

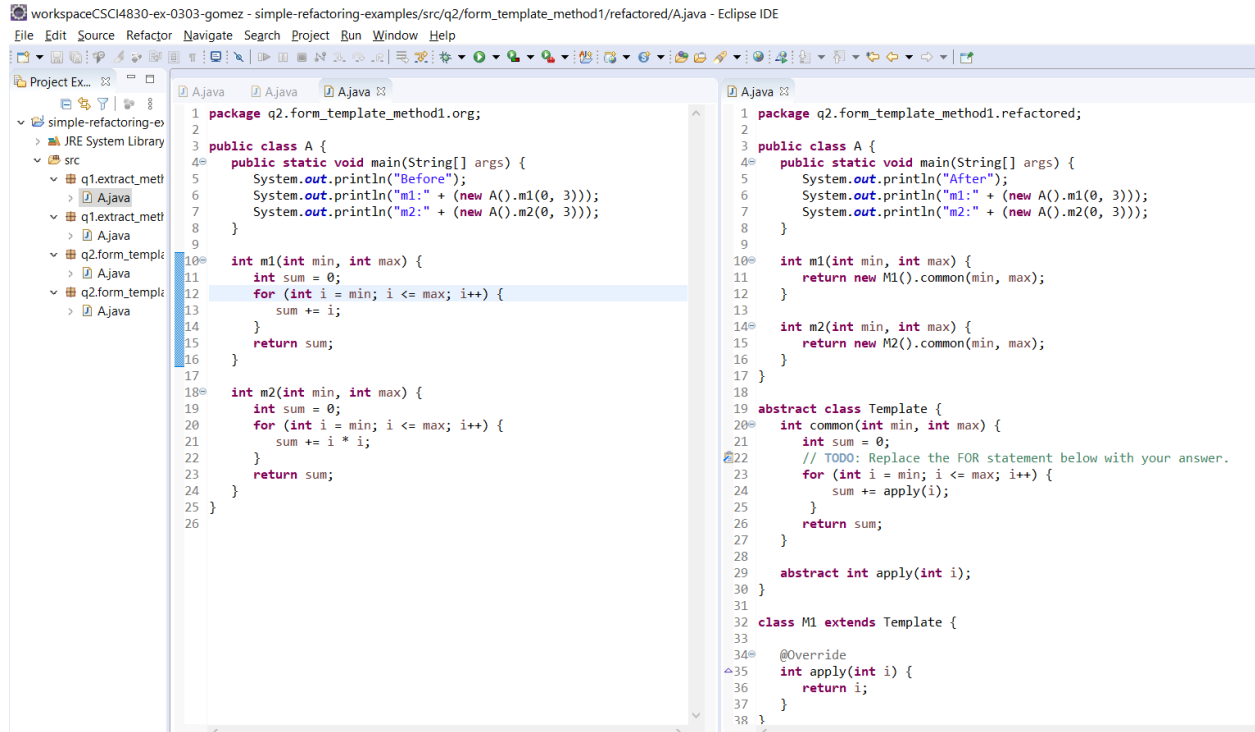
Emily Gomez

03/12-a

<https://github.com/gomezem/classwork/tree/master/workspaceCSCI4830-ex-0312a-gomez>

```
1 package q1.extract_method.refactored;
2
3 import java.util.List;
4
5 public class A {
6     Node m1(List<Node> nodes, String p) {
7         extractedMethod(nodes, p);
8         return null;
9     }
10
11     Edge m2(List<Edge> edgeList, String p) {
12         extractedMethod(edgeList, p);
13         return null;
14     }
15
16     <T extends Graph> void extractedMethod (List<T> objs, String p) {
17         for (T obj: objs) {
18             if (obj.contains(p))
19                 System.out.println(obj);
20         }
21     }
22 }
23
24 class Graph {
25     String name;
26     boolean contains(String p) {
27         return name.contains(p);
28     }
29 }
30
31 class Node extends Graph {
32     String name;
33
34     public boolean contains(String p) {
35         return name.contains(p);
36     }
37 }
38
39 class Edge extends Graph {
40     String name;
41     public boolean contains(String p) {
42         return name.contains(p);
43     }
44 }
```

Emily Gomez  
03/12-a



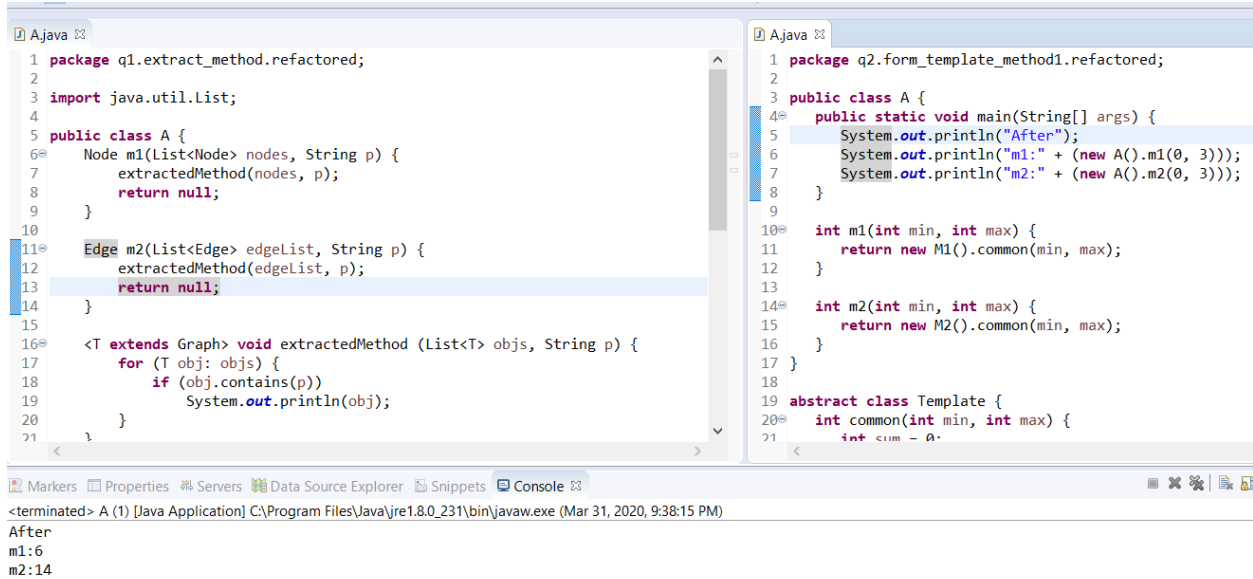
```
1 package q2.form_template_method1.org;
2
3 public class A {
4     public static void main(String[] args) {
5         System.out.println("Before");
6         System.out.println("m1:" + (new A().m1(0, 3)));
7         System.out.println("m2:" + (new A().m2(0, 3)));
8     }
9
10    int m1(int min, int max) {
11        int sum = 0;
12        for (int i = min; i <= max; i++) {
13            sum += i;
14        }
15        return sum;
16    }
17
18    int m2(int min, int max) {
19        int sum = 0;
20        for (int i = min; i <= max; i++) {
21            sum += i * i;
22        }
23        return sum;
24    }
25 }
26
```

```
1 package q2.form_template_method1.refactored;
2
3 public class A {
4     public static void main(String[] args) {
5         System.out.println("After");
6         System.out.println("m1:" + (new A().m1(0, 3)));
7         System.out.println("m2:" + (new A().m2(0, 3)));
8     }
9
10    int m1(int min, int max) {
11        return new M1().common(min, max);
12    }
13
14    int m2(int min, int max) {
15        return new M2().common(min, max);
16    }
17 }
18
19 abstract class Template {
20     int common(int min, int max) {
21         int sum = 0;
22         // TODO: Replace the FOR statement below with your answer.
23         for (int i = min; i <= max; i++) {
24             sum += apply(i);
25         }
26         return sum;
27     }
28     abstract int apply(int i);
29 }
30
31 class M1 extends Template {
32
33     @Override
34     int apply(int i) {
35         return i;
36     }
37 }
38
```

```
40 class M2 extends Template {
41
42     @Override
43     int apply(int i) {
44         return i * i;
45     }
46 }
47
```

Emily Gomez

03/12-a



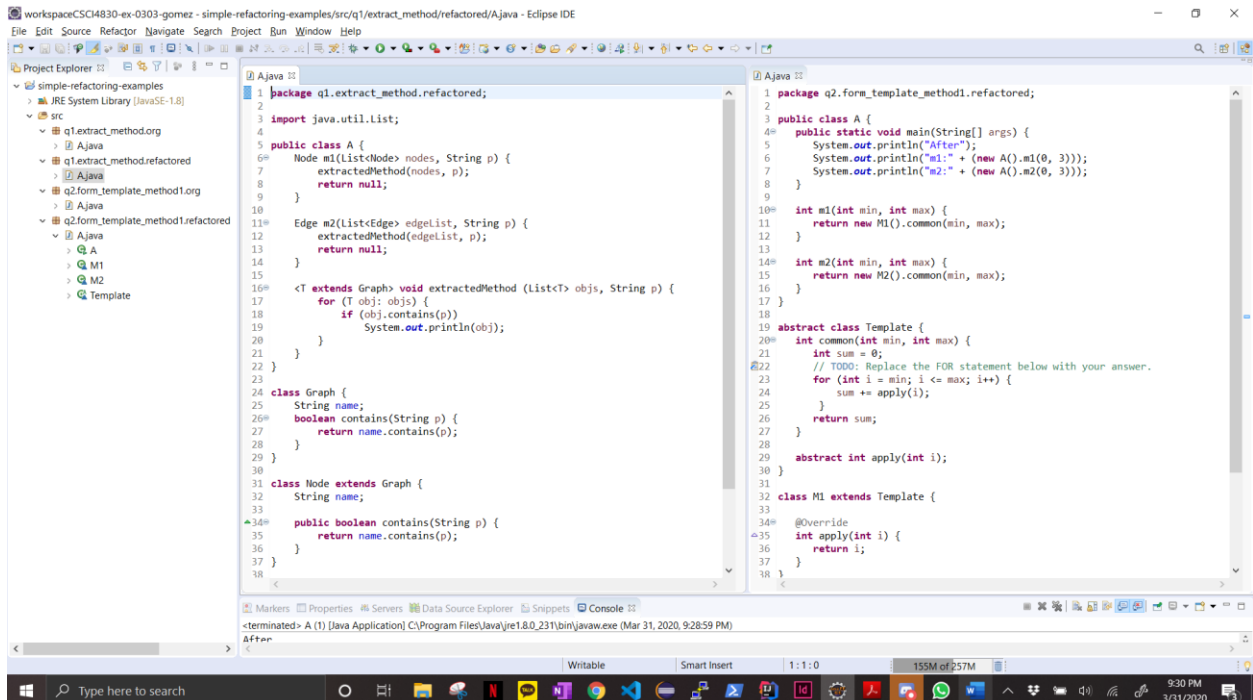
```
1 package q1.extract_method.refactored;
2
3 import java.util.List;
4
5 public class A {
6     Node m1(List<Node> nodes, String p) {
7         extractedMethod(nodes, p);
8         return null;
9     }
10
11     Edge m2(List<Edge> edgeList, String p) {
12         extractedMethod(edgeList, p);
13         return null;
14     }
15
16     <T extends Graph> void extractedMethod (List<T> objs, String p) {
17         for (T obj: objs) {
18             if (obj.contains(p))
19                 System.out.println(obj);
20         }
21     }
22 }
```

```
1 package q2.form_template_method1.refactored;
2
3 public class A {
4     public static void main(String[] args) {
5         System.out.println("After");
6         System.out.println("m1:" + (new A().m1(0, 3)));
7         System.out.println("m2:" + (new A().m2(0, 3)));
8     }
9
10    int m1(int min, int max) {
11        return new M1().common(min, max);
12    }
13
14    int m2(int min, int max) {
15        return new M2().common(min, max);
16    }
17 }
18
19 abstract class Template {
20     int common(int min, int max) {
21         int sum = 0;
```

<terminated> A (1) [Java Application] C:\Program Files\Java\jre1.8.0\_231\bin\javaw.exe (Mar 31, 2020, 9:38:15 PM)

After  
m1:6  
m2:14

Running after:



```
1 package q1.extract_method.refactored;
2
3 import java.util.List;
4
5 public class A {
6     Node m1(List<Node> nodes, String p) {
7         extractedMethod(nodes, p);
8         return null;
9     }
10
11     Edge m2(List<Edge> edgeList, String p) {
12         extractedMethod(edgeList, p);
13         return null;
14     }
15
16     <T extends Graph> void extractedMethod (List<T> objs, String p) {
17         for (T obj: objs) {
18             if (obj.contains(p))
19                 System.out.println(obj);
20         }
21     }
22 }
23
24 class Graph {
25     String name;
26     boolean contains(String p) {
27         return name.contains(p);
28     }
29 }
30
31 class Node extends Graph {
32     String name;
33
34     public boolean contains(String p) {
35         return name.contains(p);
36     }
37 }
38 }
```

```
1 package q2.form_template_method1.refactored;
2
3 public class A {
4     public static void main(String[] args) {
5         System.out.println("After");
6         System.out.println("m1:" + (new A().m1(0, 3)));
7         System.out.println("m2:" + (new A().m2(0, 3)));
8     }
9
10    int m1(int min, int max) {
11        return new M1().common(min, max);
12    }
13
14    int m2(int min, int max) {
15        return new M2().common(min, max);
16    }
17 }
18
19 abstract class Template {
20     int common(int min, int max) {
21         int sum = 0;
22         // TODO: Replace the FOR statement below with your answer.
23         for (int i = min; i <= max; i++) {
24             sum += apply(i);
25         }
26         return sum;
27     }
28
29     abstract int apply(int i);
30 }
31
32 class M1 extends Template {
33
34     @Override
35     int apply(int i) {
36         return i;
37     }
38 }
```

<terminated> A (1) [Java Application] C:\Program Files\Java\jre1.8.0\_231\bin\javaw.exe (Mar 31, 2020, 9:28:59 PM)

After