PILE

CARATTERISTICHE GENERALI)

- Pila di oggetti che possono essere inseriti ed estratti secondo comportamento LIFO
  - **Last In, First Out**
  - possono essere inseriti/estratti/ispezionati solo dalla cima della pila

UTILIZZO DI PILE

- Browser
  - andare alla pagina precedente/successiva
- Editor di testi
  - Control Z (operazione di undo)
- Java Stack
  - utilizzata nella JVM

INTERFACCIA

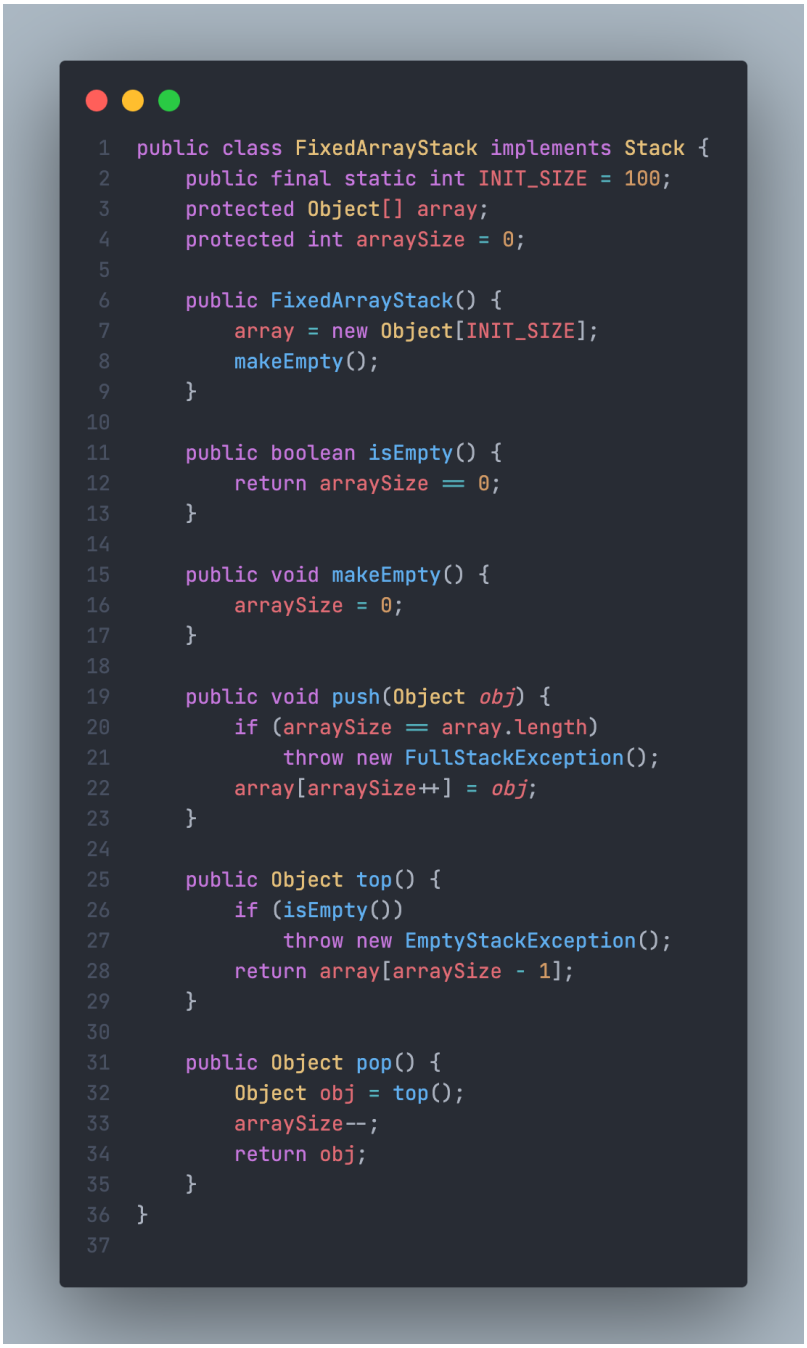
- Definisce le operazioni
  - push: inserisce un oggetto in cima alla pila
  - pop: elimina l'oggetto che si trova in cima alla pila
  - top: ispeziona elemento in cima alla pila

```
public interface Stack extends Container {
    void push(Object obj);
    Object pop();
    Object top();
}
```

REALIZZAZIONE DELLA PILA

- Struttura dati: **array "riempito solo in parte"**
- Definiamo due nuove **eccezioni**
  - `class EmptyStackException extends RuntimeException`
    - pop() su array vuoto
  - `class FullStackException extends RuntimeException`
    - push() su array pieno

- Senza ridimensionamento:

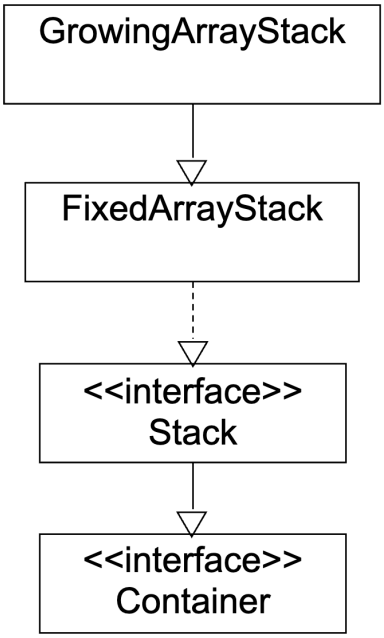


```java
public class FixedArrayStack implements Stack {
    public final static int INIT_SIZE = 100;
    protected Object[] array;
    protected int arraySize = 0;

    public FixedArrayStack() {
        array = new Object[INIT_SIZE];
        makeEmpty();
    }

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

    public void makeEmpty() {
        arraySize = 0;
    }

    public void push(Object obj) {
        if (arraySize == array.length)
            throw new FullStackException();
        array[arraySize++] = obj;
    }

    public Object top() {
        if (isEmpty())
            throw new EmptyStackException();
        return array[arraySize - 1];
    }

    public Object pop() {
        Object obj = top();
        arraySize--;
        return obj;
    }
}

```

- Con Ridimensionamento

```java
public class GrowingArrayStack extends FixedArrayStack {
    public void push(Object obj) {
        if (array.length == arraySize) {
            array = resize(2 * arraySize);
        }
        array[arraySize++] = obj;
    }

    protected Object[] resize(int newLength) // solita tecnica
    {
        if (newLength < array.length)
            throw new IllegalArgumentException();
        Object[] newArray = new Object[newLength];
        System.arraycopy(array, 0, newArray, 0, array.length);
        return newArray;
    }
}
```

- Nel complesso:



ANALISI PRESTAZIONI

- Dipendono dalla **definizione** della **struttura dati** e non dalla sua interfaccia
- FixedArrayStack
  - tempo esecuzione di ogni operazione costante: $O(1)$
- Tempo di esecuzione di GrowingArrayStack
  - unica differenza è `push`

ANALISI AMMORTIZZATA DELLE PRESTAZIONI ASINTOTICHE

- Si applica all'analisi di tempi di esecuzione dei **metodi di inserimento** in strutture dati
- Analisi del tempo di esecuzione medio nel caso **peggiore**
- su `push` con costante **moltiplicativa**
  - (n-1) volte senza resize: $O(1)$
  - n-esima volta: resize $O(n)$
  - $T(n) = \frac{[(n-1)*O(1)+O(n)]}{n} = \frac{O(n)}{n} = O(1)$
- su `push` con costante **addittiva**
  - dimensione diventa $n + k$
  - operazioni lente sono $\frac{n}{k}$ (ogni k elementi devo effettuare un resize) e sono $O(n)$
  - operazioni veloci senza resize sono dunque $n - \frac{n}{k}$
  - sia $n - \frac{n}{k} = \frac{k-1}{k}n = O(n)$ e $\frac{n}{k} = \frac{1}{k}n = O(n)$
  - $T(n) = \frac{(n-\frac{n}{k})*O(1)+(\frac{n}{k})*O(n)}{n} = \frac{O(n)+n*O(n)}{n} = \frac{O(n)}{n} + O(n) = O(1) + O(n) = O(n)$
- Considerazioni generali
  - push ha prestazioni $O(1)$ per qualsiasi costante **moltiplicativa**
  - push ha prestazioni $O(n)$ per qualsiasi costante **addittiva**

Prestazioni in sintesi

- top: $O(1)$
- pop: $O(1)$
- push: $O(1)$ (se growing, usando costante moltiplicativa)

PILE DI DATI FONDAMENTALI

- Trasformare dato fondamentale in oggetto attraverso **classi involucro** (**wrapper**)

```
Integer myIntObj1 = new Integer(2);
Integer myIntObj2 = 2; // sintassi con auto-boxing
int myInt1 = myIntObj1.intValue();
int myInt2 = myIntObj2; // sintassi con auto-unboxing
```