CSci 364
Spring, 2025
Programming Assignment #3

Design and implement volunteer computing application (client and server) using Java RMI and serializable objects.

https://en.wikipedia.org/wiki/Volunteer_computing

In volunteer computing clients perform work on behalf of a server. Individuals can volunteer their computer to perform work tasks on behalf of a scientific or educational project.

https://csgrid.org/csg/
https://en.wikipedia.org/wiki/SETI@home

Server:
The server shall make a remote object available to clients. This remote object must be in package “server” and implement the interface “api.ClientManager”. Using this object, clients can register, request work tasks, submit task results, and check their score. The server shall bind its remote object using the string “ClientManager”.

The remote object shall make available different types of work tasks to volunteers. When the client requests a work task, the remote object shall instantiate a worker object of that type (see below) with unique input values and return it to the client.

The remote object shall offer four work tasks. Each task shall be part of package “server” and shall extend the abstract class “api.Worker”. A task-specific accessor method must be written to retrieve the results of the doWork() method so that the server can retrieve the results of the client’s work.

SortWorker
- Sort a sequence of values. Include accessor methods to retrieve the sorted values.
- This worker has been implemented for you.
- The remote object shall initialize each instance of SortWorker with a random number of values between 1 and 5.

SumReducer
- Sum a sequence of numerical values (int, float, or double).
- The remote object shall initialize each instance of SumReducer with a random number of values between 1 and 5. The values to be summed shall be random integers between 1 and 1000.

PrimeChecker
- Tests whether a positive integer is prime or not.

FractionReducer
- Given a numerator and denominator, calculate the greatest common factor. Divide the input fields by the greatest common factor. Your class should have two accessor methods to retrieve the reduced numerator and reduced denominator.

Given input numerator of 12 and input denominator of 9, the reduced numerator should be 4 and the reduced denominator should be 3.

https://www.mathsisfun.com/greatest-common-factor.html

The remote object will need to maintain a number of internal data structures. The following are suggestions. A list of the work task names. An association of user names and their running scores. An association of task Ids and task names. An association of task Ids and task input objects. An association of task Ids and task outputs.

Each time a client registers, the remote object shall print all the client names and their scores to the server console window.

Each time a client submits results, the remote object shall print all inputs and results for all instances of that type of task.

The server program can have 0, 1, or 2 command line arguments. The default RMI host is "localhost", and the default RMI port is 1099. If args.length is 0, the default RMI host and port shall be used. If args.length is 1, the first argument is the RMI host. If args.length is 2, the second argument is the RMI port.

Client:
The client program is the volunteer that does work on behalf of the server/remote object. The client program does the following.

- Lookup server's ClientManager object at an RMI registry running on a host.
- Register with remote object. Get back a list of work task names from the remote object. Print the list of task names in the client console window.
- Randomly select one of the tasks, print the name of the task, and request from the remote object a work task for the selected project.
- After the work task is completed (after the client calls the doWork() method on the work task object), client shall submit the worker task object back to the remote object.
- Finally, the client shall request its running score and print the score in the console window.

The client shall register once and then request and perform work and print its running score five (5) times before exiting. The server/remote object shall remain running and ready for another client to request work.

The client shall use 1, 2, or 3 command line arguments: the client userId, the RMI host, and the RMI port, respectively. The first command line argument, the client userId, is required. The default value for the second argument, RMI host, is "localhost". The default value for the third argument, RMI port, is (int) 1099. If args.length is 1, the default values for RMI host and port will be used for the second and third arguments. If args.length is 2, the second argument is the host (String) of the remote object. If args.length is 3, the third argument is the RMI port (int).

Submit your source code and Ant script using the directory structure below. Your Ant script should have targets to clean and compile your source code.

```
hw3/
        app.policy
        build.xml
```

src/api
    [Java source files in package]
src/client
    [Java source files in package]
src/server
    [Java source files in package]

Bonus Points:

- Restrict access to classes and fields appropriately. Don't make everything public. The client will not need access to anything in the server package. The server will not need access to anything in the client package. Items in the api package may be accessed by both the client and server.
- Write a one-page report describing what you would do to ensure thread safety for the remote object and its internal data structures. What are the critical sections? Explain whether you would need monitors, semaphores, or locks and how you would use it/them. Include the report in PDF format in the same directory as the build.xml file.