TopcoderChallenger User Manual

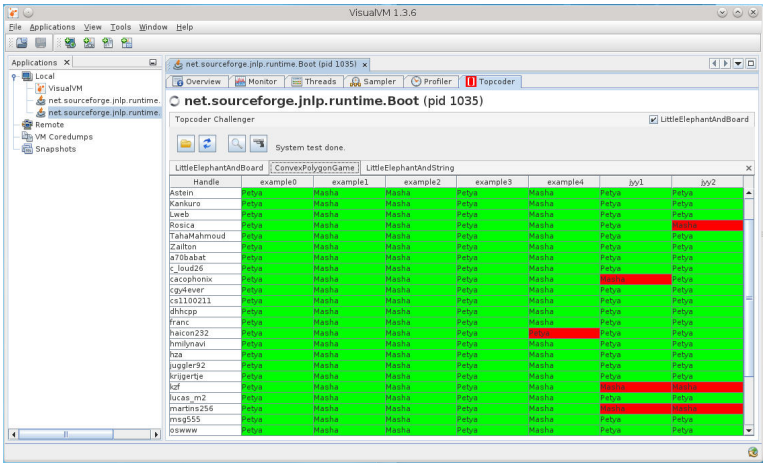
Yanyan Jiang <jiangyy@outlook.com>

November 25, 2013

1 Overview

TopcoderChallenger (we use TCC for short) is a cheating program designed for challenge phase of algorithm Topcoder contests.

TCC works as follows: user firstly specify contest meta-data such as class names, entry method names, entry method signatures and several test cases. Then each time one opens up a challenge window, he/she could use the heap dump functionality provided by VisualVM to obtain the coder's actual code. At any time, a mini "system test" could be performed to run each program against every test case. The system test results will be displayed for your reference. Below is a screenshot taken from codes crawled from SRM597 practice room with two manual test cases.



In this demo, we tested all example test cases and two extra manual corner test cases which is easily for coders to ignore. The system test result finds several wrong submissions in the practice room. It also surprisingly finds one coder's submission even cannot pass all the examples. All submissions marked red are being successfully challenged.

2 Installation

2.1 Prerequisites

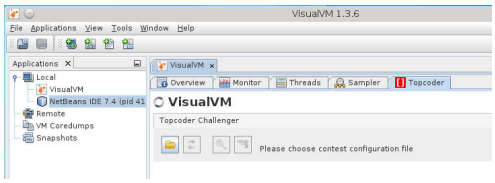
To compete in the Topcoder Algorithm contests, Java Runtime and Topcoder Contest Arena are required.

This cheat program runs both in Linux and in Windows environment as a plugin for Oracle's VisualVM. For Linux users, make sure `/usr/bin/timeout` exists[1]. For Windows users, Cygwin is required and Cygwin commands must be available via `PATH` environment variable. For both users, g++, GNU make and timeout are required to run the system test.

2.2 Installation

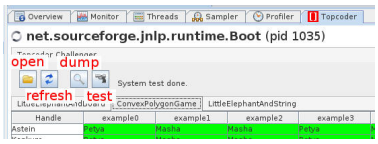
In Oracle's VisualVM, Click Tools → Plugins → Downloaded → Add Plugins and choose the TCC plugin file. Follow the instructions to make sure the plugin is correctly installed.

After a successful installation, double click on any running Java instance in VisualVM. A new tab named "Topcoder" will appear as the screenshot below.



3 Play with TopcoderChallenger

TCC interface is consisted of four buttons: open, dump, refresh and test. Their functions are briefly described below and further explained in the following subsections.



- **Open**: load in a contest configuration file. One cannot proceed with other actions unless a valid contest configuration file is loaded.
- **Refresh**: refresh a contest configuration file. This is used when a configuration file is updated.

[1] Using absolute path is for compatibility with Windows clients.

- **Dump**: make a heap dump of the current JVM opened in VisualVM. Then the heap dump is analyzed to find source code opened in a challenge window. For each found source code, it will be saved in the contest configuration file's directory.
- **Test**: run system test against dumped codes. System test results will be displayed immediately below the buttons.

Tip text next to the buttons would be helpful finding what is going on.

3.1 Writing Contest Configuration Files

Contest configuration file should be written in YAML format. You should describe each problem separately with its class name, and each problem must contain the following fields:

method. Method entry must be provided.

signature. Signature is described by a string in shape `[type of arg0], [type of arg1], ... [type of last arg] -> [return type]`. Basic type `int`, `long long`, `double`, `string` are indicated by `I`, `L`, `D`, `S`, respectively. Vectors are indicated by `V`. For instance, signature of `vector<int> f(vector <vector<string>>, double)` is indicated by `VVS,D->VI`.

tests. Tests consists of several test cases, which each one is described as a key-value pair. The key is the name of that test case and the value is the detailed description of that test case. You may use any meaningful names, but you should avoid spaces and special characters because the test case name will be part of a file name. For how to write test inputs and outputs, please refer to the example below showing a configuration file of `ConvexPolygonGame` in SRM597. Of course, you can describe multiple problems in a single contest configuration file.

```
ConvexPolygonGame:
  method: winner
  signature: |
    VI, VI -> S
  tests:
    example0: |
      [0, 1, 0], [0, 0, 1]->"Petya"
    example1: |
      [0, 4, 2], [0, 0, 2]->"Masha"
    example2: |
      [0, 100, 100, 0],[0, 0, 100, 100]->"Masha"
    example3: |
      [0, 50, 100, 50],[0, -1, 0, 1]->"Petya"
    example4: |
      [-100000,100000,100000,-100000],[-1,-1,1,1]->"Masha"
    jyy1: |
      [1,5,4,3],[1,4,5,4]->"Petya"
    jyy2: |
      [0,3,5,2],[0,2,5,3]->"Petya"
```