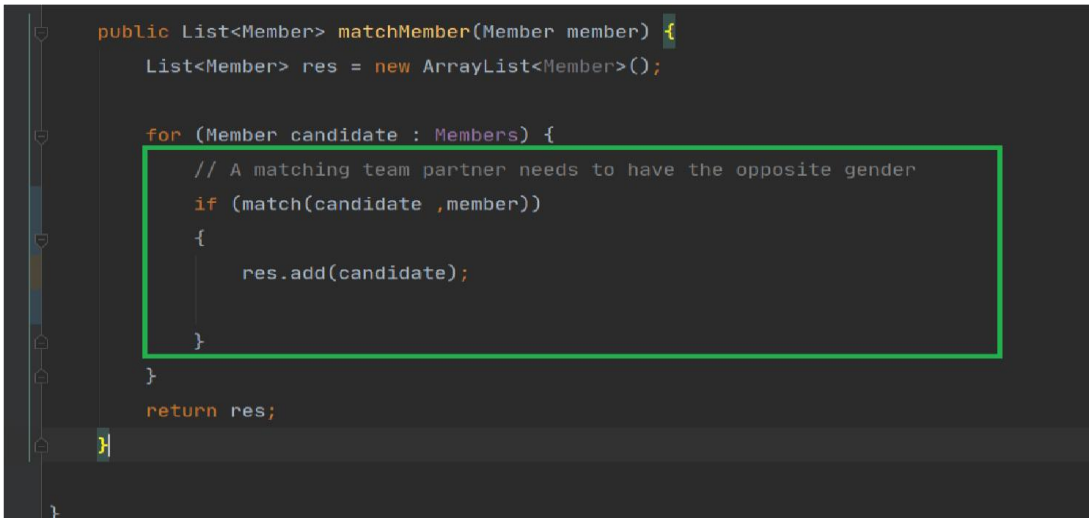
Code before Refactoring :




```
public List<Member> matchMember(Member member) {
    List<Member> res = new ArrayList<Member>();

    for (Member candidate : Members) {
        // A matching team partner needs to have the opposite gender
        if (!candidate.getGender().contentEquals(member.getGender())) {
            // if both have the same major
            if (candidate.getMajor().equals(member.getMajor())) {
                // The age difference should be less than 5
                int yearDiff = Math.abs(candidate.getBirthYear() - member.getBirthYear());
                if (yearDiff <= 5) {
                    for (String interest : candidate.getInterests()) {
                        // The member and the candidate should have at least one interest
                        if (member.getInterests().contains(interest)) {
                            res.add(candidate);
                            break;
                        }
                    }
                }
            }
        }
    }
    return res;
```

Because they are duplicate between matchMember function with match function I can replace these statements with call match function

Code after Refactoring :



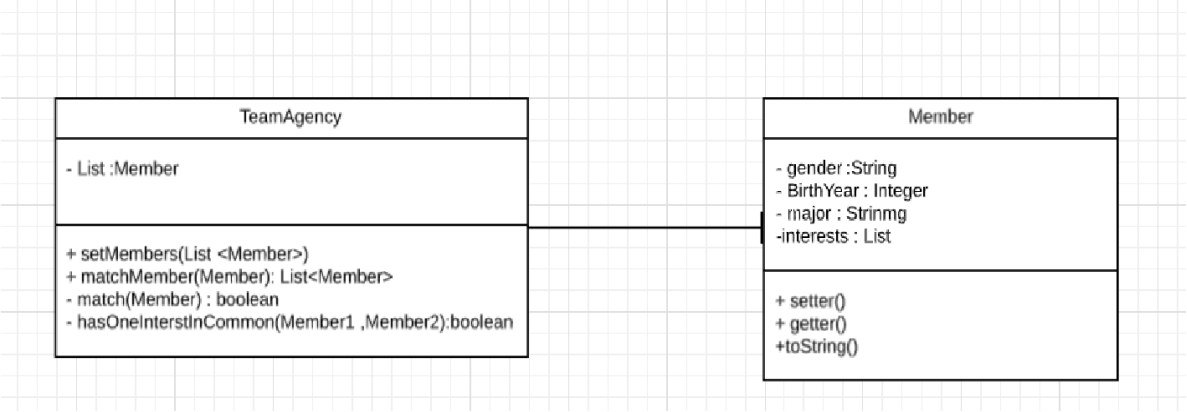
```
public List<Member> matchMember(Member member) {
    List<Member> res = new ArrayList<Member>();

    for (Member candidate : Members) {
        // A matching team partner needs to have the opposite gender
        if (match(candidate ,member))
        {
            res.add(candidate);

        }
    }
    return res;
}
```

\

Result :

- More readable code
- Less code duplication
- Isolates independent parts of code

After the refactor UML became :



Part 2: Testing

Choice	Input data set	Input property
1 true	A	No numbers
1 false	B	At least one number
2 true	B	Exactly one number
2 false	D	At least two number
3 true	D	The second number is greater than the first number
3 false	E	The second number is less than the first number
4 zero times	D	Exactly two number
4 one times	F	Exactly three number
4 more than one times	G	At least four number
5 true	H	The third number is greater than the largest value)ma1)
5 false	F	The third number is smaller than the largest value)ma1)
6 true	F	The third number is greater than the second largest value)ma2)

6 false	Z	The third number is smaller than the second largest value(ma2)

Generate a table of the input data sets based on the above table

Input data set	Contents	Expeected out put
A	(no number)	No numbers
B	{2}	2
D	{2 , 3}	3 2
E	{3 ,2 }	3 2
F	{3 ,2 ,1 }	3 2
H	{ 2, 3 ,4}	4 3
G	{5 ,2 ,3,12 }	12 5
z	{5, 4 ,1 }	5 4

erroneous code:

```
2
3   public class TestingTask {
4       static String result;
5       public static void main(String[] args) {
6           findTwoLargestNumbers(new String[] {"4","22","15","606"});
7       }
8
9   @   private static void findTwoLargestNumbers(String[] args) {
10
11          int ma1 = 0, ma2 = 0;
12          if (args.length == 0) {                          /* 1 */
13              System.out.println("No numbers");
14              result="No numbers";
15          }
16          else {
17              ma1 = Integer.parseInt(args[0]);
18              if (args.length == 1) {                      /* 2 */
19                  System.out.println("largest = " + ma1);
20                  result=""+ma1;
21              }
22              else {
23                  int obs = Integer.parseInt(args[1]);
24                  //ma2=obs;
25                  if (obs > ma1)                           /* 3 */
```

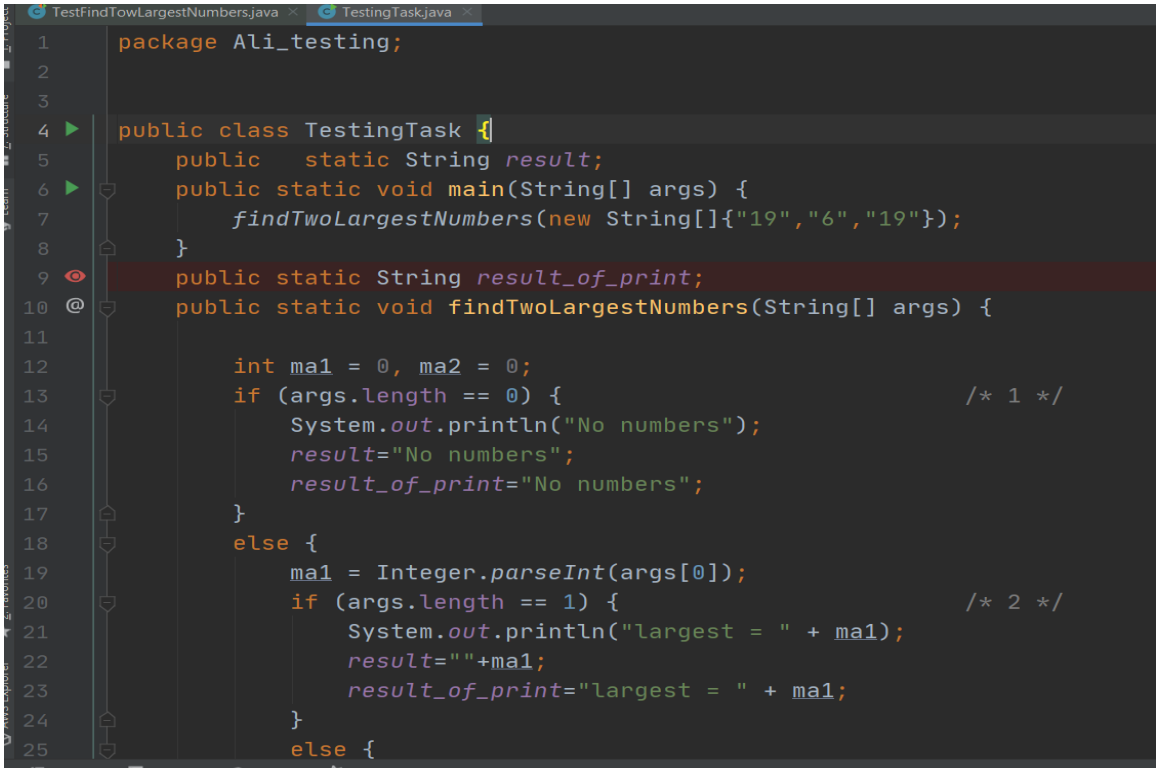
```
                if (obs > ma1)                                  /* 3 */
                {
                    ma2 = ma1;
                    ma1 = obs;
                }
                for (int i = 2; i < args.length; i++)           /* 4 */
                {
                    obs = Integer.parseInt(args[i]);
                    if (obs > ma1) /* 5 */
                    {
                        ma2 = ma1;
                        ma1 = obs;
                    } else if (obs > ma2) /* 6 */
                        ma2 = obs;
                }
                System.out.println("The two largest are " + ma1 + " and " + ma2);
                result="ma1,ma2";
            }
        }
    }

}

```

Test cases:

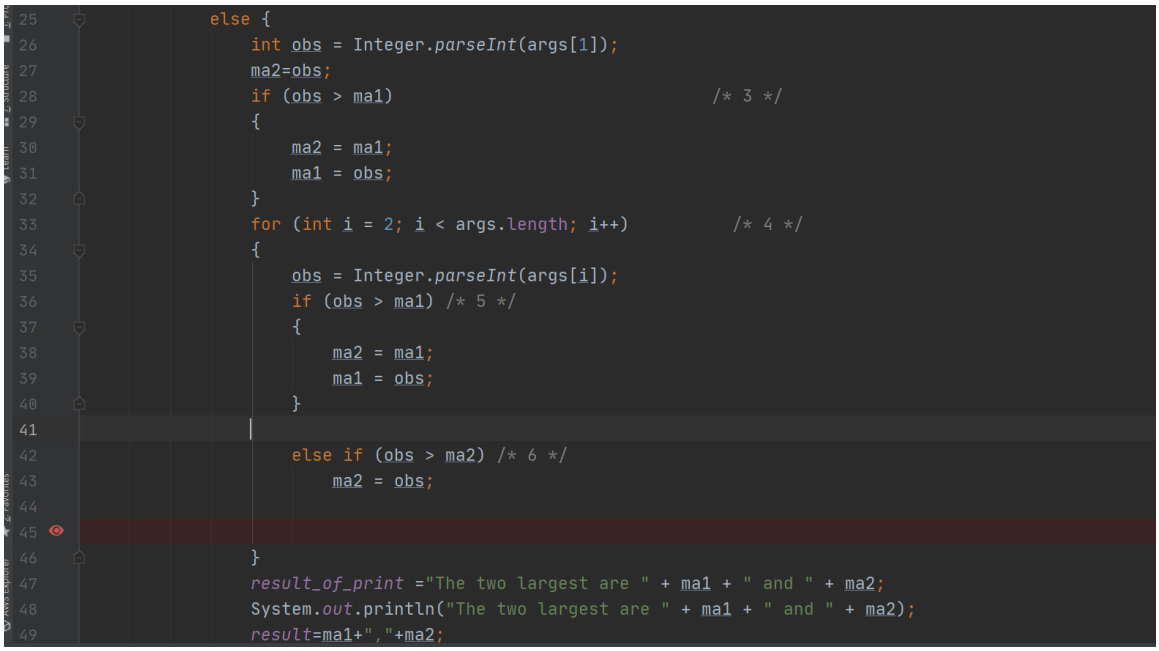
```
    @Test
    public void test_thirdNumber_greater_than_largest_value_ma1(){//h
        TestingTask.main(new String[] {"2","3","4"});
        String res=TestingTask.result;
        assertEquals(res, actual: "4,3");
        assertTrue(TestingTask.result_of_print.equals("The two largest are " + 4 + " and " + 3));
    }
    @Test
    public void test_hird_number_smaller_than_Second_largestValue_ma2(){//Z
        TestingTask.main(new String[] {"5","4","1"});
        String res=TestingTask.result;
        assertEquals(res, actual: "5,4");
        assertTrue(TestingTask.result_of_print.equals("The two largest are " + 5 + " and " + 4));

    }

    @Test
    public void test_At_least_four_number(){//Z
        TestingTask.main(new String[] {"5","2","3","12"});
        String res=TestingTask.result;
        assertEquals(res, actual: "12,5");
        assertTrue(TestingTask.result_of_print.equals("The two largest are " + 12 + " and " + 5));
    }
```

```
    @Test
    public void test_Second_lage_than_firstNumber_and_exactly_tow_number(){//D
        TestingTask.main(new String[] {"1","2"});
        String res=TestingTask.result;
        assertEquals(res, actual: "2,1");
        assertTrue(TestingTask.result_of_print.equals("The two largest are " + 2 + " and " + 1));
    }
    @Test
    public void test_SecongNumberless_than_firtNumberand_exactlyTow_number (){//E
        TestingTask.main(new String[] {"2","1"});
        String res=TestingTask.result;
        assertEquals(res, actual: "2,1");
        assertTrue(TestingTask.result_of_print.equals("The two largest are " + 2 + " and " + 1));
    }

    @Test
    public void test_third_number_is_smaller_than_largestValue_ExactlyThree_Number_thirdNumber_greater_than_the_second_large
        TestingTask.main(new String[] {"3","2","1"});
        String res=TestingTask.result;
        assertEquals(res, actual: "3,2");
        assertTrue(TestingTask.result_of_print.equals("The two largest are " + 3 + " and " + 2));
    }
```

Correct code:



```
TestFindTowLargestNumbers.java  TestingTask.java

1   package Ali_testing;
2
3
4   public class TestingTask {
5       public   static String result;
6       public static void main(String[] args) {
7           findTwoLargestNumbers(new String[]{"19","6","19"});
8       }
9       public static String result_of_print;
10      public static void findTwoLargestNumbers(String[] args) {
11
12          int ma1 = 0, ma2 = 0;
13          if (args.length == 0) {                              /* 1 */
14              System.out.println("No numbers");
15              result="No numbers";
16              result_of_print="No numbers";
17          }
18          else {
19              ma1 = Integer.parseInt(args[0]);
20              if (args.length == 1) {                          /* 2 */
21                  System.out.println("largest = " + ma1);
22                  result=""+ma1;
23                  result_of_print="largest = " + ma1;
24              }
25              else {
```



```
25              else {
26                  int obs = Integer.parseInt(args[1]);
27                  ma2=obs;
28                  if (obs > ma1)                               /* 3 */
29                  {
30                      ma2 = ma1;
31                      ma1 = obs;
32                  }
33                  for (int i = 2; i < args.length; i++)        /* 4 */
34                  {
35                      obs = Integer.parseInt(args[i]);
36                      if (obs > ma1) /* 5 */
37                      {
38                          ma2 = ma1;
39                          ma1 = obs;
40                      }
41
42                      else if (obs > ma2) /* 6 */
43                          ma2 = obs;
44
45
46                  }
47                  result_of_print ="The two largest are " + ma1 + " and " + ma2;
48                  System.out.println("The two largest are " + ma1 + " and " + ma2);
49                  result=ma1+","+ma2;
```

```
            else {
                int obs = Integer.parseInt(args[1]);
                ma2=obs;
                if (obs > ma1)                                  /* 3 */
                {
                    ma2 = ma1;
                    ma1 = obs;
                }
                for (int i = 2; i < args.length; i++)           /* 4 */
                {
                    obs = Integer.parseInt(args[i]);
                    if (obs > ma1) /* 5 */
                    {
                        ma2 = ma1;
                        ma1 = obs;
                    }
                    else if(ma1==obs)
                        ma1=obs;
                    else if (obs > ma2) /* 6 */
                        ma2 = obs;


                }
                result_of_print ="The two largest are " + ma1 + " and " + ma2;
                System.out.println("The two largest are " + ma1 + " and " + ma2);
```

```
                System.out.println("The two largest are " + ma1 + " and " + ma2);
                result=ma1+","+ma2;
            }
        }
    }

}
```

percentage of Coverage: