Praktikum

Exception Handling

1. Jelaskan mengenai definisi Exception !

2. Semua exception yang berasal dari java.lang.RuntimeException adalah *unchecked exceptions*, sedangkan exception lainnya yang tidak berasal dari java.lang.RuntimeException adalah *checked exceptions*. Jelaskan mengenai *unchecked exceptions* dan *checked exceptions* ,berikan contoh !

3. Buatlah contoh program untuk menangani exception dengan cara menangkap exception seperti di bawah ini :
   - ArithmeticException
   - ArrayStoreException
   - ClassCastException
   - ArrayIndexOutOfBoundsException
   - StringIndexOutOfBoundsException
   - NegativeArraySizeException
   - NoSuchElementException
   - NullPointerException,
   - NumberFormatException

4. Terdapat dua cara untuk menangani Exception yaitu dengan menangkap Exception dan melempar Exception. Lakukan penanganan exception dengan kedua cara tersebut pada program di bawah ini dan berilah penjelasan (apakah program termasuk *unchecked exceptions* atau *checked exceptions*?, exception disebabkan oleh apa?) !

```
public class ReadFile {
     public static void main(String args[]){
        File file = new File("Data.txt");
        BufferedReader fileReader ;

        fileReader = new BufferedReader(new FileReader(file));
        while(true){
           String line = fileReader.readLine();
           if (line == null)
              break ;
           System.out.println(line);
        }
     }
}
```

5. Buatlah sebuah class Stack, FullStackException dan EmptyStackException. Class Stack ini menggambar Stack yang menerapkan konsep LIFO (Last In First Out). Konsep LIFO ini, data yang terakhir masuk akan keluar pertama kali.
   Class Stack mempunyai atribut:
   - size : menentukan besar array untuk menyimpan data. Array berdimensi satu dengan tipe Object.

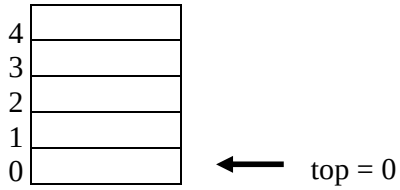
- top : merupakan tanda indeks yang paling atas, yang belum terisi. Sehingga data yang akan masuk akan dimasukkan pada indeks tersebut.
- Object[] elemen : untuk menyimpan data.

Class Stack mempunyai operasi:

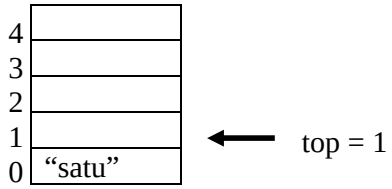
- public Stack() : jika kita membuat object Stack dengan konstruktor Stack() maka tentukan size = 5.
- public Stack(int s) : jika kita membuat object Stack dengan konstruktor Stack(int s) maka tentukan size berdasarkan parameter s.
- public int getSize() : untuk mendapatkan besar array dari Stack.
- public int getTop() : untuk mendapatkan top dari Stack
- public void push(Object o) : untuk memasukkan data ke array pada Stack.
- public Object pop() : untuk mengambil data dari array.

Deskripsi Stack

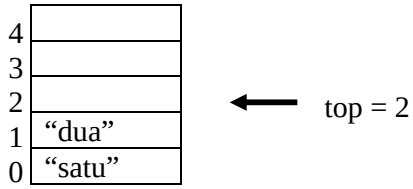
Buat objek Stack dengan nama stack, pertama kali top bernilai 0.



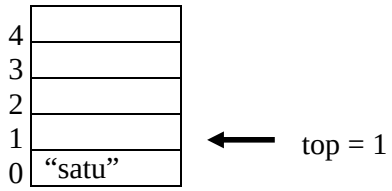
Masukkan sebuah data String “satu”, data “satu” akan dimasukkan pada indeks yang ke-0 (sesuai dengan nilai top) selanjutnya top dinaikkan 1, sehingga top = 1.



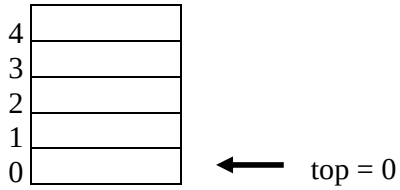
Masukkan sebuah data String “dua”, data “dua” akan dimasukkan pada indeks yang ke-1 (sesuai dengan nilai top) selanjutnya top dinaikkan 1, sehingga top = 2.



Ambil sebuah data dari Stack, data yang diambil adalah data yang dimasukkan terakhir kali (“dua”). Kurangi nilai top dengan 1 sehingga menjadi nilai top = 1. Selanjutnya ambil data pada indeks ke -1.

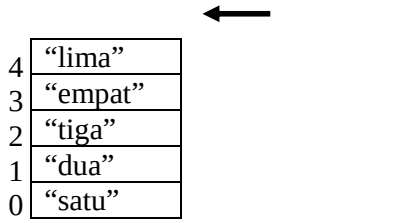


Ambil sebuah data dari Stack, data yang diambil adalah data yang terakhir ("satu"). Kurangi nilai top dengan 1 sehingga menjadi nilai top = 0. Selanjutnya ambil data pada indeks ke -0.

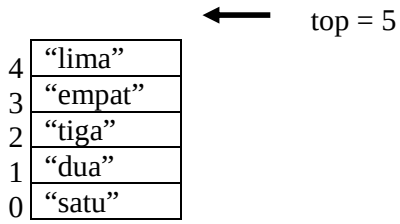


Ambil sebuah data lagi dari Stack, jika kondisi top = 0, maka Stack kosong dan melempar exception FullStackException (class ini merupakan subclass dari class RuntimeException), dan muncul peringatan bahwa "Stack kosong".

Selanjutnya isilah Stack sampai penuh.

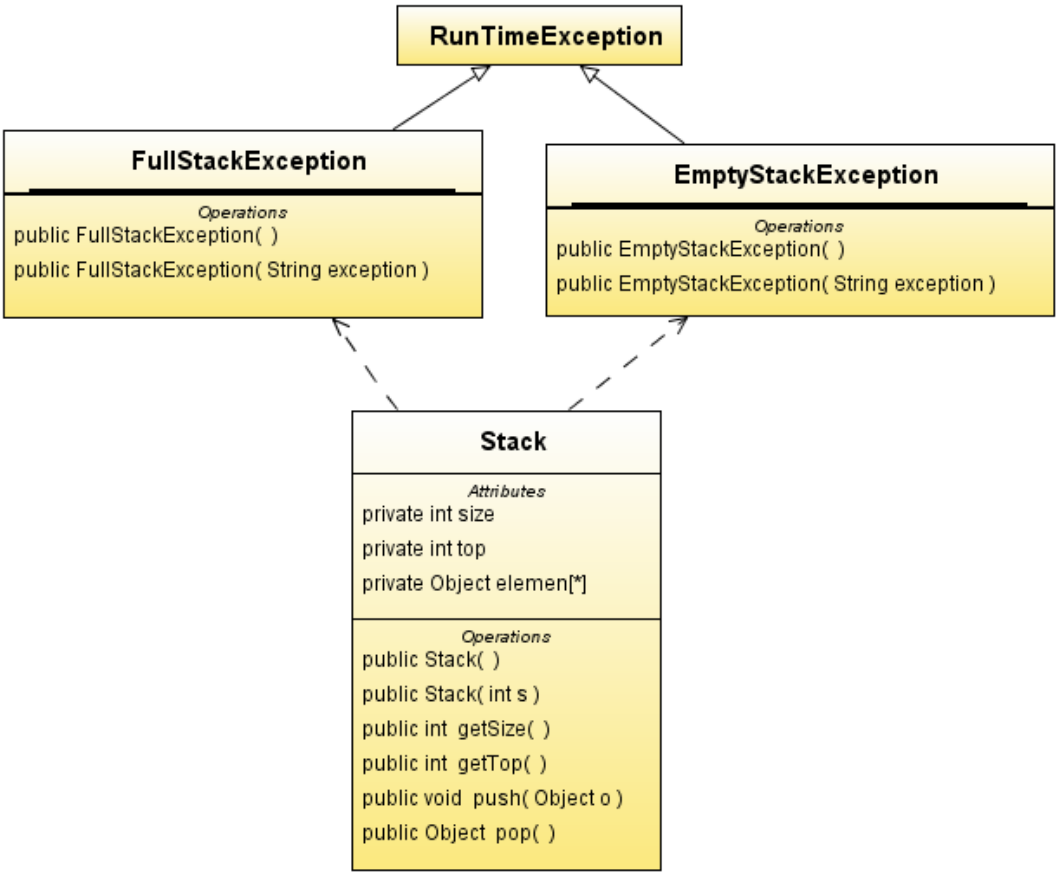


Jika kita ingin memasukkan data baru lagi, jika kondisi top = 5, maka akan melempar exception yaitu EmptyStackException (class ini merupakan subclass dari class RuntimeException).



Masukkan data lagi → Stack Full

Gambar UML Stack



UML 1

Lakukan uji coba dengan fungsi utama :

```
public class TestStack {
    public static void main(String args[]){
        Stack stack = new Stack();
        try{

            stack.push("1");
            stack.push("2");

            System.out.println(stack.pop().toString());
            System.out.println(stack.pop().toString());
            System.out.println(stack.pop().toString());

            stack.push("3");
            stack.push("4");
            System.out.println(stack.pop().toString());
            stack.push("5");
            stack.push("6");
            stack.push("7");
            stack.push("8");
            stack.push("9");

        }catch(EmptyStackException e){
```

```
            System.out.println(e.toString());

        }catch(FullStackException e){
            System.out.println(e.toString());

        }
        finally{
            System.out.println("Yang ini selalu dijalankan");
        }

    }
}
```

```
public class Test {
    Stack stack ;

    public Test(){
        stack = new Stack();
    }

    public Test(int s){
        stack = new Stack(s);
    }

    public void testStackPush(){
        try{
            stack.push("3");
            stack.push("4");
            System.out.println(stack.pop().toString());
            stack.push("5");
            stack.push("6");
            stack.push("7");
            stack.push("8");
            stack.push("9");
        }catch(FullStackException e){
            System.out.println("Full");
            //e.printStackTrace();
        }finally{
            System.out.println("Yang ini selalu dijalankan");
        }
    }
    public void testStackPop(){

        try{

        stack.push("1");
        stack.push("2");

        System.out.println(stack.pop().toString());
        System.out.println(stack.pop().toString());
        System.out.println(stack.pop().toString());

        }catch(EmptyStackException e){
            System.out.println(e.toString());
        }finally{
            System.out.println("Yang ini selalu dijalankan");
        }
    }
    public static void main(String args[]){
        Test t = new Test();
        t.testStackPop() ;
        t.testStackPush();
```

```
        }
    }
```