

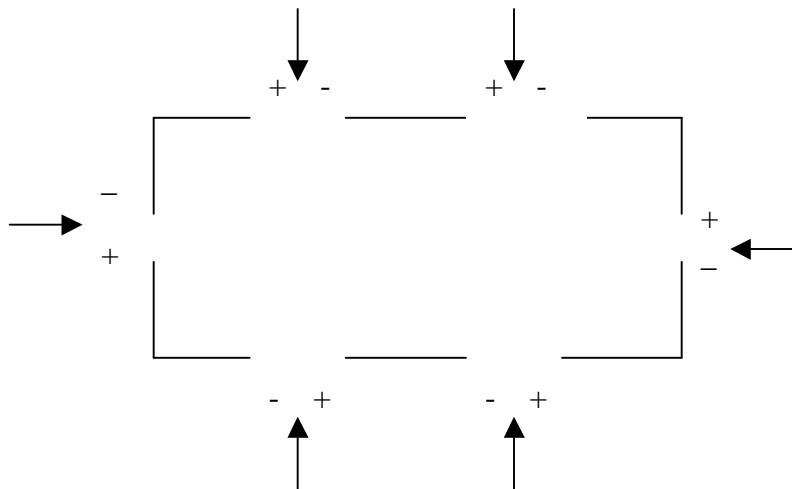
## HW 2

694T  
Spring 2000

A linear circuit has **five** slots, as shown below in the figure below (the slots are indicated by arrows). In each of these slots we can insert either a **battery** or a **resistance**. The voltage of a battery, and the value of a resistance are assumed to be **known** quantities. Make an **abstraction** of this circuit board and name it **CircuitBoard**. In this **CircuitBoard** you should be able to insert either a **battery** or a **resistor** in each one of the four slots. **CircuitBoard** should have an operation named **current** which should return the value of the current flowing in this circuit. Use the supplied classes, **LinearExp** and **SysLinearEq1by1**, to handle the linear equations in this problem. **Do not** use **if** or **switch-case** keywords anywhere in this problem. Of course, besides **CircuitBoard**, you will also need other abstractions in this problem.

Write a client class with a **main** method. In this method make a new **CircuitBoard** object, and insert **two batteries** and **three resistors** in it (you can choose the values of these circuit elements yourself). Invoke the **current** method on the **CircuitBoard** object to get the current in the circuit. **Repeat** the above step by inserting **one battery** and **four resistors** in the **CircuitBoard** object, and calculate the current again.

*Remember no if, or switch statements.*



**Note:** The classes, **LinearExp** and **SysLinearEq1by1**, along with the class **LinEqClient**, which demonstrates the use of these classes, is sitting in the package **LinearEq** under the **projects** directory of this class. The source code is also available in a PDF file on the web page for this class. These classes are **not** models of reusable design (actually just the opposite!). I have written them quickly to serve the purpose of **this problem** only, so you don't have to spend time writing code for classes to handle simple linear equations in one and two variables.

```

package linearEq;

public class LinEqClient {

    public static void main(String[] args) throws Solvelby1Exception{

        LinearExp exp1 = new LinearExp(2.0D, 3.0D, 0.0D); //exp1 = 2.0 +
3.0 x+0.0 y
        System.out.println(exp1);

        exp1.replaceByValue(1, 5.0D); // in exp1 replace x by value 5.0
        System.out.println(exp1); // exp1 = 17.0 + 0.0 x + 0.0 y

        LinearExp exp2 = new LinearExp(0.0D, 4.0D, 0.0D); // 0.0 + 4.0 x +
0.0 y
        LinearExp exp3 = exp1.add(exp2); // add exp2 to exp1
        System.out.println(exp3); // 17.0 + 4.0 x + 0.0 y

        SysLinearEq1by1 sys = new SysLinearEq1by1(exp3);
        double sol = sys.solve();
        System.out.println(sol); // sol: x = -17.0/4.0

        LinearExp exp4 = new LinearExp(2.0D, 0.0D, 4.0D); // exp4 = 2.0 +
0.0 x+4.0 y
        sys = new SysLinearEq1by1(exp4);
        sol = sys.solve();
        System.out.println(sol); // sol: y = -2.0/4.0

        LinearExp exp5 = new LinearExp(2.0D, 2.0D, 4.0D); // exp4 = 2.0 +
2.0 x+4.0 y
        sys = new SysLinearEq1by1(exp5);
        // sol = sys.solve(); throws exception

        LinearExp exp6 = new LinearExp(2.0D, 0.0D, 0.0D); // exp4 = 2.0 +
0.0 x+0.0 y
        sys = new SysLinearEq1by1(exp6);
        // sol = sys.solve(); throws exception
    }
}

```

---

```

package linearEq;

public final class SysLinearEq1by1 {
    private LinearExp exp = null;

    public SysLinearEq1by1(LinearExp exp) {
        this.exp = exp;
    }

    public double solve() throws Solvelby1Exception {

        if( exp == null ) {
            throw new Solvelby1Exception();
        }

        double[] coeff = exp.getCoeff();
    }
}

```

---

```

    if( coeff[1] != 0.0D && coeff[2] != 0.0D) {
        throw new Solvelby1Exception();
    }

    if( coeff[1] == 0.0D && coeff[2] == 0.0D) {
        throw new Solvelby1Exception();
    }

    if( coeff[1] == 0.0D ) {
        return -coeff[0]/coeff[2];
    }

    return -coeff[0]/coeff[1];
}
}

```

---

```

package linearEq;

public final class LinearExp {
    private double[] coeff = new double[3];

    public LinearExp(double d1, double d2, double d3) {
        coeff[0] = d1;
        coeff[1] = d2;
        coeff[2] = d3;
    }

    public double[] getCoeff() {
        return (double[]) coeff.clone();
    }

    public LinearExp add(LinearExp exp) {
        return new LinearExp(coeff[0]+exp.coeff[0],
                               coeff[1]+exp.coeff[1],
                               coeff[2]+exp.coeff[2]);
    }

    public LinearExp sub(LinearExp exp) {
        return new LinearExp(coeff[0]-exp.coeff[0],
                               coeff[1]-exp.coeff[1],
                               coeff[2]-exp.coeff[2]);
    }

    public void replaceByValue(int index, double value) {
        if(index == 1 || index == 2) {
            coeff[0] = coeff[0] + coeff[index]*value;
            coeff[index] = 0.0D;
        }
    }

    public String toString() {
        return "["+coeff[0]+", "+coeff[1]+", "+coeff[2]+"]";
    }
}

```

---

---

**package linearEq;**

public class Solvelby1Exception extends Exception {

    public Solvelby1Exception() {  
        super(" [Not enough info to solve Sys1by1]");  
    }

}

---