**CSC141, Computer Science I, Project 3 assignment**

Later submission is not accepted. Please focus on the control of line number to express the number of space or number of symbol. Total = 10+10+20+20+30+30 =120 points.

1. **(10) RightTriangle.java: Write code that reads in a number R from the user, and displays a figure with R rows of "$" characters as the following pattern. For instance, if the user enters a 4 for R, your program should display:**

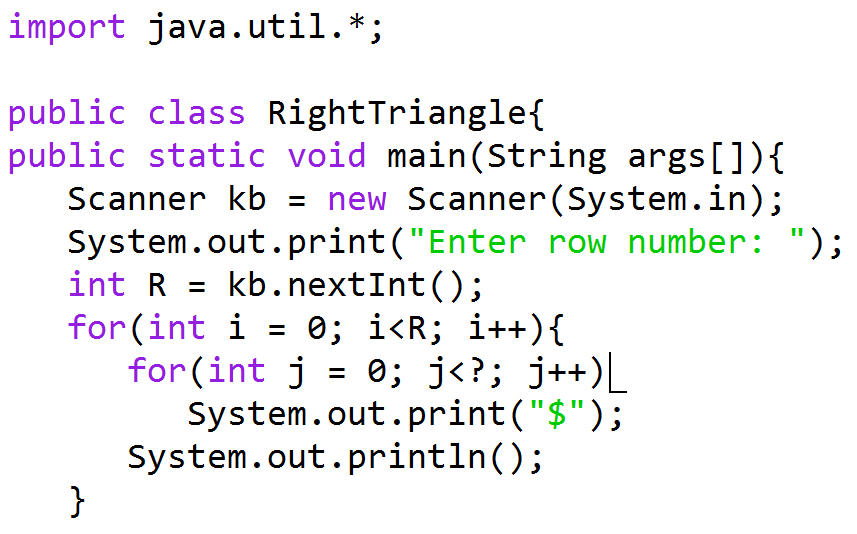
**$$$$**

**$$$**

**$$**

**$**

Step 1) Your program should like this:



Step 2) After work out the column of “ − i” and distance adjustment “?” (by using the expression “− i” and “R”) in the table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | i | # of $ | －i or i\*(-1) | ? |
| 1 | 0 | R |  | R |
| 2 | 1 | R-1 |  | R-1 |
| 3 | 2 | R-2 |  | R-2 |
| 4 | 3 | … |  | … |

Step 3) Try the part “?” that you figure out in the above table in the program of step 1!

1. **(10) RightHollowTriangle.java: Write code that reads in a number R from the user, and displays a figure with R rows of "$" characters as the following pattern. For instance, if the user enters a 4 for R, your program should display:**

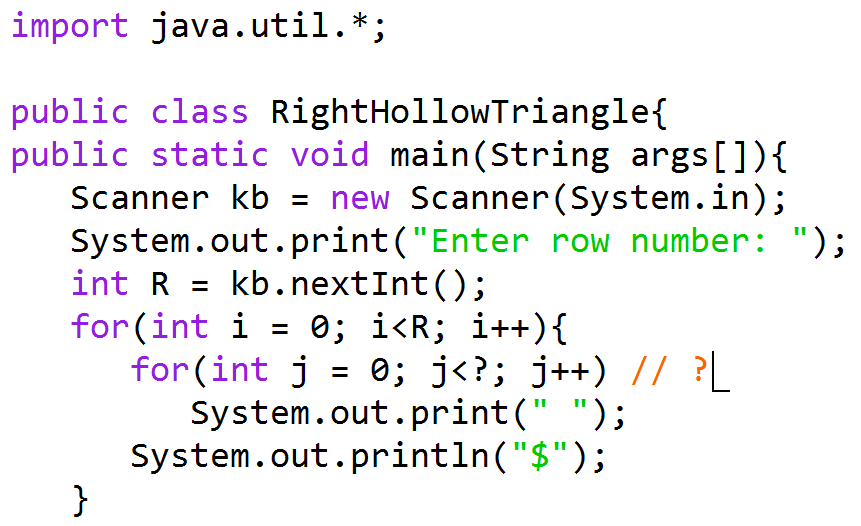
**$**

**$**

**$**

**$**

Step 1) Your program should like this:

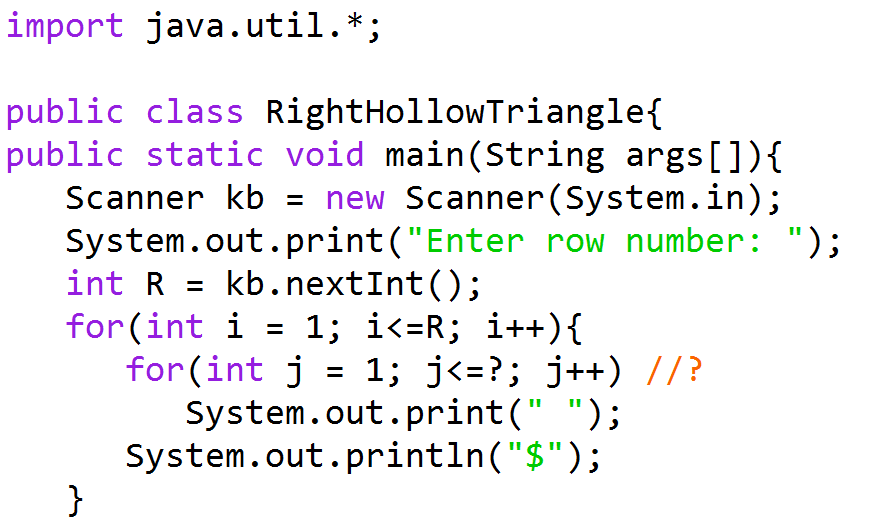


Step 2) Complete the table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | i | # of “ ” (space) | －i or i\*(-1) | ? |
| 1 | 0 | R-1 |  | R |
| 2 | 1 | R-2 |  | R-1 |
| 3 | 2 | R-3 |  | R-2 |
| 4 | 3 | … |  | … |

Step 3) Try the part “?” that you figure out in the above table in the program of step 1!

Step 4) Consider another version:



Step 5) Complete the table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | i | # of “ ” (space) | －i or i\*(-1) | ? |
| 1 | 1 | R-1 |  | R |
| 2 | 2 | R-2 |  | R-1 |
| 3 | 3 | R-3 |  | R-2 |
| 4 | 4 | … |  | .. |

Step 6) Obtain the part “?” and apply it in the program of step 4 (not 1)!

Step 7) Compare what is the difference of “?” parts in step 6 and step 3, and find the impact of “i=0” and “i<” (vs. “i=1” and “i<=”).

Step 8) For evaluation, submit either version in step 3 or step 6.

1. **(20) OddTriangle.java: Write code that reads in a number R from the user, and displays a figure with R rows of "$" characters as the following pattern. For instance, if the user enters a 4 for R, your program should display:**

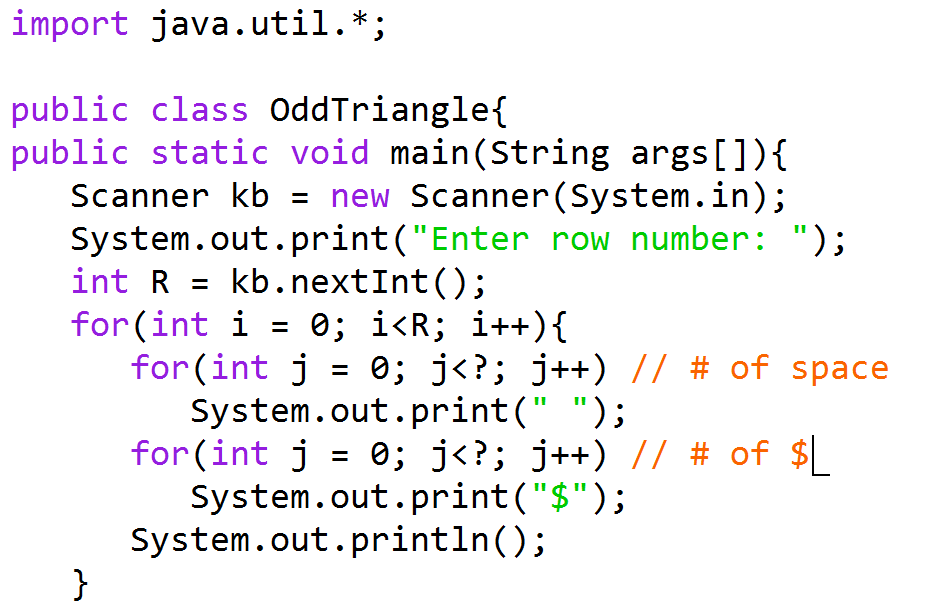
**$**

**$$$**

**$$$$$**

**$$$$$$$**

Step 1)



Step 2) Complete the table and figure out “?” for the number of “ ”

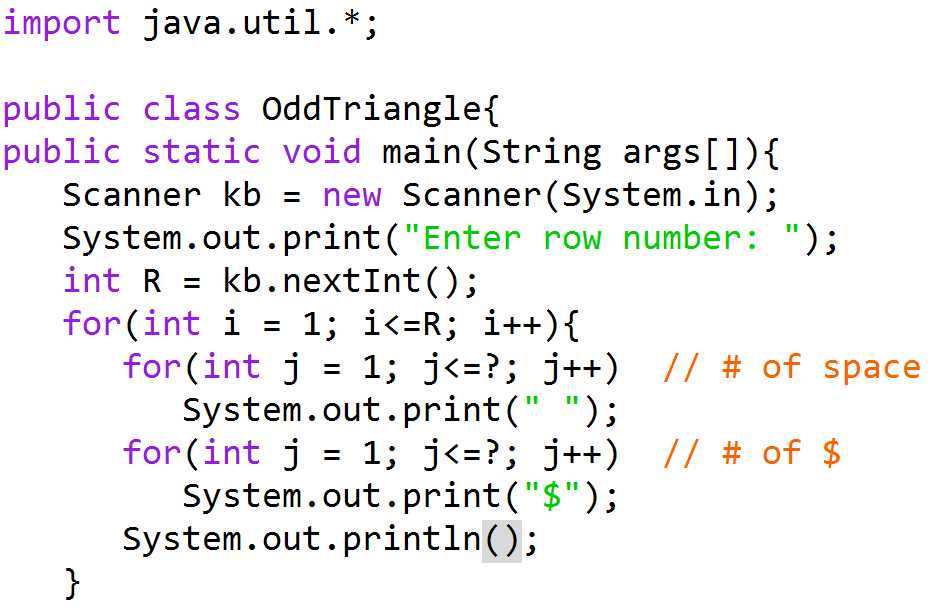
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | i | # of “ ” (space) | －i or i\*(-1) | ? |
| 1 | 0 | R-1 |  | R |
| 2 | 1 | R-2 |  | R-1 |
| 3 | 2 | R-3 |  | R-2 |
| 4 | 3 | … |  | .. |

Step 3) Complete the table and figure out “?” for the number of “$”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | i | # of “$” | 2\*i | ? |
| 1 | 0 | 1 |  | 1 |
| 2 | 1 | 3 |  | 3 |
| 3 | 2 | 5 |  | 5 |
| 4 | 3 | … |  | .. |

Step 4) Apply the result to the program and verify the code execution.

Step 5) Try a different version of code by using “i=1,” “i<=”, “j=1”, and “j<=”



Step 6) Complete the table work and figure “?” parts.

Step 7) Verify the “?” parts in code and compare with the result in early step 4.

Step 8) Submit either version in step 4 or 7 for evaluation.

1. **(20) OddHollowTriangle.java: Write code that reads in a number R from the user, and displays a figure with R rows of "$" characters as the following pattern. For instance, if the user enters a 4 for R, your program should display:**

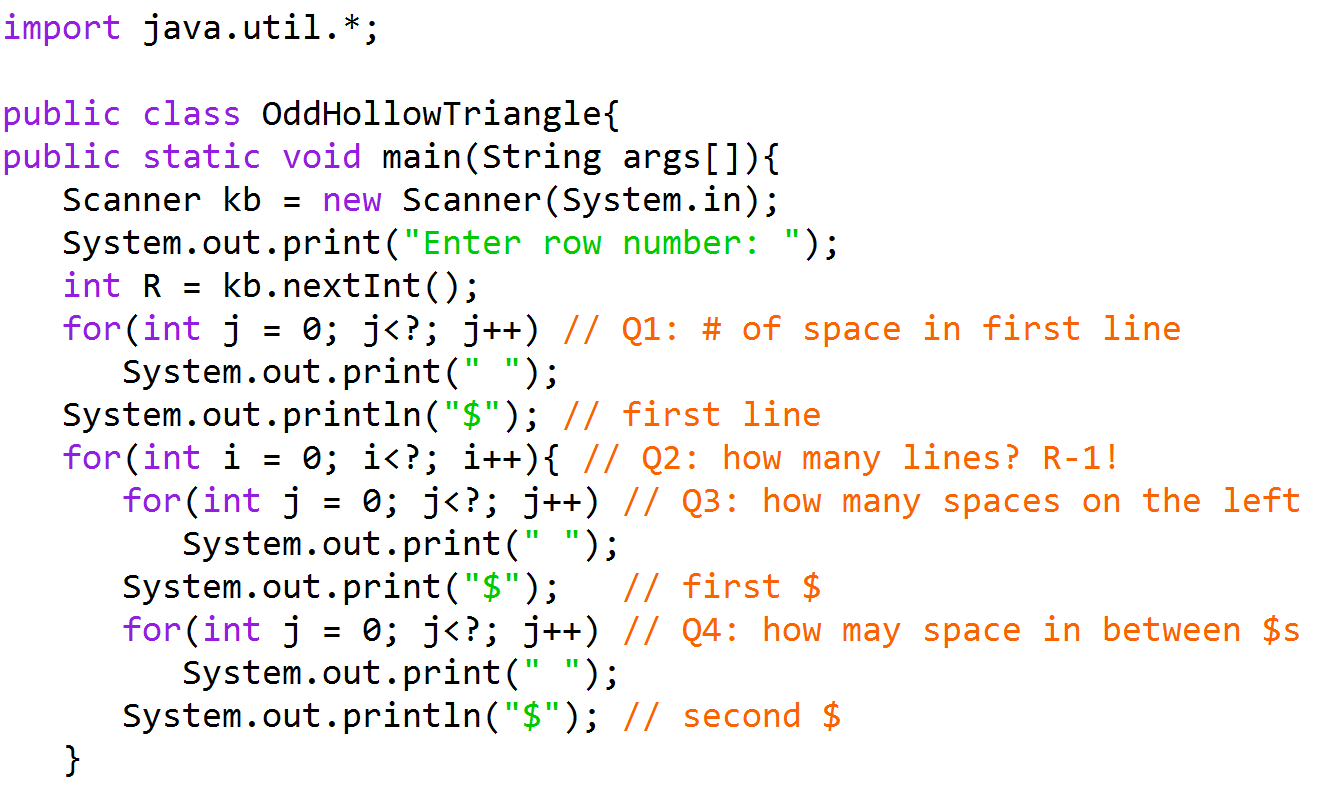
**$**

**$ $**

**$ $**

**$ $**

Step 1)



Step 2) Complete the required table work and figure out “?” in all four places (Q1−4).

Step 3) Verify your code in execution.

1. **(30) HollowDiamond.java: Write code that reads in a number R from the user, and displays a figure with (2\*R-1)-rows and (2\*R-1)-columns of "$" characters as the following pattern. Note that it is not simple 2\*R columns or 2\*R rows! For instance, if the user enters a 4 for R, your program should display:**

**$$$$$$$**

**$$$ $$$**

**$$ $$**

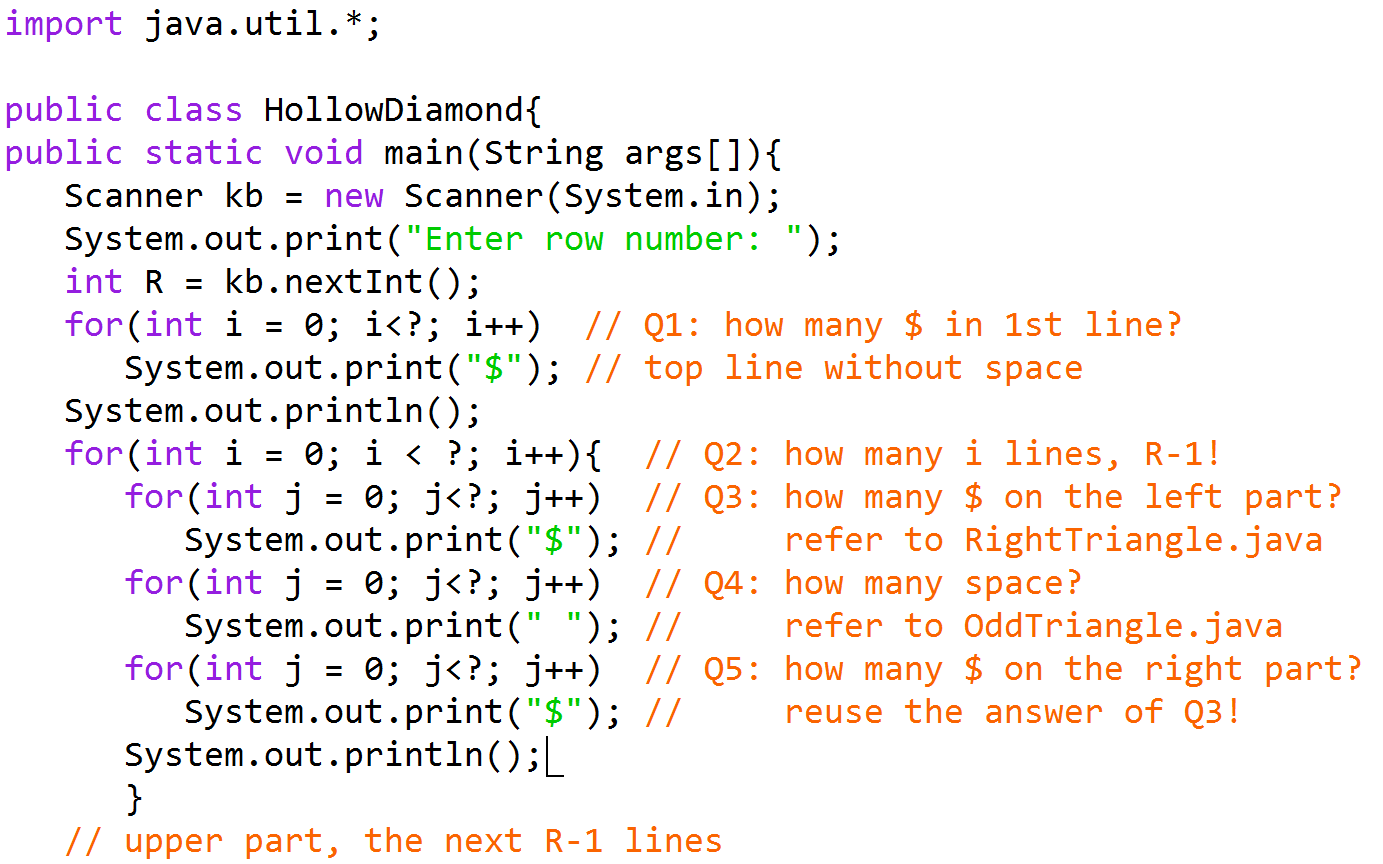
**$ $**

**$$ $$**

**$$$ $$$**

**$$$$$$$**

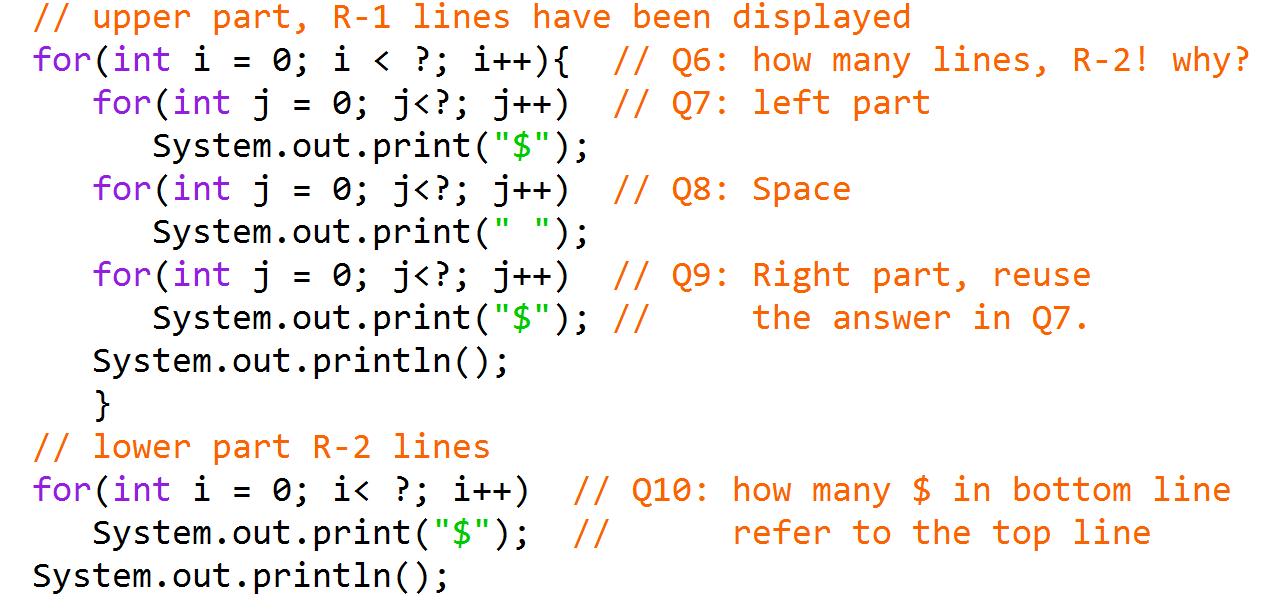
Step 1) Figure out the pattern for the upper part R lines (including the first line without space and the next (R-1)-lines):



Step 2) Complete the table work and figure out “?” in the above five places (Q1−5).

Step 3) Verify the code.

Step 4) Figure out the pattern of the lower part (R-1)-lines (including (R-2)-lines of space display and the bottom line without any space):



Step 5) Complete the required table work and figure out “?” in the above five places (Q6−10).

Step 6) Verify the code.

1. **(30) Diamond.java: Write code that reads in a number R from the user, and displays a figure with (2\*R-1)-rows and (2\*R-1)-columns of "$" characters as the following pattern. For instance, if the user enters a 4 for R, your program should display:**

**$**

**$ $**

**$ $**

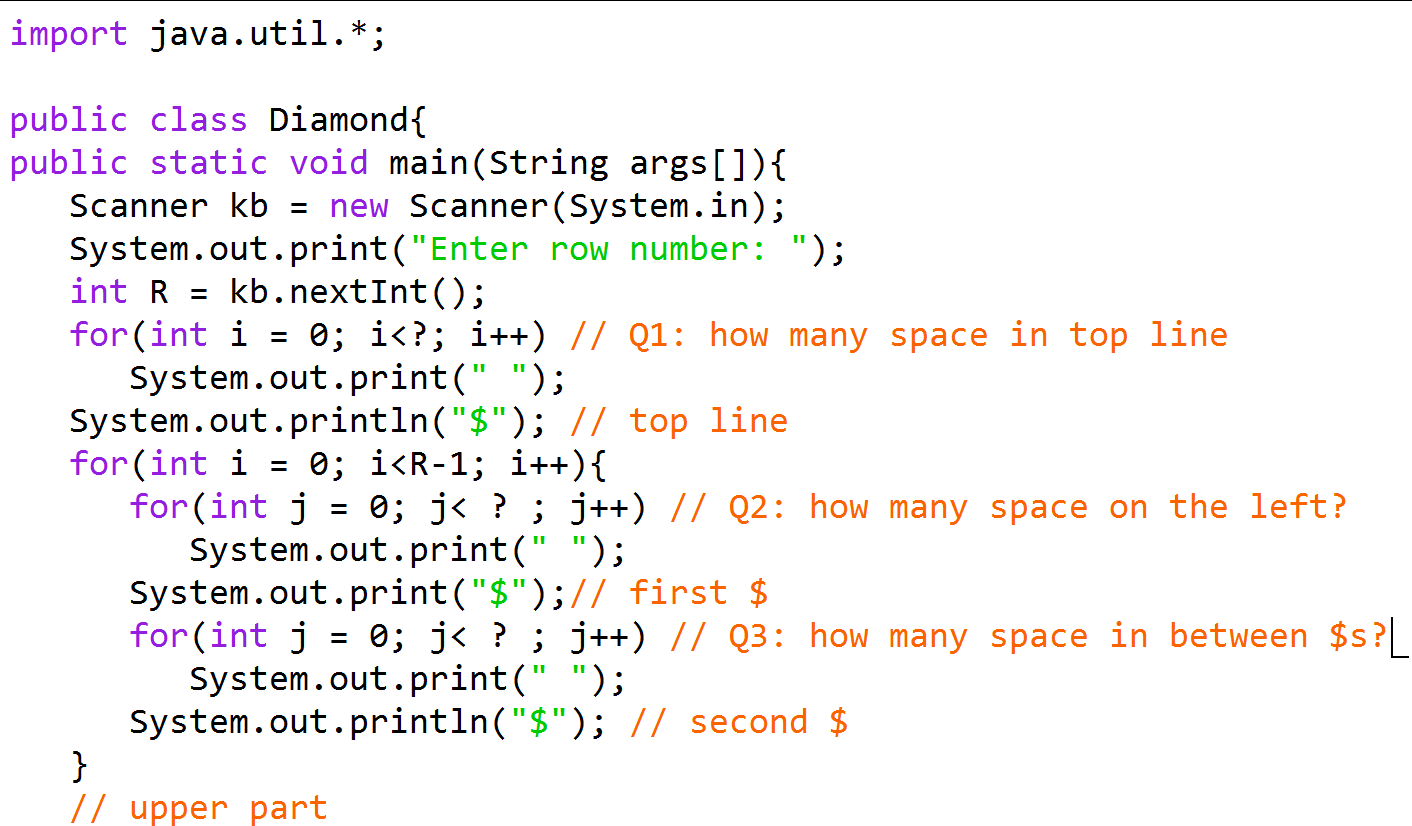
**$ $**

**$ $**

**$ $**

**$**

Step 1) Complete the upper part R lines (Q1-3):

****

Step 2) Figure out the pattern for the lower part (R-1)-lines (Q4-6):

