Alan Kaminsky • Department of Computer Science • Rochester Institute of Technology • 4492 + 2361 = 6853 Home Page

Concepts of Parallel and Distributed Systems • CSCI 251-02, -03 • Fall Semester 2015 Course Page

CSCI 251-02, -03—Concepts of Parallel and Distributed Systems Programming Project 1

Prof. Alan Kaminsky—Fall Semester 2015
Rochester Institute of Technology—Department of Computer Science

---

Overview

Write a Java program that uses multiple threads to learn about thread creation, execution, synchronization, and termination.

Help with your project: I am willing to help you with the design of your project. I am willing to help you debug your project if the code isn't working. However, for help with design or debugging issues *you must come see me in person.* Either visit me during office hours or make an appointment. I will not help you with design or debugging issues via email. If it's the evening of the project deadline and I have gone home, you are on your own. Plan and work ahead so there will be plenty of time for me to help you if necessary.

Generalized Pascal's Triangle

The generalized Pascal's triangle is a series of integers arranged in N rows. The first row contains one integer, the second row contains two integers, . . . the N-th row contains N integers. The generalized Pascal's triangle is specified by three parameters A, B, and S, which are integers greater than or equal to zero. Let $P_{r,c}$ be the triangle entry in row r and column c, $1 \le r \le N$, $1 \le c \le r$. The triangle entries are computed using these rules:

- $P_{1,1} = S$
- $P_{r,1} = A + P_{r-1,1}$
- $P_{r,r} = P_{r-1,c-1} + B$

- $P_{r,c} = P_{r-1,c-1} + P_{r-1,c}$

For example, the generalized Pascal's triangle with $N = 10$, $A = 0$, $B = 0$, and $S = 1$ is

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1

This is what is usually called just "Pascal's Triangle."

The generalized Pascal's triangle with $N = 10$, $A = 2$, $B = 3$, and $S = 4$ is

4
6 7
8 13 10
10 21 23 13
12 31 44 36 16
14 43 75 80 52 19
16 57 118 155 132 71 22
18 73 175 273 287 203 93 25
20 91 248 448 560 490 296 118 28
22 111 339 696 1008 1050 786 414 146 31

You will write a multithreaded Java program to calculate and print the generalized Pascal's triangle specified by the parameters *N*, *A*, *B*, and *S*. The program gets the parameters from the command line. The program prints the integers of the triangle in rows, with one space character between each integer and the next. There is no space character before the first integer or after the last integer in the row. Here is an example:

```
$ java GenPasTri 10 2 3 4
4
6 7
8 13 10
10 21 23 13
12 31 44 36 16
14 43 75 80 52 19
16 57 118 155 132 71 22
18 73 175 273 287 203 93 25
20 91 248 448 560 490 296 118 28
22 111 339 696 1008 1050 786 414 146 31
```

Software Requirements

1. The program must be run by typing this command line:

```
java GenPasTri <N> <A> <B> <S>
```

a. The first argument is the parameter *N*.
b. The second argument is the parameter *A*.
c. The third argument is the parameter *B*.
d. The fourth argument is the parameter *S*.
e. Each parameter must be an integer greater than or equal to zero.

Note: This means that the program's class must be named `GenPasTri`, and this class must not be in a package.

2. If the command line does not have the required number of arguments, or if any argument is erroneous, the program must print a usage message on the console and must exit. The usage message must describe the problem. The wording of the usage message is up to you.

3. The program must print on the console the generalized Pascal's triangle specified by the parameters.
    1. Each line of output must contain one row of the triangle.
    2. There must be one space character between each integer and the next.
    3. There must not be any space before the first integer.
    4. There must not be any space after the last integer.
    5. Each line of output must be terminated by a newline.

4. If the program encounters any error condition, the program must print an error message on the console and must exit. The error message must describe the problem. The wording of the error message is up to you.

   Note: If your program's output does not conform ***exactly*** *to Software Requirements 1 through 4,* ***you will lose credit*** *on your project.* See the Grading Criteria below.

Software Design Criteria

1. The program must consist of the following objects.

    a. A monitor object containing the generalized Pascal's triangle. The monitor class must have the following methods, and no others.

        - Constructor.
        - `int rows()`. This method returns the number of rows in the triangle, *N*.
        - `void putValue (int r, int c, int value)`. This method puts the given `value` into the monitor at row `r`, column `c`. Rows are indexed from 1 to *N*. Columns are indexed from 1 to `r`. It is an error if a value has already been put at row `r`, column `c`.
        - `int getValue (int r, int c)`. This method returns the value stored at row `r`, column `c`. This method does not return until the value at row `r`, column `c` has been put.

    b. A thread that prints the output.

    c. A thread that computes *one and only one* triangle element and stores it in the monitor object.

There is a separate instance of this thread *for each triangle element.*

d. A main program. The main program must create the above objects. The main program must start the threads *in this order:* first the output printing thread, then the thread that computes the last element in the last row, then the thread that computes the next-to-last element in the last row, and so on backwards through the elements of the row and the rows of the triangle. The main program must *not* wait for the threads to terminate. The main program *must not do anything else.*

2. The program must follow the thread programming patterns studied in class.
3. The program must be designed using object oriented design principles as appropriate.
4. The program must make use of reusable software components as appropriate.
5. Each class or interface must include a Javadoc comment describing the overall class or interface.
6. Each method within each class or interface must include a Javadoc comment describing the overall method, the arguments if any, the return value if any, and the exceptions thrown if any.

   Note: See my Java source files which we studied in class for the style of Javadoc comments I'm looking for.

   Note: If your program's design does not conform to Software Design Criteria 1 through 6, ***you will lose credit*** *on your project.* See the Grading Criteria below.

Submission Requirements

Your project submission will consist of a ZIP file containing the Java source file for every class and interface in your project. Put all the source files into a ZIP file named `"<username>.zip"`, replacing `<username>` with the user name from your Computer Science Department account. On a Linux system, the command for creating a ZIP file is:

```
zip <username>.zip *.java
```

Send your ZIP file to me by email at ark@cs.rit.edu. Include your full name and your computer account name in the email message, and include the ZIP file as an attachment.

When I get your email message, I will extract the contents of your ZIP file into a directory. I will set my Java class path to include the directory where I extracted your files, as well as the Parallel Java 2 Library. I will compile all the Java source files in your program using the JDK 1.8 compiler. I will then send you a reply message acknowledging I received your project and stating whether I was able to compile all the source files. If you have not received a reply within one business day (i.e., not counting weekends), please contact me. Your project is not successfully submitted until I have sent you an acknowledgment stating I was able to compile all the source files.

The submission deadline is Friday, September 18, 2015 at 11:59pm. The date/time at which your email message arrives in my inbox will determine whether your project meets the deadline.

You may submit your project multiple times up until the deadline. I will keep and grade only the most recent successful submission. There is no penalty for multiple submissions.

If you submit your project before the deadline, but I do not accept it (e.g. I can't compile all the source files), and you cannot or do not submit your project again before the deadline, the project will be late (see below). ***I strongly advise you to submit the project several days BEFORE the deadline, so there will be time to deal with any problems that may arise in the submission process.***

Grading Criteria

I will grade your project by:

- (10 points) Evaluating the design of your program, as documented in the Javadoc and as implemented in the source code.
  - All of the Software Design Criteria are fully met: 10 points.
  - Some of the Software Design Criteria are not fully met: 0 points.

- (20 points) Running your project. There will be twenty test cases, each worth 1 point. For each test case, if the program runs using the command line in Requirement 1 and the program produces the correct output, the test case will get 1 point, otherwise the test case will get 0 points. "Correct output" means "output fulfills *all* the Software Requirements *exactly*."

- (30 points) Total.

When I run your program, the Java class path will point to the directory with your compiled class files, as well as the Parallel Java 2 Library. I will use JDK 1.8 to run your program.

I will grade the test cases based *solely* on whether your program produces the correct output as specified in the above Software Requirements. *Any* deviation from the requirements will result in a grade of 0 for the test case. This includes errors in the formatting (such as extra spaces), incorrect uppercase/lowercase, output lines not terminated with a newline, and extraneous output not called for in the requirements. The requirements state *exactly* what the output is supposed to be, and there is no excuse for outputting anything different. If any requirement is unclear, please ask for clarification.

If there is a defect in your program and that same defect causes multiple test cases to fail, I will deduct points for *every* failing test case. The number of points deducted does *not* depend on the size of the defect; I will deduct the same number of points whether the defect is 1 line, 10 lines, 100 lines, or whatever.

After grading your project I will put your grade and any comments I have in your encrypted grade file. For further information, see the Course Grading and Policies and the Encrypted Grades.

Late Projects

If I have not received a successful submission of your project by the deadline, your project will be late and will receive a grade of zero. You may request an extension for the project. There is no penalty for an extension. See the Course Policies for my policy on extensions.

Plagiarism

Programming Project 1 must be entirely your own individual work. I will not tolerate plagiarism. If in my judgment the project is not entirely your own work, you will automatically receive, as a minimum, a grade of zero for the assignment. See the Course Policies for my policy on plagiarism.

Resubmission

If you so choose, you may submit a revised version of your project after you have received the grade for the original version. However, if the original project was not successfully submitted by the (possibly extended) deadline or was not entirely your own work (i.e., plagiarized), you are not allowed to submit a revised version. Submit the revised version via email in the same way as the original version. I will accept a resubmission up until three days after the grades for the original version are released. The resubmission deadline will be announced on the What's New page and in the Course Schedule. You may resubmit your project multiple times up until the deadline; I will keep and grade only the most recent successful resubmission; there is no penalty for multiple resubmissions. I will grade the revised version using the same criteria as the original version, then I will subtract 3 points (10% of the maximum possible points) as a resubmission penalty. The revised grade will replace the original grade, even if the revised grade is less than the original grade.