```java
package com;

public interface Chain {

    public void setNextChain(Chain nextChain);
    public void calculate(Numbers n);


}

```

```java
package com;

public class Numbers {
    private int number1;
    private int number2;

    private String calculationWanted;

    public Numbers(int newNumber1, int newNumber2, String calcWanted){

        number1 = newNumber1;
        number2 = newNumber2;
        calculationWanted = calcWanted;

    }

    public int getNumber1(){ return number1; }
    public int getNumber2(){ return number2; }
    public String getCalcWanted(){ return calculationWanted; }
}

```

```java
package com;

public class AddNumbers implements Chain{

private  Chain nextInChain;


    public void setNextChain(Chain nextChain) {

        nextInChain = nextChain;

    }


    public void calculate(Numbers request) {

        if(request.getCalcWanted() == "add"){

            System.out.print(request.getNumber1() + " + " + request.getNumber2() + " = "+
                    (request.getNumber1()+request.getNumber2()));

        } else {

            nextInChain.calculate(request);

        }

    }
}

```

```java
package com;

public class SubtractNumbers implements Chain {

private  Chain nextInChain;

    @Override
    public void setNextChain(Chain nextChain) {

        nextInChain = nextChain;

    }

    @Override
    public void calculate(Numbers request) {

        if(request.getCalcWanted() == "sub"){

            System.out.print(request.getNumber1() + " - " + request.getNumber2() + " = "+
                    (request.getNumber1()-request.getNumber2()));

        } else {

            nextInChain.calculate(request);

        }

    }

}

```

```java
package com;

public class MultNumbers implements Chain{

    private  Chain nextInChain;

    @Override
    public void setNextChain(Chain nextChain) {

        nextInChain = nextChain;

    }

    @Override
    public void calculate(Numbers request) {

        if(request.getCalcWanted() == "mult"){

            System.out.print(request.getNumber1() + " * " + request.getNumber2() + " = "+
                    (request.getNumber1()*request.getNumber2()));

        } else {

            nextInChain.calculate(request);

        }

    }



}
```

```java
package com;

public class DivideNumbers implements Chain {

    private  Chain nextInChain;

    @Override
    public void setNextChain(Chain nextChain) {

        nextInChain = nextChain;

    }

    @Override
    public void calculate(Numbers request) {

        if(request.getCalcWanted() == "div"){

            System.out.print(request.getNumber1() + " / " + request.getNumber2() + " = "+
                    (request.getNumber1()/request.getNumber2()));

        } else {

            System.out.print("Only works for add, sub, mult, and div");

        }
    }

}

```

```java
package com;

public class TestCalcChain {

    public static void main(String[] args){



        Chain chainCalc1 = new AddNumbers();
        Chain chainCalc2 = new SubtractNumbers();
        Chain chainCalc3 = new MultNumbers();
        Chain chainCalc4 = new DivideNumbers();



        chainCalc1.setNextChain(chainCalc2);
        chainCalc2.setNextChain(chainCalc3);
        chainCalc3.setNextChain(chainCalc4);



        Numbers request = new Numbers(4,2,"divgf");

        chainCalc4.calculate(request);

    }

}
```