Assignment 1: HTTP Number-Guessing Game

***Due date:** Tuesday, November 7, 2017 at 22:00*

This is an individual assignment. You may discuss it with others, but your code and documentation must be written on your own. You might receive a bonus for an outstanding solution.

You must implement an automated player for a simple on-line game. The game runs within a game server accessible through HTTP. The objective of the game is to guess a number chosen by the server. Thus, the game consists of two components: an HTTP server, which is accessible online at the address research.inf.usi.ch:9999, and an HTTP client that implements a robot player. You must implement the robot client.

To start the game, the client must first register itself as a player. The client registration opens a game session for which the server chooses an integer value between -2000000000 and 2000000000 (± 2 billion), and then allows the player to submit its guesses. Each guess by a client must refer to a valid session. The server responds to each guess by indicating whether the client has guessed the number exactly or whether the number is lower or higher. A play session remains valid for some time and at most for 30 guesses. The goal of the client is to guess the exact value before the play session expires.

The following table specifies the interface of the game server. Details of the requests and responses, including response codes and their semantics are consistent with the HTTP specification. Also consistently with HTTP, sessions are implemented using cookies. (See RFC 2616 for details at https://tools.ietf.org/html/rfc2616.)

URL	method	request body	response	
`/new-player`	`GET`	none	`204`	New game session created successfully.
			`503`	The server is busy. Try again later.
`/check-number`	`POST`	number	`200`	Valid guess. The body is of type *text/plain*, and can be either '<', '=', '>' (ASCII codes 60, 61, 62), indicating that the guess was low, exact, or high, respectively.
			`403`	The guess is not within a valid game session.

Implement your client player in Java in a class called `GuessNumber`, or in Python in a program with the same name. The client player must take the host and port number of the server as command-line parameters. The client player must also trace its interactions with the server by printing a summary of each request and the corresponding reply on the console (standard output). The client must continue playing the game until it successfully guesses a number, repeatedly registering for a new game session when the current session expires. Below is a sample execution of the client player.

```
$ java GuessNumber research.inf.usi.ch 9999
  request[GET]:
  reply[204]: Welcome to the game!
  request[POST]: 10000
  reply[200]: too high!
  request[POST]: 100
  reply[200]: too low!
  request[POST]: 500
  ...
```

Use *only* the basic library functions for strings and for network sockets. You may not use any other web-specific library. Submit your solution through *iCorsi* in a single archive file. Include a *README* file with a list of all the material you used from other sources and a list of all known limitations and bugs of your solution.