Decision Management Community Challenge Nov-2025

"Advanced Website Design"

https://dmcommunity.org/challenge/challenge-nov-2025/

This challenge is an advanced version of last month's Challenge Oct-2025, when a freelance developer needs to design a website with as many features as possible, but also maximizing the total value of the selected features. Now we have several additional requirements:

- Features 3 and 4 can be chosen only together.
- Feature 2 cannot be combined with Feature 3.
- Only one of the features 8, 9, and 10 can be selected. Otherwise, the cost of each of these features will be increased by 10%.
- If 5 or more features are selected, 5% discount is provided.

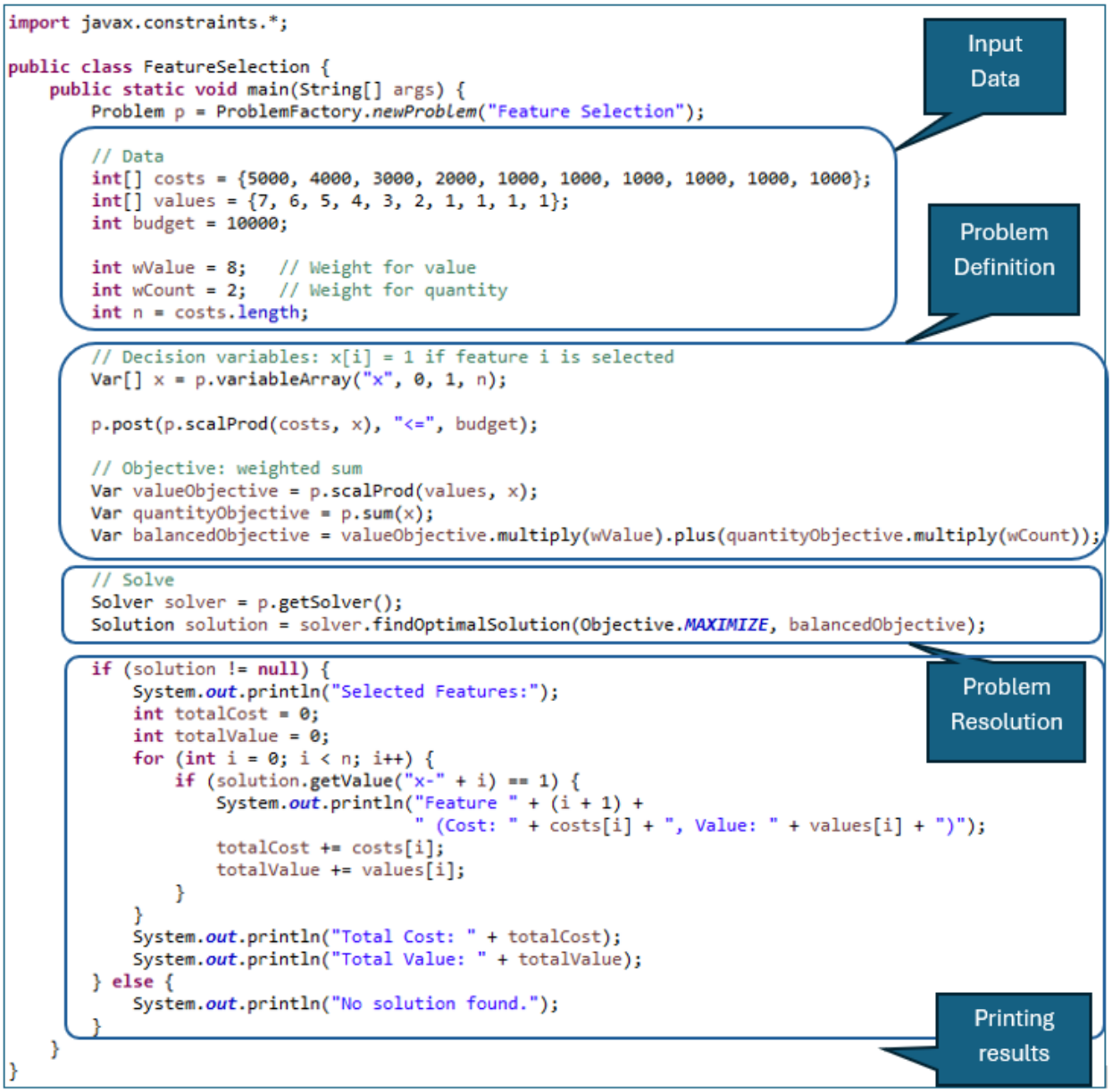
We still need to offer a design that maximizes the total value while minimizing the total cost.

This solution simply expands my JSR-331 solution by adding a few constraints to the problem definition:

Problem Definition

```
// Decision variables: x[i] = 1 if feature i is selected
Var[] x = p.variableArray("x", 0, 1, n);

// Budget constraint
p.post(p.scalProd(costs, x), "<=", budget);

// Features 3 and 4 can be chosen only together
p.post(x[2], "=", x[3]);

// Feature 2 cannot combine with Feature 3
p.post(x[1].plus(x[2]), "<=", 1);

// Only one of the features 8, 9, and 10 can be selected
p.post(x[7].plus(x[8]).plus(x[9]), "<=", 1);

// Objectives
Var valueObjective = p.scalProd(values, x);
Var quantityObjective = p.sum(x);
Var balancedObjective = valueObjective.multiply(wValue).plus(quantityObjective.multiply(wCount));
```

For simplicity of the comparison, here is the complete JSR-331 solution without additional constraints:




```java
import javax.constraints.*;

public class FeatureSelection {
    public static void main(String[] args) {
        Problem p = ProblemFactory.newProblem("Feature Selection");

        // Data
        int[] costs = {5000, 4000, 3000, 2000, 1000, 1000, 1000, 1000, 1000, 1000};
        int[] values = {7, 6, 5, 4, 3, 2, 1, 1, 1, 1};
        int budget = 10000;

        int wValue = 8;   // Weight for value
        int wCount = 2;   // Weight for quantity
        int n = costs.length;

        // Decision variables: x[i] = 1 if feature i is selected
        Var[] x = p.variableArray("x", 0, 1, n);

        p.post(p.scalProd(costs, x), "<=", budget);

        // Objective: weighted sum
        Var valueObjective = p.scalProd(values, x);
        Var quantityObjective = p.sum(x);
        Var balancedObjective = valueObjective.multiply(wValue).plus(quantityObjective.multiply(wCount));

        // Solve
        Solver solver = p.getSolver();
        Solution solution = solver.findOptimalSolution(Objective.MAXIMIZE, balancedObjective);

        if (solution != null) {
            System.out.println("Selected Features:");
            int totalCost = 0;
            int totalValue = 0;
            for (int i = 0; i < n; i++) {
                if (solution.getValue("x-" + i) == 1) {
                    System.out.println("Feature " + (i + 1) +
                                       " (Cost: " + costs[i] + ", Value: " + values[i] + ")");
                    totalCost += costs[i];
                    totalValue += values[i];
                }
            }
            System.out.println("Total Cost: " + totalCost);
            System.out.println("Total Value: " + totalValue);
        } else {
            System.out.println("No solution found.");
        }
    }
}
```

The requirement “If 5 or more features are selected, 5% discount is provided” could be easily added after the problem resolved with one line before printing the totalCost:

```
if (count > 5) totalCost *= 0.95;
```

To be fair, The above implementation does not cover this part of the third constraint: “*Otherwise, the cost of each of these features will be increased by 10%*”. This constraint could be in interest only if we want to minimize the total cost – you can find the proper implementation here.