Практическая работа № 22. Абстрактные типы данных. Стек

Цель данной практической работы – научиться разрабатывать программы с абстрактными типами данных на языке Джава и применять паттерн MVC при разработке программ

Теоретические сведения

Стек — это линейная структура данных, которая следует принципу LIFO (Last In First Out) . Стек имеет один конец, куда мы можем добавлять элементя и извлекать их оттуда, в отличие от очереди котрая имеет два конца (спереди и сзади).Стек содержит только один указатель top(верхушка стека), указывающий на самый верхний элемент стека. Всякий раз, когда элемент добавляется в стек, он добавляется на вершину стека, и этот элемент может быть удален только из стека только сверху. Другими словами, стек можно определить как контейнер, в котором вставка и удаление элементов могут выполняться с одного конца, известного как вершина стека.

Примеры стеков – пирамида, стопка тарелок или книг, магазин в пистолете

Стеку присущи следующие характеристики:

- Стек — это абстрактный тип данных с заранее определенной емкостью, что означает, что эта структура данных имеет ограниченный размер, то есть может хранить количество элементов, опредееленное размерностью стека.
- Это структура данных, в которой строго определен порядок вставки и удаления элементов, и этот порядок может быть LIFO или FILO.

Порядок работы со стеком

Рассмотрим пример, допустим стек работает по схеме LIFO. Как видно на рис. ниже, в стеке пять блоков памяти; поэтому размер стека равен 5.

Предположим, мы хотим хранить элементы в стеке, и предположим, что в начале стек пуст. Мы приняли размер стека равным 5, как показано на рис.22.1 ниже, в который мы будем помещать элементы один за другим, пока стек не заполнится.

Поскольку наш стек заполнен, то количество элементов в нем равно 5. В приведенных выше случаях мы можем наблюдать, что он заполняется элементами снизу вверх, при каждом добавлении нового элемента в стек. Стек растет снизу вверх.

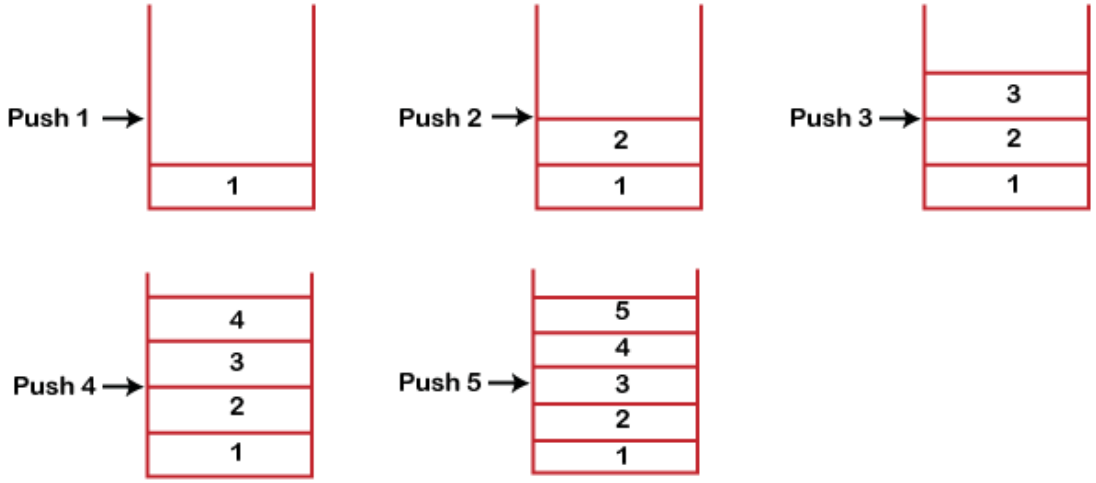

Рисунок 22.1. Пример работы стека из пяти элементов

В приведенном выше случае значение 5 вводится первым, поэтому оно будет удалено только после удаления всех остальных элементов.

Стандартные операции со стеком

Ниже приведены некоторые общие операции, реализованные в стеке:

- push(): когда мы добавляем элемент в стек, эта операция называется push. Если стек заполнен, возникает состояние переполнения.
- pop(): Когда мы удаляем элемент из стека, эта операция называется pop. Если стек пуст, это означает, что в стеке нет элементов, это состояние известно как состояние потери значимости.
- isEmpty(): определяет, пуст стек в настоящий момент или нетили нет.
- isFull(): определяет, заполнен стек или нет.
- peek(): возвращает элемент в заданной позиции.
- count(): возвращает общее количество элементов, доступных в стеке.
- change(): изменяет элемент в заданной позиции.
- display(): печатает все элементы, доступные в стеке.

Опрация push()

Рассмотрим шаги, связанные с выполнением этой операции:

1) Прежде чем вставить элемент в стек, мы проверяем, заполнен ли стек.
2) Если мы пытаемся вставить элемент в стек, а стек уже полон, то возникает условие *переполнения* .

Когда мы инициализируем стек, мы устанавливаем значение верхушки стека как top как -1, чтобы убедиться, что стек пуст.

Когда новый элемент помещается в стек, сначала увеличивается значение вершины, т. е . **top=top+1,** и элемент будет помещен в новую позицию верхушки стека (top). Элементы будут вставляться до тех пор, пока мы не достигнем *максимального* размера стека.

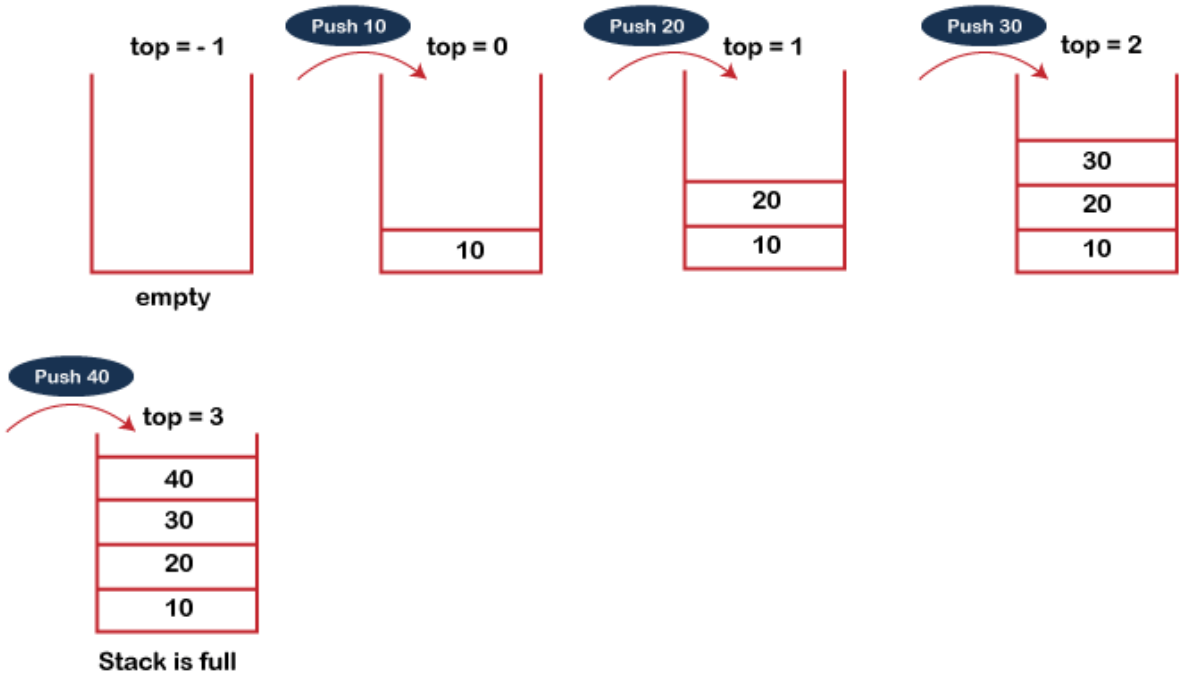

Рисунок 22.2. Пример выполнения операции push() на стеке из пяти элементов

Операция pop()

Рассмотрим шаги, связанные с выполнением этой операции:

1) Перед удалением элемента из стека мы проверяем, не пуст ли стек.
2) Если мы попытаемся удалить элемент из пустого стека, то возникнет состояние потери значимости.
3) Если стек не пуст, мы сначала обращаемся к элементу, на который указывает вершина.

4) После выполнения операции извлечения значение top уменьшается на 1, т . е. top=top-1.

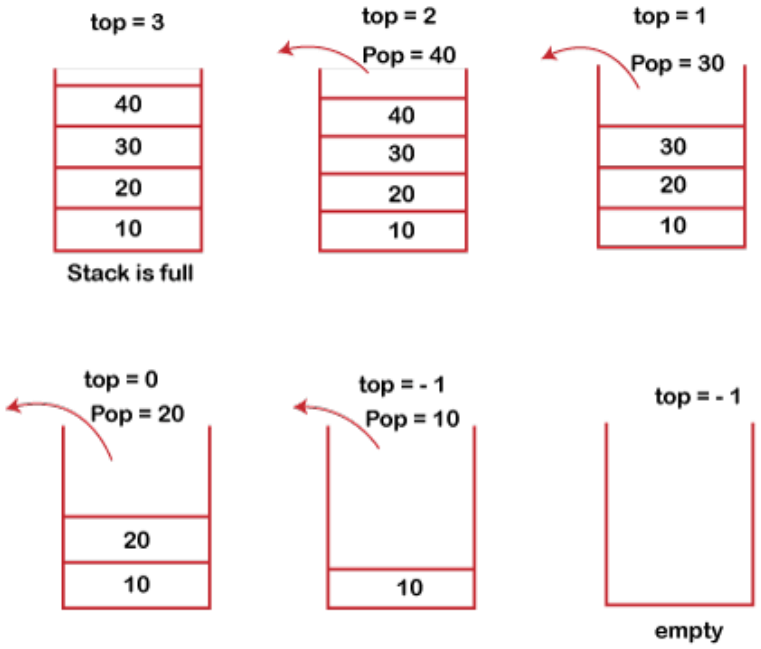

Рисунок 22.3. Пример выполнения операции pop() на стеке из пяти элементов

Пример реализации класса стека представлен на листинге 22.1

Листинг 22.1 – Реализация стека на массиве на языке Джава

```
class Stack {
  // store elements of stack
  private int arr[];
  // represent top of stack
  private int top;
  // total capacity of the stack
  private int capacity;

  // Creating a stack
  Stack(int size) {
    // initialize the array
    // initialize the stack variables
    arr = new int[size];
    capacity = size;
    top = -1;
```

```
}

// push elements to the top of stack
public void push(int x) {
  if (isFull()) {
    System.out.println("Stack OverFlow");

    // terminates the program
    System.exit(1);
  }

  // insert element on top of stack
  System.out.println("Inserting " + x);
  arr[++top] = x;
}

// pop elements from top of stack
public int pop() {

  // if stack is empty
  // no element to pop
  if (isEmpty()) {
    System.out.println("STACK EMPTY");
    // terminates the program
    System.exit(1);
  }

  // pop element from top of stack
  return arr[top--];
}

// return size of the stack
public int getSize() {
  return top + 1;
}
```

```
// check if the stack is empty
public Boolean isEmpty() {
  return top == -1;
}

// check if the stack is full
public Boolean isFull() {
  return top == capacity - 1;
}

// display elements of stack
public void printStack() {
  for (int i = 0; i <= top; i++) {
    System.out.print(arr[i] + ", ");
  }
}

public static void main(String[] args) {
  Stack stack = new Stack(5);

  stack.push(1);
  stack.push(2);
  stack.push(3);

  System.out.print("Stack: ");
  stack.printStack();

  // remove element from stack
  stack.pop();
  System.out.println("\nAfter popping out");
  stack.printStack();
```

Пример реализации стека приведен в приложении А

Язык Джава предоставляет встроенный Stack, который можно использовать для реализации стека.

Листинг 22.2 Реализация стека с помощью Stack

```
import java.util.Stack;

class Main {
  public static void main(String[] args) {

    // create an object of Stack class
    Stack<String> animals= new Stack<>();

    // push elements to top of stack
    animals.push("Dog");
    animals.push("Horse");
    animals.push("Cat");
    System.out.println("Stack: " + animals);

    // pop element from top of stack
    animals.pop();
    System.out.println("Stack after pop: " +
animals);
    }
}
```

В листинге 22.2 означает:

- animals.push() - вставить элементы на вершину стека
- animals.pop() - удалить элемент из вершины стека

Обратите внимание, мы при создании стека так называемая алмазная запись - угловые скобки <String>. Это означает, что стек имеет универсальный тип данных.

Задания на практическую работу №22

Общее заданиие Написать калькулятор для чисел с использованием RPN (Reverse Polish Notation в пер. на русск. яз. - обратной польской записи)

Необходимые сведения об алгоритме

Алгоритм Обратной польской нотации (ОПН) — форма записи математических выражений, в которой операнды расположены перед

знаками операций. Также именуется как обратная польская запись, обратная бесскобочная запись (ОБЗ).

Рассмотрим запись арифметических выражений, в которых сначала следуют два операнда арифметической операции, а затем знак операции. Например:

Обратная польская нотация	**Обычная нотация**
2 3 +	2 + 3
2 3 * 4 5 * +	(2 * 3) + (4 * 5)
2 3 4 5 6 * + - /	2 / (3 - (4 + (5 * 6)))

Нотация записи выражений, представленная в левом столбце таблицы называется обратной польской нотацией (записью) (Reverse Polish Notation, RPN). В теории языков программирования эта нотация называется *постфиксной нотацией*. Обычная нотация называется алгебраической или *инфиксной нотацией* («ин» от англ. *inside*, то есть между операндами).

Есть также префиксная нотация, активно используемая в языке Си (сначала имя функции, а затем её аргументы), а также в языке LISP.

Заметьте, что скобки в обратной польской нотации не нужны. В частности, если во втором примере мы опустим скобки, выражение по прежнему будет интерпретироваться однозначно.

Транслятор RPN-выражений основан на стэке. Каждое следующее число помещается в стэк. Если встречается знак операции (обозначим его *), то два числа из стека извлекаются (a = pop(), b = pop()), для них вычисляется значение соответствующей бинарной арифметической операции, и результат помещается в стек (push(a * b)).

Задание 1. Напишите программу-калькулятор арифметических выражений записанных в обратной польской нотации (RPN-калькулятор).

Задание 2. Напишите графический интерфейс для калькулятора, используя знания полученные ранее при програмиировании GUI с использованием SWING и AWT. Используйте паттерн проектирования MVC. Интерфейс может выглядеть как на рис. 22.1 или как на рис. 22.2

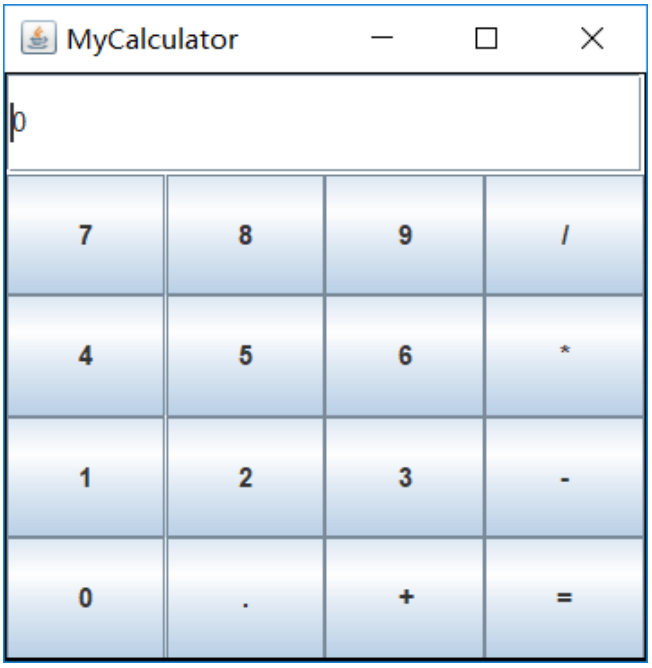

Рисунок 22.4. Общий вид графического интерфейса для программы калькулятора

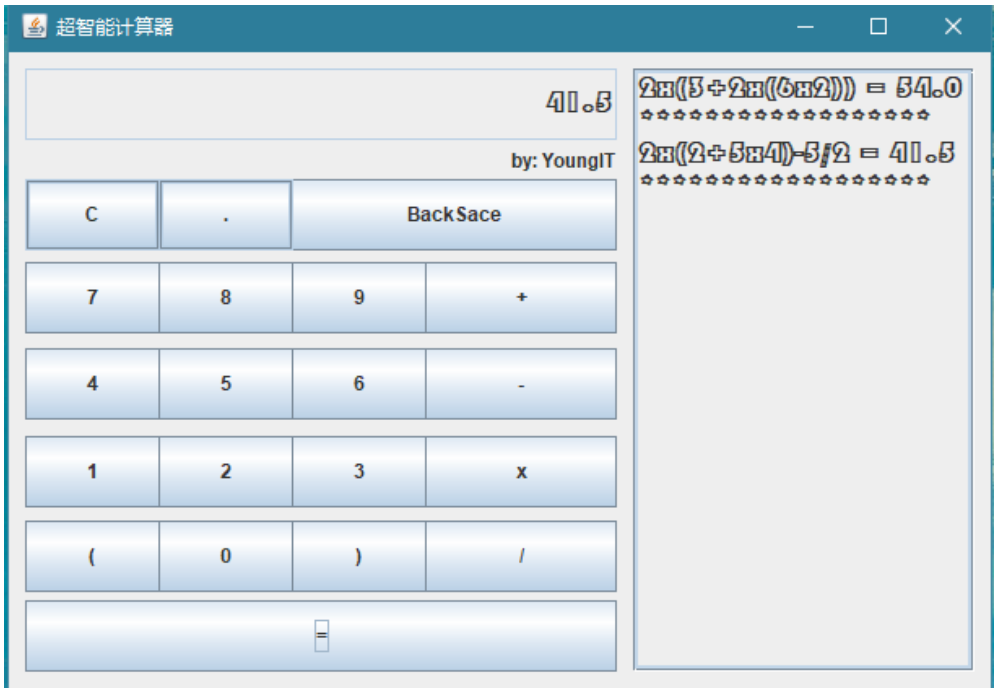

Рисунок 22.5.Общий вид графического интерфейса для прооаммы калькулятора

Задание 2. Постройте систему тестов и проверьте, что ваш калькулятор успешно проходит все тесты и «защищён от дурака» (как дурака-пользователя программы, так и дурака-программиста,

использующего ваш стек и калькулятор). Например, если вводится выражение, в котором число операций превосходит число помещенных в стек элементов (например 1 2 + *), то программа не допустит уменьшения переменной sp до отрицательных значений, а выдаст предупреждение «Невозможно выполнить POP для пустого стека».

Замечание: вы можете выполнить работу можно в двух вариантах (первый вариант проще, второй- труднее).