Never give next line in the inner loop.

Page No. 26
Date: / /

★ PATTERNS

* Print Star Pattern

```
*
* *
* * *
* * * *
```

- Nested Loops: Loops inside the loop.

Firstly,
1. lines (4)
   Outer loop → 4 times
2. No. of times (i)
   inner loop → i times
3. What to print? "*"

Dry run

```
for (int line=1; line<=4; line++)
{
  for (int star=1; star<=line; star++) {
    syso ("*");
  } system.out.println();
}
```

- use print not println.

* Print Inverted Star Pattern

```
* * * *
* * *
* *
*
```

star = n - line + 1

Dry run → int n = 4;

```
for (line=1; line<=4; line++) {
  for (int star=1; star<=n-line+1;
       star++) {
    syso ("*");
  } system.out.println ();
} system.o
```

* Print Half-Pyramid Pattern

```
1
1 2
1 2 3
1 2 3 4
```

Dry run →

```
for (int line=1; line<=4; line++) {
  for (int num=1; num<=line; num++) {
    syso (num);
  } system.out.println();
}
```

* Print Character Pattern

A
B C
D E F
G H I J

Dry run → int n = 4;
char ch = 'A';

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= i; j++) {
        syso (ch);
        ch++;
    }
    system.out.println();
}
```

★ ADVANCE PATTERN

* Print Hollow Rectangle Pattern

*	*	*	*	*	ROW 1
*				*	ROW 2
*				*	ROW 3
*	*	*	*	*	ROW 4

COL
1 2 3 4 5

① total lines (total rows)
② boundary, row → 1, 4
column → 1, 5
∴ (row = 1 || col = 1 || row = 4 || col = 5).

```
public static void hollow_rectangle (int totrows, int totcols){
    for (int i = 1; i <= totrows; i++) {
        for (int j = 1; j <= totcols; j++) {
            if (i == 1 || i == totrows || j == 1 || j == totcols) {
                system.out.print ("* ");
            }
            else {
                system.out.print ("  ");
            }
        } Syso( );
    }
}
```

DRY RUN → j <= totcols total rows = 4
i = 1, j = 1, 2, 3, 4, 5 total cols = 5
i = 2, j = 1, 2, 3, 4, 5
i = 3, j = 1, 2, 3, 4, 5
i = 4, j = 1, 2, 3, 4, 5

* * * * *
* *
* *
* * * * *

* Rotated half pyramids

```
*          -> row 1  => space = 3 ,  * = 1 => row = *
* *        -> row 2  => space = 2 ,  * = 2 => row = *
* * *      -> row 3  => space = 1 ,  * = 3 => row = *
* * * *    -> row 4  => space = 0 ,  * = 4 => row = *
```

```
public static void half_pyramid (int row) {
  for (int i = 1; i <= row; i++) {
    for (int j = 1; j <= row - i; j++) {
      System.out.print(" ");
    }
    for (int j = 1; j <= i; j++) {
      System.out.print("* ");
    }
    syso();
  }
}
```

ROW = 4

DRY RUN =>

i = 1 , j = 1 to (4-1) = 1 to 3
print " "
j = 1 to 1 <= i(1) = 1

i = 2 , j = 2 to (4-2) = 1 to 2
j = 2 to 2 <= i(2) = 2.

i = 3 , j = 3 to (4-3) = 1 to 1
j = 3 to 3 <= i(3) = 3.

i = 4 , j = 4 to (4-4) = 0 spaces
j = 4 to 4 <= i(4) = 4.

```
_ _ _ *
_ _ * *
_ * * *
* * * *
```

* Inverted half-pyramid with Numbers

	1 to 5	Printing	
1 2 3 4 5	→ line = 1	1 to 5	i = 1 to 5
1 2 3 4	→ line = 2	1 to 4	j = n - i + 1
1 2 3	→ line = 3	1 to 3	= 5 - 1 + 1
1 2	→ line = 4	1 to 2	= 5
1	→ line = 5	1 to 1	

```
public static void half_num_pyramid (int n) {
    for (i=1; i<=n; i++) {
        for (j=1; j<=n-i+1; j++) {
            System.out.print (j + " ");
        }
        Syso();
    }
}
psvm {
    half_num_pyramid (5);
}
```

DRY RUN ⇒ (n - i + 1)

line → i=1, j=1; j≤5-1+1=5 → print j

i=2, j=1; j≤5-2+1=4 → print j

i=3, j=1; j≤5-3+1=3 → print j

i=4, j=1; j≤5-4+1=2 → print j

i=5, j=1; j≤5-5+1=1 → print j

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

* FLOYD's Triangle

		i=1 to 5	j=1 to i
1	→	line=1	counter = 1;
2 3	→	line=2	SYSO (counter)
4 5 6	→	line=3	counter++;
7 8 9 10	→	line=4	
11 12 13 14 15	→	line=5	

```
public static void floydTriangle (int rows) {
    int counter = 1;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            System.out.print (counter);
            counter++;
        }
        SYSO ();
    }
}
psvm {
    floydTriangle (5);
}
```

DRY RUN ⇒

* 0-1 Triangle

```
                    j<=i
1                   i=1 => j = 1 to 1
0 1                 i=2 => j = 1 to 2
1 0 1               i=3 => j = 1 to 3
0 1 0 1             i=4 => j = 1 to 4
1 0 1 0 1           i=5 => j = 1 to 5
```

(row+col)
(i+j) -> even -> "1"
(i+j) -> odd -> "0"

```
public static void bintriangle (int rows) {
    for (i=1; i<= rows; i++) {
        for (j=1; j<=i; j++) {
            if ((i+j) % 2 == 0) {
                System.out.print ("1");
            else {
                System.out.print ("0");
            }
        }
        syso();
    }
}
psvm {
    bintriangle (5);
}
```

i=1, j=1 to j<=1, (1+1)%2==0
True => "1"

i=2, j=1 to j<=2, (2+1)%2==0
False => "0"

j=2 to j<=2, (2+2)%2==0
True => "1"

i=3, j=1 to j<=3, (3+1)%2==0
True => "1"

j=2 to j<=3, (3+2)%2==0
False => "0"

j=3 to j<=3, (3+3)%2==0
True => "1"

i=4,

DRY RUN =>

```
1
0 1
1 0 1
```

अगर कोई Mirror image बनाना है तो
Simply Outer loop n से 1 तक चलादो
⇒ for(i=n; i>=1; i--)

Page No. 44
Date: / /

* BUTTERFLY Pattern

	i	2(n-i)
*	i=1	stars ⇒ 1 then spaces ⇒ 6
* *	i=2	stars ⇒ 2 then space ⇒ 4
* * *	i=3	stars ⇒ 3 " spaces ⇒ 2
* * * * * * * *	i=4	stars ⇒ 4 " spaces ⇒ 0
* * * * * * * *	i=4	
* * * * * *	i=3	
* * * *	i=2	
* *	i=1	

1 to n
n to 1

```
public static void butterfly (int n) {
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=i; j++) {
            System.out.print("*");
        }
        for (int j=1; j<=2*(n-i); j++) {
            System.out.print(" ");
        }
        for (int j=1; j<=i; j++) {
            System.out.print("*");
        }
        syso();
    }
    for (int i=n; i>=1; i--) {
        same logic code
    }
}
psvm {
    butterfly (4);
}
```

i=1, j=1, j≤1 ⇒ "*"
j=1, j≤2(4-1)=6
, j=2, 2≤6
j=3, 3≤6
, j=1, j≤1 ⇒ "*"

* Solid Rhombus

```
        * * * * *   i=1
      * * * * *     i=2
    * * * * *       i=3
  * * * * *         i=4
* * * * *           i=5
```

stars = 5 ; Spaces = 4

stars - i

" " = 3

" " = 2

" " = 1

" " = 0

```
public static void rhombus (int n) {
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n-i; j++) {
            System.out.print (" ");
        }
        for (int j=1; j<=n; j++) {
            System.out.print ("* ");
        }
        syso();
    }
}
psvm {
    rhombus (5);
}
```

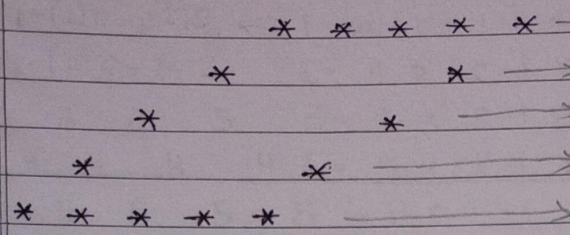
i=1, j=1, j≤5-i=4, print " "
j=2, j≤4 print " "
j=3, j≤3 print " "
j=4, j≤4 print " "
j=1, j≤n i.e 5, print "*"
j=2, j≤5 print "*"
j=3, j≤5 print "*"
j=4, j≤5 print "*"
j=5 j≤5 print "*"

DRY RUN ⇒

```
        * * * * *
      * * * * *
```

i=2, j=1, j≤5-2=3, print " "
j=2, j≤3 Print " "
j=3, j≤3 print " "
j=1, j≤5 "*"
j=2, j≤5 "*"
j=3, j≤5 "*"
j=4
j=5, j≤5 "*"

- Hollow Rhombus




```
public static void HollowRhombus(int n) {
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n-i; j++) {
            System.out.print(" ");
        }
        for (int j=1; j<=n ; j++) {
            if (i==1 || i==n || j==1 || j==n) {
                System.out.print("* ");
            }
            else {
                system.out.print(" ");
            }
        }
        syso();
    }
}
```

DRY RUN =>

- **DIAMOND PATTERN**

Spaces stars spaces

← 3 → * ← 3 → i=1 → star=1 ⇒ 2i-1 = 2(1)-1 = 1
← 2 → * * * ← 2 → i=2 " =3 2i-1 = 2(2)-1 = 3
← 1 → * * * * * ← 1 → i=3 " =5
* * * * * * * i=4 " =7
* * * * * * *
  * * * * *
    * * *
      *

```
public static void diamond_pattern (int n) {
   for (int i=1; i<=n; i++) {
      for (int j=1; j<=n-i; j++) {
         System.out.print ("  ");
      }
      for (int j=1; j<=(2*i)-1; j++) {
         System.out.print ("* ");
      }
      for (int j=1; j<=n-i; j++) {
         System.out.print ("  ");
      }
      syso();
   }
   for (int i=n; i>=1; i--) {
      //same logic
   }
}
```

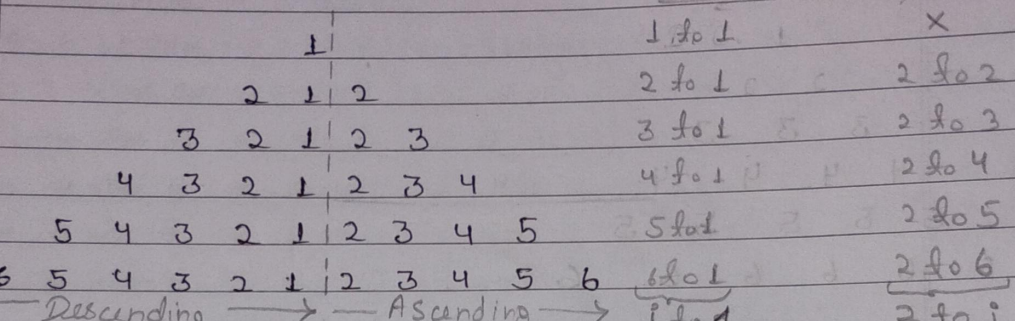
- **Number Pyramid Pattern**

		spaces	Num
1	i=1	5	1 → 1x
2 2	i=2	4	2 → 2x
3 3 3	i=3	3	3 → 3x
4 4 4 4	i=4	2	4 → 4x
5 5 5 5 5	i=5	1	5 → 5x
6 6 6 6 6 6	i=6	0	6 → 6x

```
public static void num_pyramid (int n) {
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n-i; j++) {
            System.out.print (" ");
        }
        for (int j=1; j<=i; j++) {
            System.out.print (i + " ");
        }
        for (int j=1; j<=n-i; j++) {
            System.out.print (" ");
        }
        syso();
    }
}
psvm {
    num-pyramid (5);
}
```

Page No. 49
Date : / /

- **PALINDROMIC Pattern**




```
public static void palindrome_pyramid (int n) {
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n-i; j++) {
            System.out.print(" ");
        }
        for (int j=i; j>=1; j--) {
            system.out.print(j);
        }
        for (int j=2; j<=i; j++) {
            system.out.print(j);
        }
        syso();
    }
}
```