

```
1 package com;
2
3 public interface Chain {
4
5     public void setNextChain(Chain nextChain);
6     public void calculate(Numbers n);
7
8
9 }
10
```

```
1 package com;
2
3 public class Numbers {
4     private int number1;
5     private int number2;
6
7     private String calculationWanted;
8
9     public Numbers(int newNumber1, int newNumber2, String calcWanted){
10
11         number1 = newNumber1;
12         number2 = newNumber2;
13         calculationWanted = calcWanted;
14
15     }
16
17     public int getNumber1(){ return number1; }
18     public int getNumber2(){ return number2; }
19     public String getCalcWanted(){ return calculationWanted; }
20 }
21
```

```

1 package com;
2
3 public class AddNumbers implements Chain{
4
5     private Chain nextInChain;
6
7
8     public void setNextChain(Chain nextChain) {
9
10         nextInChain = nextChain;
11
12     }
13
14
15     public void calculate(Numbers request) {
16
17         if(request.getCalcWanted() == "add"){
18
19             System.out.print(request.getNumber1() + " + " + request.getNumber2() + " = "+
20                 (request.getNumber1()+request.getNumber2()));
21
22         } else {
23
24             nextInChain.calculate(request);
25
26         }
27
28     }
29 }
30

```

```

1 package com;
2
3 public class SubtractNumbers implements Chain {
4
5     private Chain nextInChain;
6
7     @Override
8     public void setNextChain(Chain nextChain) {
9
10         nextInChain = nextChain;
11
12     }
13
14     @Override
15     public void calculate(Numbers request) {
16
17         if(request.getCalcWanted() == "sub"){
18
19             System.out.print(request.getNumber1() + " - " + request.getNumber2() + " = "+
20                 (request.getNumber1()-request.getNumber2()));
21
22         } else {
23
24             nextInChain.calculate(request);
25
26         }
27
28     }
29 }
30 }
31

```

```

1 package com;
2
3 public class MultNumbers implements Chain{
4
5     private Chain nextInChain;
6
7     @Override
8     public void setNextChain(Chain nextChain) {
9
10         nextInChain = nextChain;
11     }
12
13     @Override
14     public void calculate(Numbers request) {
15
16         if(request.getCalcWanted() == "mult"){
17
18             System.out.print(request.getNumber1() + " * " + request.getNumber2() + " = " +
19                 (request.getNumber1()*request.getNumber2()));
20
21         } else {
22
23             nextInChain.calculate(request);
24
25         }
26     }
27
28 }
29
30
31
32 }

```

```

1 package com;
2
3 public class DivideNumbers implements Chain {
4
5     private Chain nextInChain;
6
7     @Override
8     public void setNextChain(Chain nextChain) {
9
10         nextInChain = nextChain;
11     }
12
13     @Override
14     public void calculate(Numbers request) {
15
16         if(request.getCalcWanted() == "div"){
17
18             System.out.print(request.getNumber1() + " / " + request.getNumber2() + " = " +
19                 (request.getNumber1()/request.getNumber2()));
20
21         } else {
22
23             System.out.print("Only works for add, sub, mult, and div");
24
25         }
26     }
27
28 }
29
30

```

```
1 package com;
2
3 public class TestCalcChain {
4
5     public static void main(String[] args){
6
7
8
9         Chain chainCalc1 = new AddNumbers();
10        Chain chainCalc2 = new SubtractNumbers();
11        Chain chainCalc3 = new MultNumbers();
12        Chain chainCalc4 = new DivideNumbers();
13
14
15
16        chainCalc1.setNextChain(chainCalc2);
17        chainCalc2.setNextChain(chainCalc3);
18        chainCalc3.setNextChain(chainCalc4);
19
20
21
22        Numbers request = new Numbers(4,2,"divgf");
23
24        chainCalc4.calculate(request);
25
26    }
27
28 }
```