

Decision Management Community's Challenge Feb-2025

"Mr Bates vs The Post Office"

<https://dmcommunity.org/challenge/challenge-feb-2025/>

A solution produced by Copilot using <http://JavaSolver.com>

I asked Copilot:

Use JavaSolver to solve the Challenge described at <https://dmcommunity.org/challenge/challenge-feb-2025/>

Copilot said:

I couldn't find the specific details of the February 2025 challenge on the Decision Management Community website. However, if you can provide the details of the challenge, I can help you solve it using JavaSolver.

Please share the problem statement or any specific requirements for the challenge, and I'll assist you in creating a solution using JavaSolver.

I said:

Here is the problem description. The Post Office in DS City has come up with a new measure to limit packages that can be sent under a new flat rate, regardless of weight or volume. The "strap measure" of a box is the length of the longest side plus two times the sum of the two shorter sides. For the new flat rate for sending a packet, the strap measure may not exceed 100 inches.

Your e-commerce client Mr. Bates sells large comforters of 9,000 cubic inches in volume and wants to make use of this new flat rate if possible. Can you design a box that will be large enough to hold these comforters and that can be sent at a flat rate price?

Copilot said:

Let's solve this optimization problem using JavaSolver. We'll define the variables representing the dimensions of the box and set up the constraints and objective function to find a suitable box that meets the requirements.

Problem Breakdown

1. **Variables:** Define the dimensions of the box LL , WW , and HH (length, width, height).

2. **Volume Constraint:** Ensure the box volume is at least 9,000 cubic inches ($L \times W \times H \geq 9000$).
3. **Strap Measure Constraint:** Ensure the strap measure does not exceed 100 inches ($L + 2(W + H) \leq 100$).
4. **Objective:** Find a box that meets these constraints.

JavaSolver Code Example

```
import javax.constraints.*;
```

```
public class BoxOptimization extends Problem {  
    public void define() {  
        // Create the problem  
        Problem p = ProblemFactory.newProblem("BoxOptimization");  
  
        // Define variables L, W, H with domains from 1 to 100  
        Var L = p.variable("L", 1, 100);  
        Var W = p.variable("W", 1, 100);  
        Var H = p.variable("H", 1, 100);  
  
        // Add volume constraint  $L * W * H \geq 9000$   
        p.post(L.multiply(W).multiply(H), ">=", 9000);  
  
        // Add strap measure constraint  $L + 2(W + H) \leq 100$   
        p.post(L.plus(W.multiply(2)).plus(H.multiply(2)), "<=", 100);  
  
        // Define the objective function (minimize strap measure)  
        Var strapMeasure = L.plus(W.multiply(2)).plus(H.multiply(2));  
        p.add("Objective", strapMeasure);  
        Solver solver = p.getSolver();  
        Solution solution = solver.findOptimalSolution(Objective.MINIMIZE, strapMeasure);  
    }  
}
```

```

// Print the solution
if (solution != null) {
    solution.log();
    System.out.println("Optimal Box Dimensions:");
    System.out.println("Length (L): " + solution.getValue("L"));
    System.out.println("Width (W): " + solution.getValue("W"));
    System.out.println("Height (H): " + solution.getValue("H"));
} else {
    System.out.println("No solution found.");
}
}

public static void main(String[] args) {
    BoxOptimization problem = new BoxOptimization();
    problem.define();
}
}

```

Explanation

1. **Define Variables:** We define the dimensions LL , WW , and HH with domains from 1 to 100.
2. **Add Constraints:**
 - **Volume Constraint:** $L \times W \times H \geq 9000$
 - **Strap Measure Constraint:** $L + 2(W + H) \leq 100$
3. **Objective Function:** We aim to minimize the strap measure $L + 2(W + H)$.
4. **Solve the Problem:** We use the solver to find the optimal solution that minimizes the strap measure while satisfying the constraints.

This code will help you find the optimal box dimensions that can hold the comforters and be sent at the flat rate price.

As I already had JavaSolver installed on my machine, I decided to run this code “as is”. I only needed to remove unnecessary “extends Problem {” after class BoxOptimization. When I ran this program, I received the following results:

JSR-331 "Constraint Programming API" Release 2.3.1

JSR-331 Implementation based on Constrainer 5.4.1

Optimal solution is found. Objective: 100

Solution #1:

L[28] W[17] H[19] Objective[100]

Optimal Box Dimensions:

Length (L): 28

Width (W): 17

Height (H): 19

Still for the sake of other people, **I asked Copilot:**

How do I run this JavaSolver program on my machine?

Copilot:

To run the JavaSolver program on your machine, follow these steps:

Prerequisites

1. **Java Development Kit (JDK):** Ensure you have JDK installed on your machine. You can download it from the Oracle website or use OpenJDK.
2. **JavaSolver Library:** Download the JavaSolver library from the JavaSolver GitHub repository.

Steps to Run the Program

1. **Set Up Your Project:**
 - Create a new directory for your project.
 - Inside this directory, create a subdirectory named lib and place the JavaSolver JAR file in it.
2. **Create the Java Program:**

- Create a new Java file named BoxOptimization.java and paste the provided code into it.

3. Compile the Program:

- Open a terminal or command prompt.
- Navigate to the directory where your BoxOptimization.java file is located.
- Compile the Java program using the following command:
- `javac -cp lib/javasolver.jar BoxOptimization.java`

4. Run the Program:

- Run the compiled Java program using the following command:
- `java -cp .:lib/javasolver.jar BoxOptimization`
- Note: On Windows, use a semicolon (;) instead of a colon (:) to separate the classpath entries:
- `java -cp .;lib/javasolver.jar BoxOptimization`

Example Directory Structure

your-project-directory/

├── BoxOptimization.java

└── lib/

 └── javasolver.jar

Example Commands

Navigate to your project directory

`cd your-project-directory`

Compile the Java program

`javac -cp lib/javasolver.jar BoxOptimization.java`

Run the Java program

```
java -cp ./lib/javasolver.jar BoxOptimization
```

This will compile and run the JavaSolver program, and you should see the output with the optimal box dimensions that meet the given constraints.

If you encounter any issues or have further questions, feel free to ask!