CO518 (2020-21)

Assessment - 2

Sanjay Bhattacherjee

November 23, 2020

Submission deadline: 23:59 hrs on Friday, 11th December, 2020 (Week 11).

What do you submit? You will have to submit **one zip file** following the specified convention below.

- Your zip file should be named as: <your ID>.zip. For example, if your ID is ab123@kent.ac.uk, the name of your file should be "ab123.zip".
- Your zip file should contain two files - each corresponding to one task. These files should each be named after the respective task as below:

  "ab123_task1.java" and
  "ab123_task2.java".

Where to submit? Using the link on Moodle.

General Instructions:

- You have to complete two programming tasks as described in this sheet. Each program should be written using the Java programming language in a single separate file as described above, so that it can be directly executed using *command-line arguments*. So, each of these files *should have a*

  ```
  public static void main (String[] args)
  ```

  function where the array `args` will carry the command-line arguments provided by the user.
- Each program should provide an input-output interface in the precise format as detailed under **Sample I/O** respectively. If you do not follow the format, marks may be deducted. The blanks denoted as ______ in the **Sample I/O** will be substituted with actual values while running the program.
- You may use pre-written programming libraries and classes like `Stack<>`, `ArrayList<>`, `FileInputStream`, `FileWriter`, and so on.

Marking criteria: The total marks for this assessment has been distributed among the two tasks. The mark allotment for each task has been indicated at the beginning of the task description. You will be marked based on the following criteria.

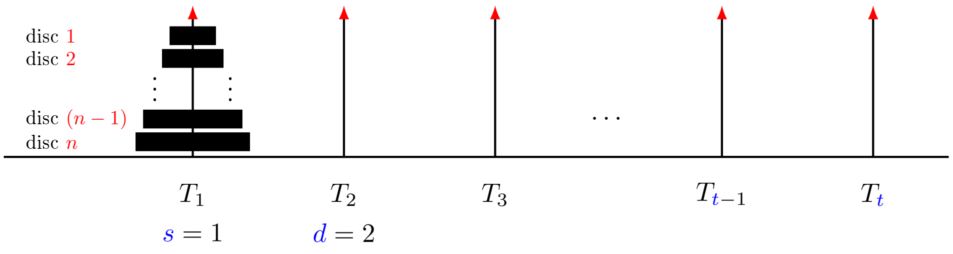
- Syntactical correctness: If your code encounters compilation errors you will be marked with zero for that task.
- Functional correctness: If your code works correctly for all input values it is tested with, you will qualify to get 0.8 fraction of the mark allotted to that task. Correctness will be judged based on your adherence with the **Sample I/O** format specified above as well as the correctness of the computations done by the program. You will receive partial marks if you have not met all the requirements of the task.
- Code quality: To qualify for full marks in a task, your code should have sufficient *inline comments* and its functionality explained at all major steps. Approximately one comment every 3 lines of code would be considered sufficient.

Task 1: Generalised Tower of Hanoi. Marks: 50%

The generalised tower of Hanoi problem is identified by the following four parameters

1. the number n (≥ 1) of discs,
2. the number t (≥ 3) of towers,
3. the source tower s, and
4. the destination tower d.

The towers (also called pegs) are labelled with $1, 2, \ldots, t$ and the discs are labelled with $1, 2, \ldots, n$. If $i < j$, then disc i is smaller in radius/size than j. In other words, the discs are labelled in ascending/increasing order of their sizes. An *instance* of the problem is identified by the 4-tuple (n, t, s, d).



Write a program that will take as input an instance (n, t, s, d) of the generalised tower of Hanoi problem and provide as output *a sequence of disc moves between towers.* The disc moves should be such that the following will always hold.

> *For two discs labelled i and j stacked at the same tower, if $i < j$, then disc i will be above disc j in the stack.*

The program should be executable via command line. Following is an example of command-line execution of the program `ab123_task1` with the arguments $(n = 6, t = 5, s = 1, d = 2)$.

```
java ab123_task1 6 5 1 2
```

The output should be provided in two forms *simultaneously.*

1. **On screen:** The output should be printed on the screen in a verbose form.

   Output on screen for $(n = 6, t = 5, s = 1, d = 2)$**:**

```
Move disk 1 from T1 to T4
Move disk 2 from T1 to T2
Move disk 3 from T1 to T3
Move disk 2 from T2 to T3
Move disk 1 from T4 to T3
Move disk 4 from T1 to T5
Move disk 5 from T1 to T4
Move disk 6 from T1 to T2
Move disk 5 from T4 to T2
Move disk 4 from T5 to T2
Move disk 1 from T3 to T4
Move disk 2 from T3 to T1
Move disk 3 from T3 to T2
Move disk 2 from T1 to T2
Move disk 1 from T4 to T2
```

2. **In a file:** The output should also be written to a file in a compact form. The output file *should follow the following format precisely* so that it can be input to a program (written by Sanjay for task 2) to check it for correctness.

   (a) The output should be arranged in rows / lines containing integers separated by a `space` or `tab`.

   (b) The first row should have the following four integers

   `number_of_discs` `number_of_towers` `source_tower` `destination_tower`

   (c) Every subsequent row in the file should have three integers

   `disc_number` `source_tower` `destination_tower`

   together representing **a single move**.

   The output file should be named as follows:

   `<your_ID>_ToH_n<n>_t<t>_s<s>_d<d>.txt`

The output file (named `sb2213_ToH_n6_t5_s1_d2.txt`) corresponding to the above on-screen output has been provided on Moodle (at this link). The content is as below. The gap between two numbers in a row of the file is a "tab" character.

Output file `sb2213_ToH_n6_t5_s1_d2.txt`**:**

```
6 5 1 2
1 1 4
2 1 2
3 1 3
2 2 3
1 4 3
4 1 5
5 1 4
6 1 2
5 4 2
4 5 2
1 3 4
2 3 1
3 3 2
2 1 2
1 4 2
```

A dummy program for this task has been provided on Moodle at this link. Please note that this code is for instructive purposes only. If you choose to use any part of this code, please include all necessary checks on inputs and test the code extensively for errors.

Task 2: Correctness Checker for the Generalised Tower of Hanoi. Marks: 50%
Write a program that will take as input a file containing a sequence of moves for an instance (n, t, s, d) of the generalised tower of Hanoi problem. You may assume that the input file is in a format as mentioned above for task 1. The program should be executable via command line. Following is an example of command-line execution of the program `ab123_task2` with the file name being provided as command line input.

```
java ab123_task3 ab123_ToH_n6_t5_s1_d2.txt
```

The program will first read the values (n, t, s, d) from the first line/row of the input file to identify the problem instance. Then it will read all the moves sequentially and check if they are all valid. The program should check that at least the following conditions are true.

1. If `disc_number` being moved is indeed at the top of `source_tower`.
2. If `disc_number` being moved is smaller than the disc on top of `destination_tower`.
3. If *after executing all the moves* given in the input file, all towers except `destination_tower` are empty.

Sample output for `sb2213_ToH_n6_t5_s1_d2.txt`**:**

```
The status of all the towers at the start is as follows:
Tower 1:  [6, 5, 4, 3, 2, 1]
Tower 2:  []
Tower 3:  []
Tower 4:  []
Tower 5:  []

Move:  disc 1 from tower 1 to tower 4
Before the move:
Source tower 1:  [6, 5, 4, 3, 2, 1]
Destination tower 4:  []
After the move:
Source tower 1:  [6, 5, 4, 3, 2]
Destination tower 4:  [1]

Move:  disc 2 from tower 1 to tower 2
Before the move:
Source tower 1:  [6, 5, 4, 3, 2]
Destination tower 2:  []
After the move:
Source tower 1:  [6, 5, 4, 3]
Destination tower 2:  [2]

Move:  disc 3 from tower 1 to tower 3
Before the move:
Source tower 1:  [6, 5, 4, 3]
Destination tower 3:  []
After the move:
Source tower 1:  [6, 5, 4]
Destination tower 3:  [3]

Move:  disc 2 from tower 2 to tower 3
Before the move:
Source tower 2:  [2]
Destination tower 3:  [3]
After the move:
```

```
Source tower 2:  []
Destination tower 3:  [3, 2]

Move:  disc 1 from tower 4 to tower 3
Before the move:
Source tower 4:  [1]
Destination tower 3:  [3, 2]
After the move:
Source tower 4:  []
Destination tower 3:  [3, 2, 1]

Move:  disc 4 from tower 1 to tower 5
Before the move:
Source tower 1:  [6, 5, 4]
Destination tower 5:  []
After the move:
Source tower 1:  [6, 5]
Destination tower 5:  [4]

Move:  disc 5 from tower 1 to tower 4
Before the move:
Source tower 1:  [6, 5]
Destination tower 4:  []
After the move:
Source tower 1:  [6]
Destination tower 4:  [5]

Move:  disc 6 from tower 1 to tower 2
Before the move:
Source tower 1:  [6]
Destination tower 2:  []
After the move:
Source tower 1:  []
Destination tower 2:  [6]

Move:  disc 5 from tower 4 to tower 2
Before the move:
Source tower 4:  [5]
Destination tower 2:  [6]
After the move:
Source tower 4:  []
Destination tower 2:  [6, 5]

Move:  disc 4 from tower 5 to tower 2
Before the move:
Source tower 5:  [4]
Destination tower 2:  [6, 5]
After the move:
Source tower 5:  []
Destination tower 2:  [6, 5, 4]

Move:  disc 1 from tower 3 to tower 4
Before the move:
Source tower 3:  [3, 2, 1]
Destination tower 4:  []
After the move:
Source tower 3:  [3, 2]
Destination tower 4:  [1]

Move:  disc 2 from tower 3 to tower 1
Before the move:
Source tower 3:  [3, 2]
Destination tower 1:  []
After the move:
Source tower 3:  [3]
Destination tower 1:  [2]

Move:  disc 3 from tower 3 to tower 2
Before the move:
Source tower 3:  [3]
Destination tower 2:  [6, 5, 4]
After the move:
Source tower 3:  []
Destination tower 2:  [6, 5, 4, 3]

Move:  disc 2 from tower 1 to tower 2
Before the move:
Source tower 1:  [2]
Destination tower 2:  [6, 5, 4, 3]
After the move:
Source tower 1:  []
Destination tower 2:  [6, 5, 4, 3, 2]

Move:  disc 1 from tower 4 to tower 2
Before the move:
Source tower 4:  [1]
Destination tower 2:  [6, 5, 4, 3, 2]
After the move:
Source tower 4:  []
Destination tower 2:  [6, 5, 4, 3, 2, 1]
```

```
The status of all the towers at the end of the sequence of moves is as follows:
Tower 1:  []
Tower 2:  [6, 5, 4, 3, 2, 1]
Tower 3:  []
Tower 4:  []
Tower 5:  []

The sequence of moves is correct.
```

In another file `sb2213_ToH_n6_t5_s1_d2_error.txt` also provided on Moodle (at this link), one of the moves of the file `sb2213_ToH_n6_t5_s1_d2.txt` has been manually changed to an erroneous value (such that the destination tower has a smaller disc on top). On running the program for task 2 with this erroneous input file, the output should be as follows:

Sample output for `sb2213_ToH_n6_t5_s1_d2_error.txt`**:**

```
The status of all the towers at the start is as follows:
Tower 1:  [6, 5, 4, 3, 2, 1]
Tower 2:  []
Tower 3:  []
Tower 4:  []
Tower 5:  []

Move:  disc 1 from tower 1 to tower 4
Before the move:
Source tower 1:  [6, 5, 4, 3, 2, 1]
Destination tower 4:  []
After the move:
Source tower 1:  [6, 5, 4, 3, 2]
Destination tower 4:  [1]

Move:  disc 2 from tower 1 to tower 2
Before the move:
Source tower 1:  [6, 5, 4, 3, 2]
Destination tower 2:  []
After the move:
Source tower 1:  [6, 5, 4, 3]
Destination tower 2:  [2]

Move:  disc 3 from tower 1 to tower 3
Before the move:
Source tower 1:  [6, 5, 4, 3]
Destination tower 3:  []
After the move:
Source tower 1:  [6, 5, 4]
Destination tower 3:  [3]

Move:  disc 2 from tower 2 to tower 3
Before the move:
Source tower 2:  [2]
Destination tower 3:  [3]
After the move:
Source tower 2:  []
Destination tower 3:  [3, 2]

Move:  disc 1 from tower 4 to tower 3
Before the move:
Source tower 4:  [1]
Destination tower 3:  [3, 2]
After the move:
Source tower 4:  []
Destination tower 3:  [3, 2, 1]

Move:  disc 4 from tower 1 to tower 3
Before the move:
Source tower 1:  [6, 5, 4]
Destination tower 3:  [3, 2, 1]
Move Error:  Destination tower:  [3, 2, 1] has a smaller disc than 4 on the top

The status of all the towers is as follows:
Tower 1:  [6, 5, 4]
Tower 2:  []
Tower 3:  [3, 2, 1]
Tower 4:  []
Tower 5:  []

The sequence of moves is incorrect.
```

A dummy program for this task has been provided on Moodle at this link. Please note that this code is for instructive purposes only. If you choose to use any part of this code, please include all necessary checks on inputs and test the code extensively for errors.

Released on: Monday, 23rd November, 2020 (Week 9)

Queries to: Sanjay Bhattacherjee (`s.bhattacherjee@kent.ac.uk`)