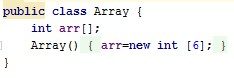
**Code Explaination**

**public static void** main(String[] args) {  
 Scanner input=**new** Scanner(System.***in***);  
 System.***out***.print(**"\nEnter number of round : "**);  
 **int** n=input.nextInt();  
 String number;  
 Array array1=**new** Array(),array2=**new** Array();  
 RollTheDice dice=**new** RollTheDice();  
 *//int arr[];* **for**(**int** i=0;i<n;i++) // Θ(n)  
 {  
 System.***out***.print(**"\nEnter sequence of the rotation for dice : "**);  
 number=input.next();  
*// System.out.println(s);* dice.initializeDice(i,array1,array2);  
 **for**(**int** j=0;j<number.length();j++) Θ(n)  
 {  
 **switch**(number.charAt(j))  
 {  
 **case 'F'**:  
 dice.forwardRotation(array1,array2);  
 **break**;  
 **case 'B'**:  
 dice.backwardRotation(array1,array2);  
 **break**;  
 **case 'L'**:  
 dice.leftRotation(array1,array2);  
 **break**;  
 **case 'R'**:  
 dice.rightRotation(array1,array2);  
 **break**;  
 **case 'C'**:  
 dice.clockWiseRotation(array1,array2);  
 **break**;  
 **case 'D'**:  
 dice.counterClockWiseRotation(array1,array2);  
 **break**;  
 **default**:  
 System.***out***.println(**"\nWrong letter\n"**);  
 System.*exit*(0);  
 }  
 }  
 System.***out***.println(**"\nNumber in the front side -"**+(i+1)+array1.**arr**[4]);  
 }  
 input.close();  
 }  
  
 **public void** counterClockWiseRotation(Array ob1,Array ob2)  
 {  
 **int** arr[]=ob1.**arr**,brr[]=ob2.**arr**;   
 arr[2]=brr[5];  
 arr[3]=brr[4];  
 arr[4]=brr[2];  
 arr[5]=brr[3];  
 **for**(**int** i=0;i<6;i++) Θ(n)  
 brr[i]=arr[i];  
 ob1.**arr**=arr;  
 ob2.**arr**=brr;  
 }  
  
 **public void** clockWiseRotation(Array ob1,Array ob2)  
 {  
 **int** arr[]=ob1.**arr**,brr[]=ob2.**arr**;   
 arr[2]=brr[4];  
 arr[3]=brr[5];  
 arr[4]=brr[3];  
 arr[5]=brr[2];  
 **for**(**int** i=0;i<6;i++) Θ(n)  
 brr[i]=arr[i];  
 ob1.**arr**=arr;  
 ob2.**arr**=brr;  
 }  
  
 **public void** rightRotation(Array ob1,Array ob2)  
 {  
 **int** arr[]=ob1.**arr**,brr[]=ob2.**arr**;  
 arr[0]=brr[2];  
 arr[1]=brr[3];  
 arr[2]=brr[1];  
 arr[3]=brr[0];  
 **for**(**int** i=0;i<6;i++) Θ(n)  
 brr[i]=arr[i];  
 ob1.**arr**=arr;  
 ob2.**arr**=brr;  
 }  
  
 **public void** leftRotation(Array ob1,Array ob2)  
 {  
 **int** arr[]=ob1.**arr**,brr[]=ob2.**arr**;  
 arr[0]=brr[3];  
 arr[1]=brr[2];  
 arr[2]=brr[0];  
 arr[3]=brr[1];  
 **for**(**int** i=0;i<6;i++) Θ(n)  
 brr[i]=arr[i];  
 ob1.**arr**=arr;  
 ob2.**arr**=brr;  
 }  
  
 **public void** backwardRotation(Array ob1,Array ob2)  
 {  
 **int** arr[]=ob1.**arr**,brr[]=ob2.**arr**;  
 arr[0]=brr[4];  
 arr[1]=brr[5];  
 arr[4]=brr[1];  
 arr[5]=brr[0];  
 **for**(**int** i=0;i<6;i++) Θ(n)  
 brr[i]=arr[i];  
 ob1.**arr**=arr;  
 ob2.**arr**=brr;  
 }  
  
 **public void** forwardRotation(Array ob1,Array ob2)  
 {  
 **int** arr[]=ob1.**arr**,brr[]=ob2.**arr**;  
 arr[0]=brr[5];  
 arr[1]=brr[4];  
 arr[4]=brr[0];  
 arr[5]=brr[1];  
 **for**(**int** i=0;i<6;i++) Θ(n)  
 brr[i]=arr[i];  
 ob1.**arr**=arr;  
 ob2.**arr**=brr;  
 }  
  
 **public void** initializeDice(**int** i,Array ob1,Array ob2)  
 {  
 **int** arr[]=ob1.**arr**,brr[]=ob2.**arr**;  
 arr[0]=brr[0]=1;  
 arr[1]=brr[1]=6;  
 arr[2]=brr[2]=3;  
 arr[3]=brr[3]=4;  
 arr[4]=brr[4]=2;  
 arr[5]=brr[5]=5;  
 ob1.**arr**=arr;  
 ob2.**arr**=brr;  
 System.***out***.println(**"\nInitial position for the Dice - "**+(i+1)+**"\n"**);  
 System.***out***.println(**"Top Bottom Left Right Front Back"**);  
 System.***out***.println(**" 1 6 3 4 2 5"**);  
 }  
  
}

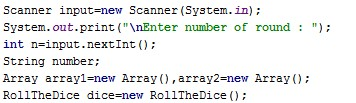
Solution

I have declared two objects of type Array to declare two array of size 6 and this two array are used to keep track which number is in top,bottom and soon. index - 0 of each array keep track of top, index - 1 of each array keep track of bottom and soon. According to given question each function changes the content of each array for this second array is used.

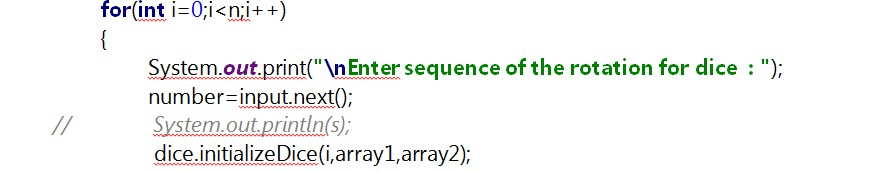
**Step1**: I have declare the size of the array as 6 so it will fits all of the dimension of the dice which is including F(Forward) B(Backward) L(Left) R(Right) C(Clockwise) and D(Counter Wise)



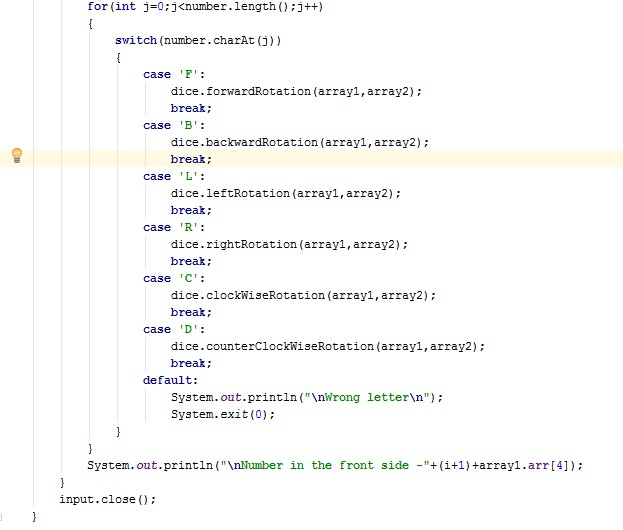
**Step2**:After We create a variable which call arr [] now we will work on the another class which is called ”RollTheDice” first we create an input scanner variable to store the input variable from user when they are input the number of the dice and Sequence of the rotation. After that we create new 2 Array variable to assign the variable of the dice. Also we create a new object called”dice” to reduce some code.



**Step3:** For this step it will ask user to input the sequence of the rotation for a dice. And also we already initiate the value of the dice as “163425”



**Step4:** In this step we will ask the user for the sequence of the rotation that they want.We provide ‘F’ as Forward ‘B’ as Backward ‘L’ as Left ‘R’ as Right ‘C’ as Clockwise ‘D’ as CounterClockwise if the user input another than these alphabet the program will return “Wrong letter” and may ask you to input the alphabet or choice again.



**Step5:**  i have just changed the contents of the array and second array is used as temporary purpose and this method is used for all functions so each function works similarly as other except that in each function changes done in different index. These following method are working the same but different result.

public void counterClockWiseRotation(Array ob1,Array ob2)  
   {  
       int arr[]=ob1.arr,brr[]=ob2.arr;  
       arr[2]=brr[5];  
       arr[3]=brr[4];  
       arr[4]=brr[2];  
       arr[5]=brr[3];  
       for(int i=0;i<6;i++)  
           brr[i]=arr[i];  
       ob1.arr=arr;  
       ob2.arr=brr;  
   }  
    
   public void clockWiseRotation(Array ob1,Array ob2)  
   {  
       int arr[]=ob1.arr,brr[]=ob2.arr;  
       arr[2]=brr[4];  
       arr[3]=brr[5];  
       arr[4]=brr[3];  
       arr[5]=brr[2];  
       for(int i=0;i<6;i++)  
           brr[i]=arr[i];  
       ob1.arr=arr;  
       ob2.arr=brr;  
   }  
    
   public void rightRotation(Array ob1,Array ob2)  
   {  
       int arr[]=ob1.arr,brr[]=ob2.arr;  
       arr[0]=brr[2];  
       arr[1]=brr[3];  
       arr[2]=brr[1];  
       arr[3]=brr[0];  
       for(int i=0;i<6;i++)  
           brr[i]=arr[i];  
       ob1.arr=arr;  
       ob2.arr=brr;  
   }  
    
   public void leftRotation(Array ob1,Array ob2)  
   {  
       int arr[]=ob1.arr,brr[]=ob2.arr;  
       arr[0]=brr[3];  
       arr[1]=brr[2];  
       arr[2]=brr[0];  
       arr[3]=brr[1];  
       for(int i=0;i<6;i++)  
           brr[i]=arr[i];  
       ob1.arr=arr;  
       ob2.arr=brr;  
   }  
    
   public void backwardRotation(Array ob1,Array ob2)  
   {  
       int arr[]=ob1.arr,brr[]=ob2.arr;  
       arr[0]=brr[4];  
       arr[1]=brr[5];  
       arr[4]=brr[1];  
       arr[5]=brr[0];  
       for(int i=0;i<6;i++)  
           brr[i]=arr[i];  
       ob1.arr=arr;  
       ob2.arr=brr;  
   }  
    
   public void forwardRotation(Array ob1,Array ob2)  
   {  
       int arr[]=ob1.arr,brr[]=ob2.arr;  
       arr[0]=brr[5];  
       arr[1]=brr[4];  
       arr[4]=brr[0];  
       arr[5]=brr[1];  
       for(int i=0;i<6;i++)  
           brr[i]=arr[i];  
       ob1.arr=arr;  
       ob2.arr=brr;  
   }

//Initialize Dice We will initialize the value of array by using these method

## public void initializeDice(int i,Array ob1,Array ob2)    {        int arr[]=ob1.arr,brr[]=ob2.arr;        arr[0]=brr[0]=1;        arr[1]=brr[1]=6;        arr[2]=brr[2]=3;        arr[3]=brr[3]=4;        arr[4]=brr[4]=2;        arr[5]=brr[5]=5;        ob1.arr=arr;        ob2.arr=brr;        System.out.println("\nInitial position for the Dice - "+(i+1)+"\n");        System.out.println("Top Bottom Left Right Front Back");        System.out.println(" 1 6 3 4 2 5");    }

## Common Data Structure Operations

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data Structure | Time Complexity | | | | | | | | Space Complexity |
|  | Average | | | | Worst | | | | Worst |
|  | Access | Search | Insertion | Deletion | Access | Search | Insertion | Deletion |  |
| [Array](http://en.wikipedia.org/wiki/Array_data_structure) | Θ(1) | Θ(n) | Θ(n) | Θ(n) | O(1) | O(n) | O(n) | O(n) | O(n) |
| [Stack](http://en.wikipedia.org/wiki/Stack_(abstract_data_type)) | Θ(n) | Θ(n) | Θ(1) | Θ(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| [Queue](http://en.wikipedia.org/wiki/Queue_(abstract_data_type)) | Θ(n) | Θ(n) | Θ(1) | Θ(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| [Singly-Linked List](http://en.wikipedia.org/wiki/Singly_linked_list#Singly_linked_lists) | Θ(n) | Θ(n) | Θ(1) | Θ(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| [Doubly-Linked List](http://en.wikipedia.org/wiki/Doubly_linked_list) | Θ(n) | Θ(n) | Θ(1) | Θ(1) | O(n) | O(n) | O(1) | O(1) | O(n) |
| [Skip List](http://en.wikipedia.org/wiki/Skip_list) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | O(n) | O(n) | O(n) | O(n) | O(n log(n)) |
| [Hash Table](http://en.wikipedia.org/wiki/Hash_table) | N/A | Θ(1) | Θ(1) | Θ(1) | N/A | O(n) | O(n) | O(n) | O(n) |
| [Binary Search Tree](http://en.wikipedia.org/wiki/Binary_search_tree) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | O(n) | O(n) | O(n) | O(n) | O(n) |
| [Cartesian Tree](https://en.wikipedia.org/wiki/Cartesian_tree) | N/A | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | N/A | O(n) | O(n) | O(n) | O(n) |
| [B-Tree](http://en.wikipedia.org/wiki/B_tree) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| [Red-Black Tree](http://en.wikipedia.org/wiki/Red-black_tree) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| [Splay Tree](https://en.wikipedia.org/wiki/Splay_tree) | N/A | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | N/A | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| [AVL Tree](http://en.wikipedia.org/wiki/AVL_tree) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |
| [KD Tree](http://en.wikipedia.org/wiki/K-d_tree) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | Θ(log(n)) | O(n) | O(n) | O(n) | O(n) | O(n) |

**Array Sorting Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Time Complexity | | | Space Complexity |
|  | Best | Average | Worst | Worst |
| [Quicksort](http://en.wikipedia.org/wiki/Quicksort) | Ω(n log(n)) | Θ(n log(n)) | O(n^2) | O(log(n)) |
| [Mergesort](http://en.wikipedia.org/wiki/Merge_sort) | Ω(n log(n)) | Θ(n log(n)) | O(n log(n)) | O(n) |
| [Timsort](http://en.wikipedia.org/wiki/Timsort) | Ω(n) | Θ(n log(n)) | O(n log(n)) | O(n) |
| [Heapsort](http://en.wikipedia.org/wiki/Heapsort) | Ω(n log(n)) | Θ(n log(n)) | O(n log(n)) | O(1) |
| [Bubble Sort](http://en.wikipedia.org/wiki/Bubble_sort) | Ω(n) | Θ(n^2) | O(n^2) | O(1) |
| [Insertion Sort](http://en.wikipedia.org/wiki/Insertion_sort) | Ω(n) | Θ(n^2) | O(n^2) | O(1) |
| [Selection Sort](http://en.wikipedia.org/wiki/Selection_sort) | Ω(n^2) | Θ(n^2) | O(n^2) | O(1) |
| [Tree Sort](https://en.wikipedia.org/wiki/Tree_sort) | Ω(n log(n)) | Θ(n log(n)) | O(n^2) | O(n) |
| [Shell Sort](http://en.wikipedia.org/wiki/Shellsort) | Ω(n log(n)) | Θ(n(log(n))^2) | O(n(log(n))^2) | O(1) |
| [Bucket Sort](http://en.wikipedia.org/wiki/Bucket_sort) | Ω(n+k) | Θ(n+k) | O(n^2) | O(n) |
| [Radix Sort](http://en.wikipedia.org/wiki/Radix_sort) | Ω(nk) | Θ(nk) | O(nk) | O(n+k) |
| [Counting Sort](https://en.wikipedia.org/wiki/Counting_sort) | Ω(n+k) | Θ(n+k) | O(n+k) | O(k) |
| [Cubesort](https://en.wikipedia.org/wiki/Cubesort) | Ω(n) | Θ(n log(n)) | O(n log(n)) | O(n) |