Assignment 1 (Damith)

- Due 10 Aug by 18:00
- Points 100

Assignment 1 - Introduction to Java RMI

Marks

The marks awarded for this assignment are worth 7% of the total mark for DS.
IMPORTANT If your code does not compile and run the awarded mark is 0.

Handin

Use the Websubmission system. For details on how to submit this exercise read through the steps of this assignment.

Procedure for this Assignment

Step 0: Getting to know Subversion

Subversion, also known as svn, is a powerful version control system to help maintain a coherent copy of a project that can be worked on from multiple locations. We will use Subversion as the handin mechanism throughout this course. Click here to learn about the features of interest to us.

Step 1: Creating the assignment directory in your svn repository

Current versions of the svn client program.
Open a terminal window on your machine, and cut-and-paste the following command:
svn mkdir --parents -m "assignments"
https://version-control.adelaide.edu.au/svn/aXXXXXX/YYYY/s2/ds/assignment1
This command will create an empty directory named YYYY/s2/ds/assignment1 in an svn repository that we have set up for you on the machine version-control.adelaide.edu.au. Your svn repository is called https://version-control.adelaide.edu.au/svn/aXXXXXXX/
Note: Replace XXXXXXX with your student ID and YYYY with the four digits representing the year we are in.
Old versions of the svn client program.
NOTE Older versions of the svn client, (pre 1.5), do not support the --parents option. If you are using an older client, the following explicit steps must be used to create the YYYY/s2/ds/assignment1 directory in your svn repository. If a directory already exists and you attempt to recreate it, you will receive an error message complaining about MKCOL.
Create the YYYY directory if it does not exist.
svn mkdir -m "assignment 0" https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY
Create the YYYY/s2 directory if it does not exist.

svn mkdir -m "assignment 0"
https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2
Create the YYYY/s2/ds directory if it does not exist.
svn mkdir -m "assignment 0"
https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds
Create the YYYY/s2/ds/assignment1 directory.
svn mkdir -m "assignment 0"
https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds/assignment1

Step 2: Checking out a working version of your assignment.

In your home account, change to a directory above the one where you want to place your working files for this assignment. Then type:
svn checkout
https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds/assignment1
ds-YY-s2-assign1
A new directory called ds-YY-s2-assign1 will now be created in your current working directory. The contents of this now become a working-copy of the files for all assignments for this offering of Distributed Systems. Note that this working copy contains hidden sub-directories needed by svn to operate correctly.
Note that you can have more than one working copy of a project. For example you could have one working copy at home and one at university and you can use svn to help keep them in sync. See the svn documentation for details on how this can be done. However, for now, we will assume you have just the one working copy.

Step 3: Saving your files in your repository

The assignments require you to edit and create files. If you create a new file or directory that is part of your assignment you must add it to the list of files managed by svn. You can add a file, Test.java, to the svn repository by typing:
svn add Test.java
Note that the directory containing the new file must already have been added to svn.
As you work on your assignment, you should commit your changes early and often. That is, you should regularly ask svn to copy changed files in your working copy to your repository. To commit your files to the repository you change to the directory containing the working copy of your files and type:
svn commit -m "meaningful message about what I just changed."
Note, that the message after the -m in the line above can be anything. However, it should be relevant to the state of the project at this moment. Also note that running commit is the only way your repository will get to know about the latest versions of your files. When we mark your submission we will look at what has been committed to your repository and, also, how often and what times you commited to your repository.
There are two major benefits to be gained by maintaining an up to date copy of your assignment in the svn repository. Firstly, svn can act as a backup mechanism. If you accidently lose a file you can retrieve it from the svn repository. Secondly, when you post questions on the forums the lecturer(s) can access your svn repository and give you more helpful answers.

Step 4: Assignment Submission

The next step is to submit your assignment using the Computer Science Web Submission System.

By now you have written your assignment, committed your changed files to the repository along the way and thoroughly tested them. The Web Submission System will not mark your assignment but will perform some limited checks using the current versions of your files in your repository. When you attempt to make a new submission you will be presented with an assessment cover sheet. This includes a link to your repository so you can check what files you are submitting. The cover sheet also includes a declaration that you are submitting your own work.

Note that, if the testing highlights some errors then you can always edit your files, commit them and make a new submission using the Web Submission System.

Assignment Description

Objective

To gain an understanding of how remote method invocation in Java works, and to develop a working example of a Java RMI system. This will be essential in developing future applications as you will quickly learn all of the pitfalls inherent in developing standards-based clients and multi-threaded servers.

To gain an understanding of how a distributed system works, this first assignment phase involves developing a simple Java RMI application. This will involve developing both the client and server side of a distributed application: a simple reverse Polish calculator server. We will be testing your client against our servers and your servers with our clients - make sure that you adhere to the standards we define!

The calculator server operates a stack and clients push values and operations on to the stack. While, usually, each client should have its own stack, you may use a single stack on the server. You may also assume that operations are always carried out in a sensible order: that is, we will only push an operator after pushing two operands and we will only pop when there is a value on the stack. You may also assume that the operator provided will only be one of the four displayed types and that you may use integer arithmetic for all calculations. Finally, you are not required to handle the case of division by zero.

Following the directions discussed in lectures, you should create a Java RMI Server that supports the following remote methods:

- void pushValue(int operand)
- This method will take the value of operand and push it on to the top of the calculator stack.
- void pushOperator(String operator)
- This method will push a String containing an operator ("+","-","*","/") to the stack, which will cause the server to pop the two operands already on the stack, apply the operation and then push the result back on to the stack.
- int pop()
- This method will pop the top of the calculator stack and return it to the client.
- boolean isEmpty()
- This method will return true if the stack is empty, false otherwise.

- int delayPop(int millis)
- This method will wait millis milliseconds before carrying out the pop operation as above.

Importantly: Your implementation should use the following files:

- Calculator.java - the interface that defines the remote operations implemented by your remote service.
- CalculatorImplementation.java - the implementation class for the remote operations.
- CalculatorServer.java - the server class.
- CalculatorClient.java - a test client that should connect to the server, and test its operation.

You will need to create and add these files to your SVN repository.
Don't forget to commit your work frequently and to submit before the due date!

Assessment

Your assignment will be marked out of 100 points, as following:

- 50 points for the functionality of your code. (see checklist below)

Checklist: The functionality of your assignment 1 refers to:
o All clients access the same stack on the server
o pushValue works – one client, many clients (more than 3)
o pushOperator works – one client, many clients (more than 3)
o pop works – one client, many clients (more than 3)
o delayPop works – one client, many clients (more than 3)
Bonus (extra 10 points)

Clients have their own stack on the server: each client accesses their own stack, rather than the common one

- 35 points for the quality of your automated testing, both in testing the server with single and multiple clients.
- 15 points for the quality of your code. (see checklist below)

Code Quality Checklist

Do!

write comments above the header of each of your methods, describing
what the method is doing, what are its inputs and expected outputs
o describe in the comments any special cases
o create modular code, following cohesion and coupling principles

Don’t!

use magic numbers
o use comments as structural elements (see video)
o mis-spell your comments
o use incomprehensible variable names
o have long methods (not more than 80 lines)
o allow TODO blocks