ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)
Кафедра информационных управляющих систем

ОТЧЁТ

по лабораторной работе №5

по дисциплине «Структура и алгоритмы обработки данных в информационных системах и сетях»

Выполнил: студент группы ИСТ-813, /Кравец А.Ю./

«04» декабря 2020 г. _____/А.Ю. Кравец/

Принял: ст. преподаватель Антонов В.В.

«05» декабря 2020 г. _____/В.В. Антонов/

Санкт-Петербург
2020

Задание: рассмотреть следующие абстрактные типы данных: список, стек, очередь, словарь.

1 Описание абстрактных типов данных

1.1 Список

Список – абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза.

1.2 Стек

Стек – структура данных, в которой доступ к элементам организован по принципу LIFO. Добавление элемента возможно только в вершину стека, удаление элемента возможно только из вершины стека.

1.3 Очередь

Очередь – структура данных с дисциплиной доступа к элементам FIFO. Добавление элемента возможно лишь в конец очереди, выборка – только из начала очереди.

1.4 Словарь

Словарь – абстрактный тип данных, позволяющий хранить пары вида «ключ – значение» и поддерживающий операции добавления пары, а также поиска и удаления пары по ключу.

2 Работа с абстрактными типами данных в Java

2.1 Работа со списком – класс ArrayList

Добавление элементов реализуется двумя функциями:

```
void add(int index, Object element)
```

- Вставляет указанный элемент в указанный индекс позиции в этом списке. Выбрасывает IndexOutOfBoundsException, если указанный индекс выходит за допустимые пределы (index < 0 || index > size()).

```
boolean add(Object o)
```

- Добавляет указанный элемент в конец этого списка.

Удаление элементов реализуется двумя функциями:

```
Object remove(int index)
```

- Удаляет элемент в указанной позиции в этом списке. Вызывает IndexOutOfBoundsException, если индекс выходит за допустимые пределы (index < 0 || index >= size()).

```
void clear()
```

- Удаляет все элементы из этого списка.

Получение элемента по индексу реализуется функцией:

```
Object get(int index)
```

- Возвращает элемент в указанной позиции в этом списке. Вызывает IndexOutOfBoundsException, если указанный индекс выходит за допустимые пределы (index < 0 || index >= size()).

Проверка на пустоту реализуется с помощью метода:

```
boolean isEmpty()
```

- Возвращает значение true, если список не содержит элементов

2.2 Работа со стеком – класс Stack

Добавление элемента реализуется функцией:

```
Object push(Object element)
```

- Вталкивает элемент в стек. Элемент также возвращается.

Удаление элементов реализуется двумя функциями:

```
Object pop()
```

- Возвращает элемент, находящийся в верхней части стека, удаляя его в процессе.

```
void clear()
```

- Удаляет все элементы из этого стека.

Получение верхнего элемента стека реализуется двумя функциями:

```
Object pop()
```

- Возвращает элемент, находящийся в верхней части стека, удаляя его в процессе.

```
Object peek()
```

- Возвращает элемент, находящийся в верхней части стека, но не удаляет его.

Проверка на пустоту реализуется с помощью метода:

```
boolean empty()
```

- Проверяет, является ли стек пустым. Возвращает true, если стек пустой. Возвращает false, если стек содержит элементы.

2.3 Работа с очередью – класс LinkedList

Добавление элемента реализуется функцией:

```
boolean offer(Object obj)
```

- Добавляет элемент obj в конец очереди. Если элемент удачно добавлен, возвращает true, иначе - false

Удаление элементов реализуется тремя функциями:

```
Object poll()
```

- Возвращает с удалением элемент из начала очереди. Если очередь пуста, возвращает значение null

```
Object remove()
```

- Возвращает с удалением элемент из начала очереди. Если очередь пуста, генерирует исключение NoSuchElementException

```
void clear()
```

- Удаляет все элементы из этой очереди

Получение первого элемента реализуется функциями:

```
Object peek()
```

- Возвращает без удаления элемент из начала очереди. Если очередь пуста, возвращает значение null

```
Object poll()
```

- Возвращает с удалением элемент из начала очереди. Если очередь пуста, возвращает значение null

```
Object remove()
```

- Возвращает с удалением элемент из начала очереди. Если очередь пуста, генерирует исключение NoSuchElementException

Проверка на пустоту реализуется с помощью метода:

```
boolean isEmpty()
```

- Возвращает значение true, если очередь не содержит элементов

2.4 Работа со словарем – класс HashMap

Добавление элемента реализуется функцией:

```
Object put(Object key, Object value)
```

- Связывает указанное значение с указанным ключом на этом Map.

Удаление элементов реализуется двумя функциями:

```
Object remove(Object key)
```

- Удаляет отображение для этого ключа с этого Map, если присутствует.

```
void clear()
```

- Удаляет все соответствия с этого словаря.

Получение элемента по индексу реализуется функцией:

```
Object get(Object key)
```

- Возвращает значение, для которого указанный ключ отображается в этой хэш-карте идентификатора, или null (нуль), если Map не содержит отображения для этого ключа.

Обход пар «ключ-значение» словаря реализуется через итератор:

Инициализация итератора:

```
Iterator iter = map.entrySet().iterator()
```

Получение следующего значения итератора : `iter.next()`

Проверка наличия элементов в итераторе: `iter.hasNext()`

Пара «ключ-значение» словаря инициализируется как:

```
Map.Entry pair
```

- *Обращение к ключу пары*: pair.getKey()
- *Обращение к значению пары*: pair.getValue()

Проверка на пустоту реализуется с помощью метода:

`boolean isEmpty()`

- Возвращает true, если этот словарь не содержит отображений значений ключа.

Вывод:

В данной работе были рассмотрены следующие абстрактные типы данных: список, стек, очередь, словарь.