Is programming in DMN better than in Java?

Alex Goldin

In 2016 I submitted a simple Java solution for the DMCommunity.org Challenge "Rebooking Passengers" to show that trying to do programming in DMN is much more awkward to compare with a general purpose programming language. Now I looked at a DMN-based solution for the Challenge "Map Coloring" and again was surprised that people still try to use DMN for procedural programming. I believe DMN is the decision modeling standard but not a programming language. Just look at the DMN solution that includes a recursive function with formal parameters and several intermediate lists. Even a programmer will have a tough time to understand how it works.

I understand when people define and solve this problem using declarative tools such as CP Optimizer or Java Solver. They only define a problem and allow a standard engine to solve it. There are no needs to write a solving algorithm. But if someone wants to use a procedural approach and to write such an algorithm, would not it be easier to use any commonly known programming language such as Python or Java? For example, a procedural solution can be presented as two embedded loops:

```
for(Country country : countries) {
  for(String color : colors) {
       if (!isColorUsedByNeighbourOf(color,country)) {
              country.color = color;
              System.out.println(country.name + " = " + country.color);
              break;
       }
  }
}
```

At least it is not difficult to understand and does not require any additional tools to execute. The entire working code in Java is on the next page. When you execute this code, it produces the expected solution:

```
Belgium = red
Denmark = red
France = green
Germany = blue
Netherlands = green
Luxembourg = yellow
```

```java
public class MapColoringJava {

    static String[] colors = { "red", "green", "blue", "yellow" };
    static String[] countryNames = {"Belgium","Denmark","France","Germany","Netherlands","Luxembourg"};

    class Country {
        String name;
        String color;
        ArrayList<String> neighbours = new ArrayList<String>();

        public Country(String name, String ...neighbours) {
            this.name = name;
            this.color = null;
            for(String neighbour : neighbours) {
                this.neighbours.add(neighbour);
            }
        }

        public boolean isNeighbour(Country country) {
            for(int i = 0; i < neighbours.size(); i++) {
                if (neighbours.get(i).equals(country.name))
                    return true;
            }
            return false;
        }
    }

    Country[] countries = new Country[countryNames.length];

    public void define() {
        countries[0] = new Country("Belgium", "France","Netherlands","Germany");
        countries[1]  = new Country("Denmark", "Germany");
        countries[2]  = new Country("France", "Belgium","Luxembourg","Germany");
        countries[3]  = new Country("Germany", "France","Belgium","Luxembourg","Denmark","Netherlands");
        countries[4]  = new Country("Netherlands", "Belgium","Germany");
        countries[5]  = new Country("Luxembourg", "France","Belgium","Germany");
    }

    public void solve() {

        for(Country country : countries) {
            for(String color : colors) {
                if (!isColorUsedByNeighbourOf(color,country)) {
                    country.color = color;
                    System.out.println(country.name + " = " + country.color);
                    break;
                }
            }
        }
    }

    public boolean isColorUsedByNeighbourOf(String color, Country country) {
        for(Country otherCountry : countries) {
            if (country == otherCountry) continue;
            if (country.isNeighbour(otherCountry) && color.equals(otherCountry.color) ) {
                return true;
            }
        }
        return false;
    }

    public static void main(String[] args) {
        MapColoringJava map = new MapColoringJava();
        map.define();
        map.solve();
    }
}
```