Стеком (в переводе с английского – stack – стопка; stack — стопка; читается как стэк) называется линейная динамическая структура данных, добавление и исключение элементов в которую и производится с одного конца, называемого вершиной стека. Стек работает по принципу LIFO (Last-In, First-Out) - "поступивший последним, обслуживается первым».

Примеры Стека

Примеры применения стека — любая рекурсивная задача (“так, старую итерацию пока отложу в стопку, а сейчас надо обрабатывать новую итерацию!“), например, перебор маршрутов исследовательского робота в пещере неизвестной конфигурации. Самые первые калькуляторы были напрямую сделаны как стеки. Вместо “2+2” нужно было вводить “2 2 +”. Первые два элемента (“операнды”) клались в стек, пока не будет введён плюс (“оператор”). Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Основными операциями над стеками являются:

• добавление элемента - push()

• удаление элемента - pop()

Абстрактные типы данных (АТД) или структуры данных

Абстрактный тип данных (ADT) является абстракцией структуры данных.

АТД определяет:

• Хранение данных

• Операции выполняемые над данными

• Условия возникновения ошибок, связанных с выполнением операций Пример: АТД для моделирования фондовой биржи

Данные это заказы на покупку / продажу.

Поддерживаются операции

• Заказ на покупку(акций)

• Заказ на продажу(акций)

• Отмена заказа(заказ) Условия возникновения ошибок:

• Купить / продать несуществующие акции

• Отменить несуществующий заказ

АТД стек

В АТД Стеке можно хранить произвольные объекты хранит произвольные объекты. Вставки и удаления следуют за последним элементом в LIFO. Основные операции работы со стеком:

• push(object): вставка (вталкивание) нового элемента в стек.

• object pop(): удаляет из стека (выталкивает) и возвращает последний добавленный элемент. Вспомогательные операции работы со стеком:

• object top(): возвращает последний вставленный элемент, не удаляя его (чтение элемента).

• integer size(): возвращает количество элементов, которые хранятся в стеке.

• boolean isEmpty(): показывает, является ли стек пустым. Интерфейс для Стека в Java
В Java есть интерфейс, соответствующий нашему АТД Stack. Требуется импорт для определения класса. EmptyStackException

Для чего может пригодиться стек? Например, ты создаешь на Java какую-то карточную игру. Колода карт лежит на столе. Отыгранные карты отправляются в сброс. Ты можешь реализовать и колоду, и сброс используя два стека. Игроки берут себе карты с верха колоды — тот же принцип, что и с письмами. Когда игроки отправляют карты в сброс, новые карты ложатся поверх старых. Вот как будет выглядеть первый набросок нашей игры, реализованный на основе стека: **public class** Card {

```
  public Card(String name) {
     this.name = name;
  }

  private String name;

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  @Override
  public String toString() {
     return "Card{" +
           "name='" + name + '\'' +
           '}';
  }
}

import java.util.Stack;

public class SimpleCardGame {

  // колода
  private Stack<Card> deck;
```

```
  //  сброс
  private Stack<Card> graveyard;

  public Card getCardFromDeck() {
     return deck.pop();
  }

  public void discard(Card card) {
     graveyard.push(card);
  }

  public Card lookTopCard() {

     return deck.peek();
  }

  //  ..геттеры, сеттеры и т.д.
} 
```
Как мы и сказали ранее, у нас есть два стека: колода и сброс. Структура данных “стек” реализована в Java в классе **java.util.Stack**. В нашей карточной игре есть 3 метода, описывающие действия игроков:

- взять карту из колоды (метод getCardFromDeck());
- сбросить карту (метод discard());
- посмотреть верхнюю карту(метод lookTopCard()). Допустим, это будет бонусная механика “Разведка“, которая позволит игроку узнать, какая карта следующей попадет в игру.

Внутри наших методов вызываются методы класса Stack:

- **push()** — добавляет элемент на верх стека. Когда мы отправляем карту в сброс, она ложится поверх сброшенных ранее карт;
- **pop()** — удаляет верхний элемент из стека и возвращает его. Этот метод идеально подходит для реализации механики “игрок берет карту”
- **peek()** — возвращает верхний элемент стека, но не удаляет его из стека

Давай посмотрим, как будет работать наша игра:
```
import java.util.Stack;

public class Main3 {

  public static void main(String[] args) {

     //  создаем колоду и добавляем в нее карты
     Stack<Card> deck = new Stack<>();
     deck.push(new Card("Рагнарос"));
     deck.push(new Card("Пират Глазастик"));
     deck.push(new Card("Сильвана Ветрокрылая"));
     deck.push(new Card("Миллхаус Манашторм"));
     deck.push(new Card("Эдвин ван Клифф"));

     //  создаем сброс
```

```
        Stack<Card> graveyard = new Stack<>();

        //  начинаем игру
        SimpleCardGame game = new SimpleCardGame();
        game.setDeck(deck);
        game.setGraveyard(graveyard);

        //  первый игрок берет 3 карты из колоды
        Card card1 = game.getCardFromDeck();
        Card card2 = game.getCardFromDeck();
        Card card3 = game.getCardFromDeck();

        System.out.println("Какие карты достались первому игроку?");
        System.out.println(card1);
        System.out.println(card2);
        System.out.println(card3);

        //  первый игрок отправляет в сброс 3 своих карты
        game.discard(card1);
        game.discard(card2);
        game.discard(card3);

        System.out.println("Какие карты находятся в сбросе?");
        System.out.println(game.getGraveyard().pop());
        System.out.println(game.getGraveyard().pop());
        System.out.println(game.getGraveyard().pop());
    }
```

} Итак, мы добавили в нашу колоду пять карт. Первый игрок взял 3 из них. Какие же карты ему достались? **Вывод в консоль:**

```
Card{name='Эдвин ван Клифф'}
Card{name='Миллхаус Манашторм'}
Card{name='Сильвана Ветрокрылая'}
```

Обрати внимание, в каком порядке карты были выведены в консоль. Карта “Эдвин ван Клифф” в колоду попала последней (пятой по счету), и именно ее игрок взял первой. “Миххлаус” попал в колоду предпоследним, и его игрок взял вторым. “Сильвана” попала в колоду третьей с конца, и досталась игроку третьей. Далее игрок сбрасывает карты. Сначала он сбрасывает Эдвина, потом Миллхауса, потом Сильвану. После чего мы поочередно выводим в консоль карты, которые лежат у нас в сбросе: **Вывод в консоль:**

```
Card{name='Сильвана Ветрокрылая'}
Card{name='Миллхаус Манашторм'}
Card{name='Эдвин ван Клифф'}
```

И снова мы видим как работает стек! Сброс в нашей игре тоже является стеком (как и колода). “Эдвин ван Клифф” был сброшен первым. Вторым был сброшен “Миллхаус Манашторм” — и лег поверх Эдвина в сбросе. Далее была сброшена Сильвана — и эта карта легла уже поверх Миллхауса. Как видишь, ничего сложного в работе стека нет. Тем не менее, знать эту структуру данных необходимо — о ней довольно часто

спрашивают на собеседованиях, а на ее основе нередко строятся более сложных структуры данных.