CS162 ASSIGNMENT 4

NAME:

ARCHIT AGRAWAL

ROLL NO. :

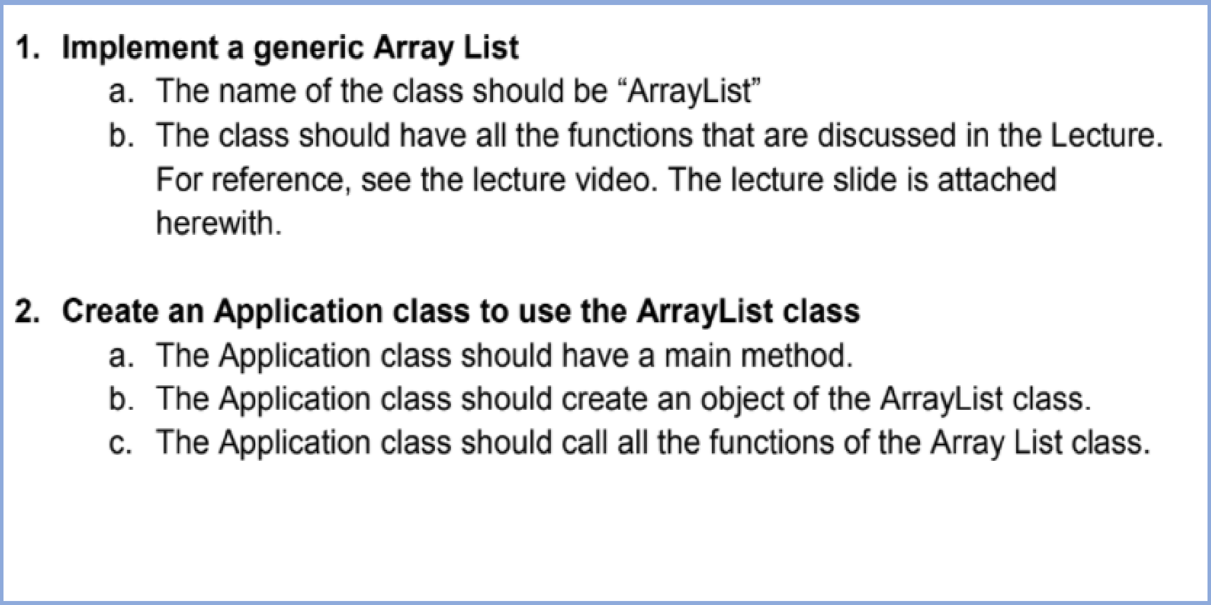
202052307

SECTION:

A

ARCHIT AGRAWAL 202052307

Question



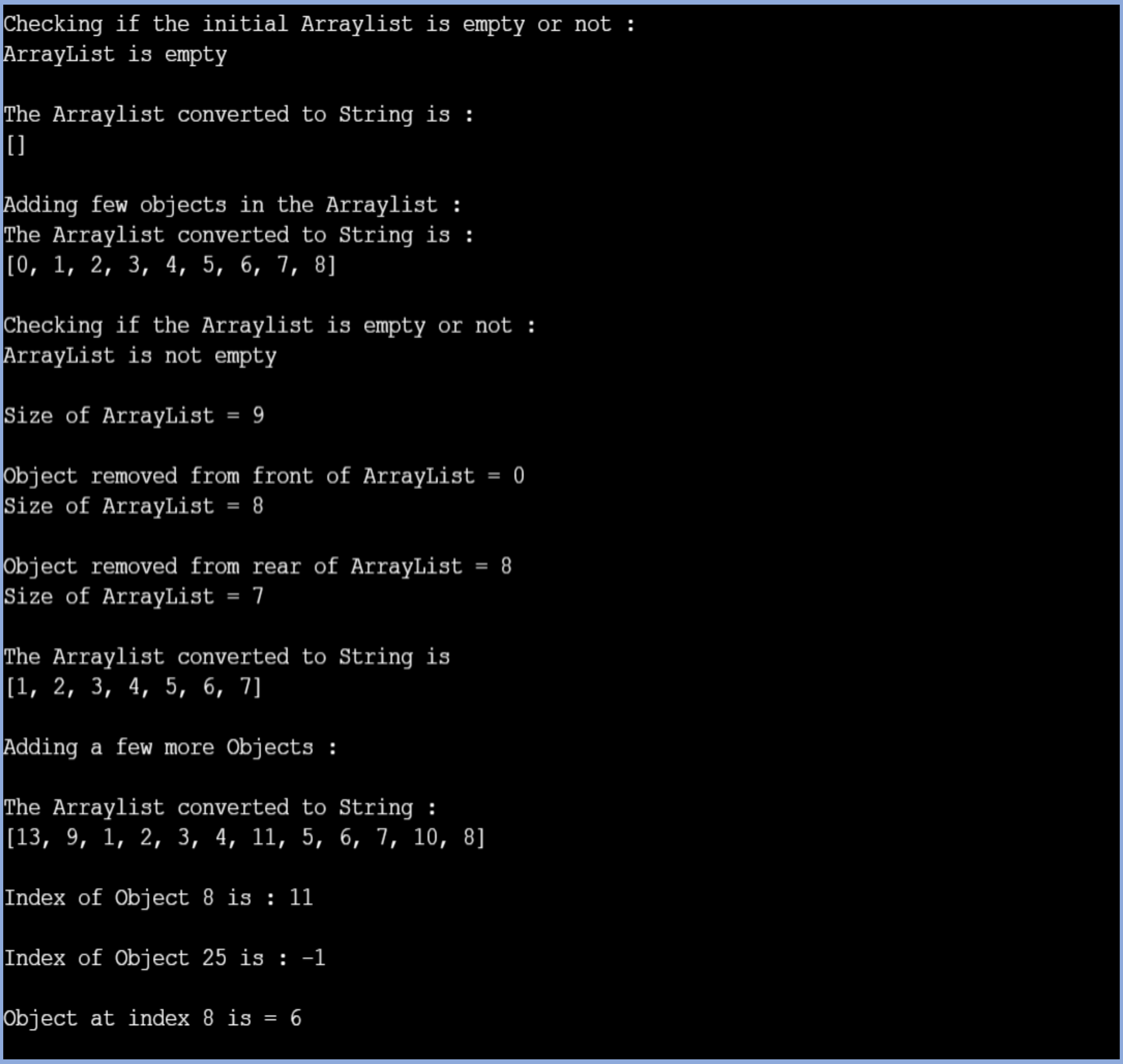
1. **Implement a generic Array List**
    a. The name of the class should be "ArrayList"
    b. The class should have all the functions that are discussed in the Lecture. For reference, see the lecture video. The lecture slide is attached herewith.

2. **Create an Application class to use the ArrayList class**
    a. The Application class should have a main method.
    b. The Application class should create an object of the ArrayList class.
    c. The Application class should call all the functions of the Array List class.

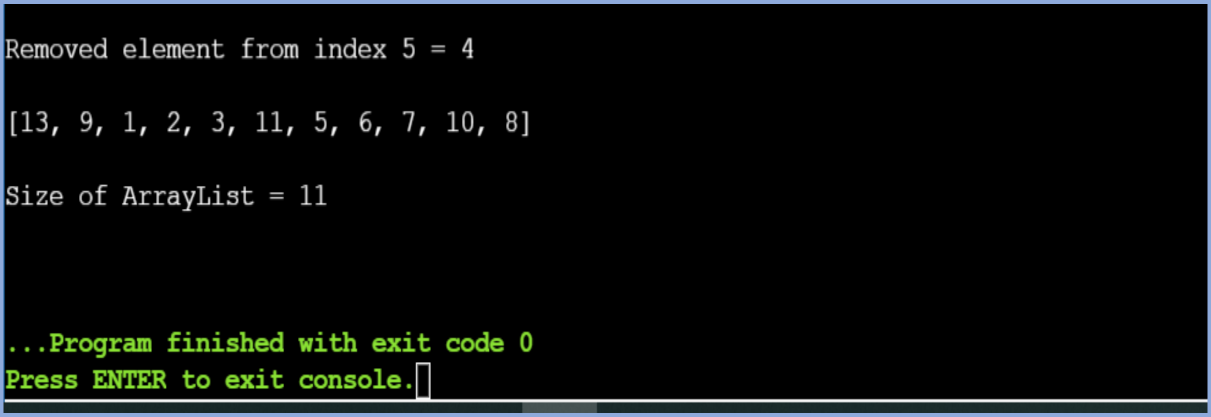
CODE

```
//package com.company;
public class Application{
    public static void main(String []args){
        ArrayList <Integer> list = new ArrayList<Integer>(10);  // creating
an arraylist with initial capacity of 10

        System.out.println();
        System.out.println("Checking if the initial Arraylist is empty or
not : ");
        if(list.isEmpty()){
            System.out.println("ArrayList is empty");
        } else {
            System.out.println("ArrayList is not empty");
        }
        System.out.println();

        System.out.println("The Arraylist converted to String is :");
        System.out.println(list.toString());
        System.out.println();

        System.out.println("Adding few objects in the Arraylist :");
        for(int i = 0; i < 9; i++){
            list.add(i , i);
        }

        System.out.println("The Arraylist converted to String is :");
        System.out.println(list.toString());
        System.out.println();

        System.out.println("Checking if the Arraylist is empty or not : ");

        if(list.isEmpty()){
            System.out.println("ArrayList is empty");
        } else {
            System.out.println("ArrayList is not empty");
        }
        System.out.println();

        System.out.println("Size of ArrayList = "+list.size());
        System.out.println();

        System.out.println("Object removed from front of ArrayList =
"+list.removeFront());
        System.out.println("Size of ArrayList = "+list.size());
        System.out.println();

        System.out.println("Object removed from rear of ArrayList =
"+list.removeRear());
        System.out.println("Size of ArrayList = "+list.size());
        System.out.println();

        System.out.println("The Arraylist converted to String is");
        System.out.println(list.toString());
        System.out.println();
        System.out.println("Adding a few more Objects :");
        System.out.println();
        list.addToFront(9);
        list.addToRear(10);
        list.add(11, 5);
        list.addToRear(8);
        list.addToFront(13);

        System.out.println("The Arraylist converted to String :");
        System.out.println(list.toString());
        System.out.println();

        System.out.println("Index of Object 8 is : "+list.indexOf(8));
        System.out.println();

        System.out.println("Index of Object 25 is : "+list.indexOf(25));
        System.out.println();

        System.out.println("Object at index 8 is = "+list.get(8));
        System.out.println();

        System.out.println("Removed element from index 5 = "
+list.remove(5));
        System.out.println();

        System.out.println(list.toString());
        System.out.println();

        System.out.println("Size of ArrayList = "+list.size());
        System.out.println();

    }
}

class ArrayList <T>{

    private T[] arr;
    private int size;

    public ArrayList(int initialCapacity){
        if(initialCapacity < 1){
            throw new IllegalArgumentException("Initial Capacity must be >=
1");
        }
        arr = (T[])new Object[initialCapacity];
        size = 0;
    }

    public ArrayList(){
        this(10);
    }

    public void addToFront(T obj){
        add(obj, 0);
    }

    public void addToRear(T obj){
        add(obj, size);
    }

    public boolean isEmpty(){
        return size == 0;
    }

    public int size(){
        return size;
    }
    /*

    in the methods 'removeFront' and 'removeRear' the user will not enter
any index
    the user will just call the method and it will remove either the front
(0) index or the rear(size - 1) index
    but if the list is empty, the remove(int index) method will be called
    where checkIndex will be called and it will throw
IndexOutOfBoundsException
    as the user has not entered any index, I do not want to throw
IndexOutOfBoundsException
    so I made a custom exception, 'EmptyListException'

     */

    public T removeFront(){
        checkListNotEmpty();             //EmptyListException is thrown
here, if needed
        T obj = remove(0);
        return obj;
    }

    public T removeRear(){
        checkListNotEmpty();             //EmptyListException is thrown
here, if needed
        T obj = remove(size - 1);
        return obj;
    }

    public void checkIndex(int index){
        if(index < 0 || index > size - 1){
            throw new IndexOutOfBoundsException("Index = "+ index + " Size
= "+size);
        }
    }

    public T get(int index){
        checkIndex(index);
        return arr[index];
    }

    public int indexOf(T obj){
        for(int i = 0; i < size; i++){
            //the 'if' condition used is to prevent any
'NullPointerException'
            if((arr[i] == null && obj == null) || (arr[i] != null &&
arr[i].equals(obj))){
                return i;
            }
        }
        return -1;
    }

    public T remove(int index){
        checkIndex(index);
        T temp = arr[index];
        for(int i = index + 1; i < size; i++){
            arr[i - 1] = arr[i];
        }
        size--;
        arr[size] = null;
        return temp;
    }

    public void add(T obj, int index){
        if(index < 0 || index > size){
            throw new IndexOutOfBoundsException("Index = "+ index +" Size =
"+size);
        }
        //checking capacity
        if(size == arr.length){
            T []newArray = (T[])new Object[2 * size];
            int i = 0;
            for(T o : arr){
                newArray[i++] = o;
            }
            this.arr = newArray;
        }
        //shift elements to right
        for(int i = size; i > index; i--){
            arr[i] = arr[i - 1];
        }
        arr[index] = obj;
        size++;
    }

    public String toString(){
        if(size == 0){
            return "[]";
        }
        StringBuilder sb = new StringBuilder();
        sb.append("[");
        for(int i = 0; i < size - 1; i++){
            sb.append((arr[i] != null ? arr[i].toString() : "null") + ",
");
        }
        sb.append((arr[size - 1] != null ? arr[size - 1].toString() :
"null") + "]");
        return sb.toString();
    }

    public void checkListNotEmpty(){
        if(size == 0){
            throw new EmptyListException();
        }
    }
}

class EmptyListException extends RuntimeException{
    public EmptyListException(){
        super("List is Empty");
    }
}
```

OUTPUT



```
Checking if the initial Arraylist is empty or not :
ArrayList is empty

The Arraylist converted to String is :
[]

Adding few objects in the Arraylist :
The Arraylist converted to String is :
[0, 1, 2, 3, 4, 5, 6, 7, 8]

Checking if the Arraylist is empty or not :
ArrayList is not empty

Size of ArrayList = 9

Object removed from front of ArrayList = 0
Size of ArrayList = 8

Object removed from rear of ArrayList = 8
Size of ArrayList = 7

The Arraylist converted to String is
[1, 2, 3, 4, 5, 6, 7]

Adding a few more Objects :

The Arraylist converted to String :
[13, 9, 1, 2, 3, 4, 11, 5, 6, 7, 10, 8]

Index of Object 8 is : 11

Index of Object 25 is : -1

Object at index 8 is = 6
```



```
Removed element from index 5 = 4

[13, 9, 1, 2, 3, 11, 5, 6, 7, 10, 8]

Size of ArrayList = 11



...Program finished with exit code 0
Press ENTER to exit console.
```

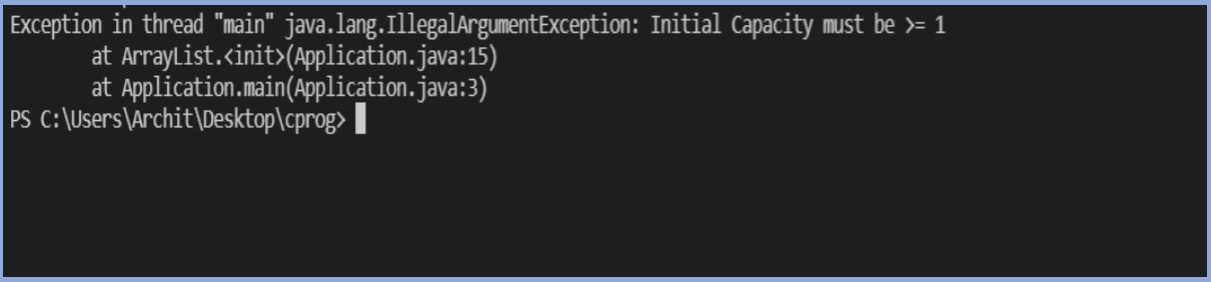
<u>Code for throwing Exceptions -:</u>

The changes required in the code above in order to throw exceptions are to be done in Application class only.

Hence, instead of pasting the whole code again and again, only the class Application code is pasted.

- *<u>For throwing IllegalArgumentException in constructor of ArrayList</u>*

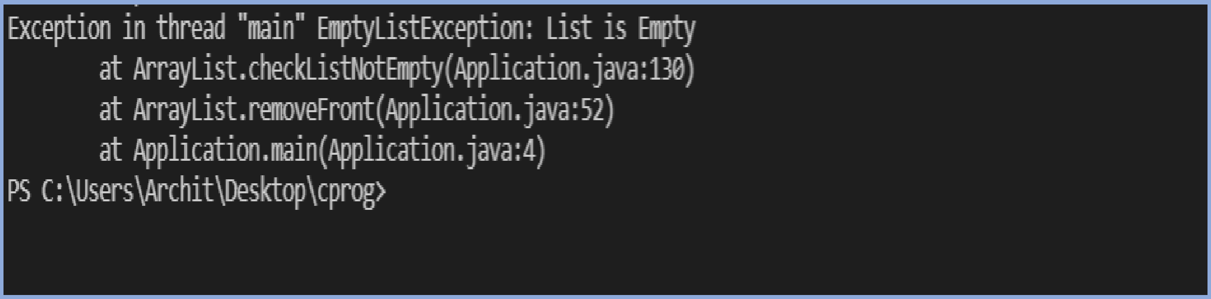
```
public class Application {
    public static void main(String []args){
        ArrayList <Integer> list = new ArrayList<Integer>(0);   // creating
an arraylist with initial capacity of 10

    }
}
```



```
Exception in thread "main" java.lang.IllegalArgumentException: Initial Capacity must be >= 1
        at ArrayList.<init>(Application.java:15)
        at Application.main(Application.java:3)
PS C:\Users\Archit\Desktop\cprog>
```

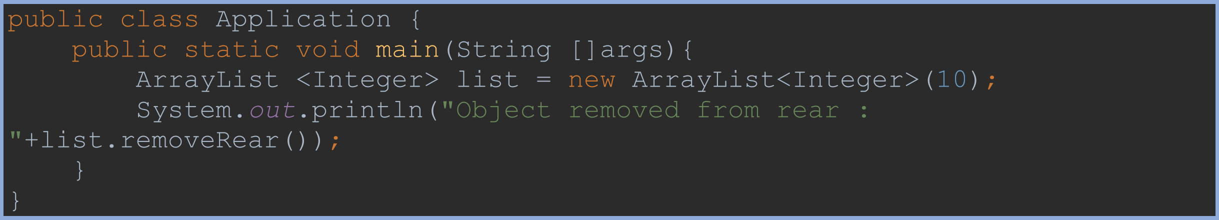
- *<u>For throwing EmptyListException in method removeFront</u>*

```
public class Application {
    public static void main(String []args){
        ArrayList <Integer> list = new ArrayList<Integer>(10);
        System.out.println("Object removed from front :
"+list.removeFront());
    }
}
```

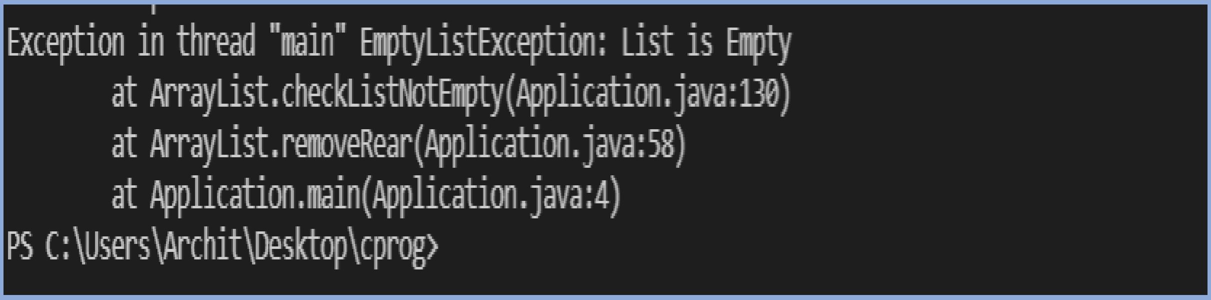


```
Exception in thread "main" EmptyListException: List is Empty
        at ArrayList.checkListNotEmpty(Application.java:130)
        at ArrayList.removeFront(Application.java:52)
        at Application.main(Application.java:4)
PS C:\Users\Archit\Desktop\cprog>
```

- *For throwing EmptyListException in method removeRear*



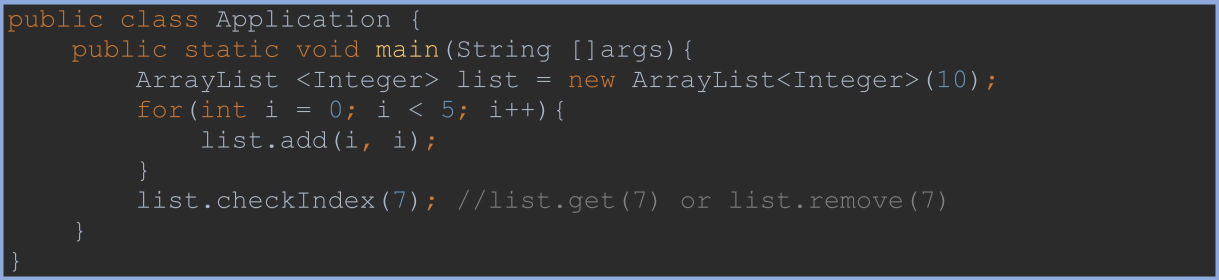
```
public class Application {
    public static void main(String []args){
        ArrayList <Integer> list = new ArrayList<Integer>(10);
        System.out.println("Object removed from rear :
"+list.removeRear());
    }
}
```



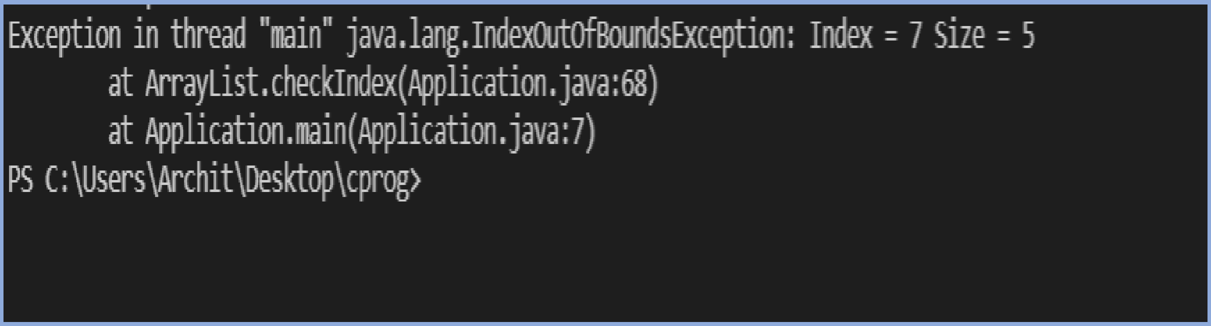
```
Exception in thread "main" EmptyListException: List is Empty
        at ArrayList.checkListNotEmpty(Application.java:130)
        at ArrayList.removeRear(Application.java:58)
        at Application.main(Application.java:4)
PS C:\Users\Archit\Desktop\cprog>
```

- *For throwing IndexOutOfBoundsException in method checkIndex, get, remove*

  Since the method 'get' and 'remove' call the 'checkIndex' method , hence *IndexOutOfBoundsException* will be thrown if any inconsistent index is entered

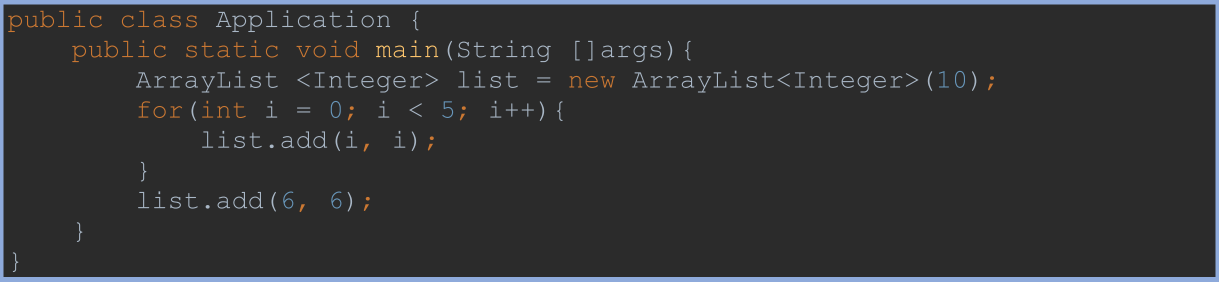


```
public class Application {
    public static void main(String []args){
        ArrayList <Integer> list = new ArrayList<Integer>(10);
        for(int i = 0; i < 5; i++){
            list.add(i, i);
        }
        list.checkIndex(7); //list.get(7) or list.remove(7)
    }
}
```

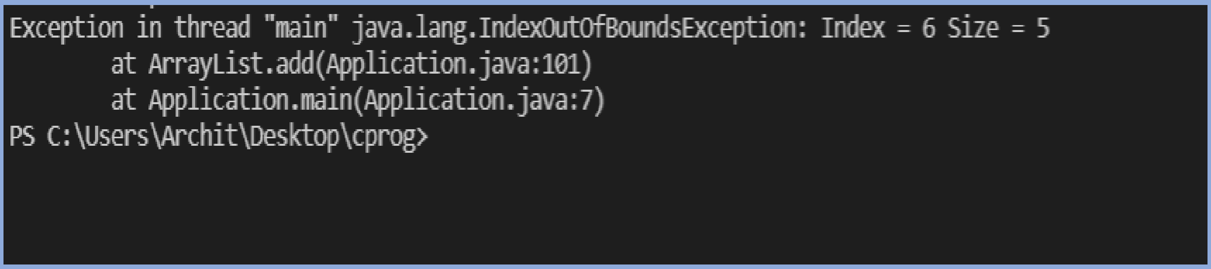


```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index = 7 Size = 5
        at ArrayList.checkIndex(Application.java:68)
        at Application.main(Application.java:7)
PS C:\Users\Archit\Desktop\cprog>
```

- *<u>For throwing IndexOutOfBoundsException in method add</u>*



```
public class Application {
    public static void main(String []args){
        ArrayList <Integer> list = new ArrayList<Integer>(10);
        for(int i = 0; i < 5; i++){
            list.add(i, i);
        }
        list.add(6, 6);
    }
}
```



```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index = 6 Size = 5
        at ArrayList.add(Application.java:101)
        at Application.main(Application.java:7)
PS C:\Users\Archit\Desktop\cprog>
```