

1. Overview

Write a program to move the stack of blocks from **source** to specific **targets** based on the size of blocks. Movement of Robot arms, picking and dropping can be controlled using the **Robot** methods described below.

The methods of Robot class:

up(), **down()**, **extend()**, **contract()**, **raise()**, **lower()**, **pick()**, **drop()**, **speedUp()**, **slowDown()**

□ The height (**h**) of the main arm can be changed by 1 unit using methods **up()** and **down()**. The height (**h**) of the this arm must lie in the range 2 to 14, and is initially set to 2.

□ The second arm can be moved horizontally by 1 unit using **extend()** or **contract()**. The width (**w**) must lie in the range 1 to 10, and is initially set to 1.

□ The depth (**d**) of the third arm can be changed by 1 unit using **lower()** or **raise()**.

The depth cannot be less than 0 but must be less than the height of the main arm ($d < h$).

In the initial position depth **d** is set to 0 (it is not visible).

□ Robot class has two other methods **slowDown**(int factor) and **speedUp**(int factor) to

adjust the time between moves by the specified factor.

□ An item can be picked from the top of the stack of blocks at source using **pick()**.

□ It can be dropped at the top of the stack of blocks at one of the three targets using **drop()**, depending on the size of the block. The height of blocks can only be 1, 2 or 3.

Blocks of height 1, 2 and 3 must be placed in columns 1, 2 and 3 respectively.

Other Constraints

□ Algorithms for moving the blocks must avoid collisions with other blocks and bars.

Height of obstacles and Blocks

The height of bars are preset at 5 each for part **A** but can be input by the user for parts **B** and

C. Similarly the heights of blocks at the source are preset to 1, 2 and 3 (topmost) for parts **A**

and **B**, but can be input by user for part **C**, but the total height of blocks cannot exceed 12.

Part A

All blocks at source are of size 1, 2 and 3 respectively and all bars are of size 5.

Part B

All blocks at source are 1, 2 and 3 as before but the bar heights may vary.

Part C

Block sizes can be specified in the range 1-3 and bar sizes can be specified in the range 1 to 8.

Example: java Robot 786483 3131211

The code you need to write to control the Robot moves

For all three sections below you are required to complete the **control()** method of the **RobotControl** class shown below. The **control()** method will be automatically called when

the program is run. **RobotControl** class has a reference *r* set to refer to the Robot object.

Hence to move the first arm of the Robot upwards call *r.up()*.

```
class RobotControl
{ private Robot r;
public RobotControl(Robot r) { this.r = r; }
public void control(int barHeights[], int blockHeights[])
{
// write your code here
// to control robot using r.up(); r.extend() ...
}
}
```

Arguments Passed to the control method

This method is also passed two arrays containing the heights of the bars (*barHeights*) and the blocks (*blockHeights*) based on user input or default values. If the instructions you specify for the Robot result in hitting an obstacle or going beyond the limits, program will terminate immediately with an appropriate Error Message.

Requirements

A. Assume all the bars are of height 5 units and the blocks are of heights 1 (bottom), 2 and 3 (top) units respectively. Complete the **control()** method of the RobotControl class to move all the blocks to the respective targets.
(Sample call: **java Robot**).

B. Allow the user to supply the heights of the 6 bars as command line arguments where the heights of the bars may vary from 1 to 8. Complete the **control()** method of the RobotControl class to move all the blocks to the respective targets.
(Sample: **java Robot 734561**). Based on these arguments, int array **barHeights[]** will be automatically set as in:
barHeights[0] = 7; barHeights[1] = 3; barHeights[5] = 1;

C. Allow the user to supply the heights of the 6 obstacles as well as the height of the four blocks as command line arguments. Complete the **control()** method of the RobotControl class to move all the blocks to the respective targets.
(Sample: **java Robot 876054 2312**). Based on these arguments int arrays **barHeights[]**, **blockHeights[]** will be automatically set as in .
barHeights[0] = 8; barHeights[1] = 7; barHeights[5] = 4;
blockHeights[0] = 2; blockHeights[1] = 3; ... blockHeights[3] = 2;

Additional marks will also be allocated for performance based on the number of steps used. Note: You are expected to use loops and nested loops whenever appropriate