Laboratorium 8

Szymon Kałuża

2022-05-10

1 Implementacja listy

```
public class ArrOne{
    private Integer[] arr;
    public ArrOne(int value){
        Integer[] vv = new Integer[1];
        vv[0] = value;
        this.arr = vv;
    }
    public void add(int value){
        Integer[] val = new Integer[arr.length + 1];
        for(int j = 0; j < arr.length; j++){
            val[j] = arr[j];
        }
        val[arr.length] = value;
        this.arr = val;
    }
    public void removeTail(){
        Integer[] val = new Integer[arr.length - 1];
        for(int j = 0; j < arr.length - 1; j++){
            val[j] = arr[j];
        }
        this.arr = val;
    }
    public int naiveSearch(int x){
        for(int j = 0; j < arr.length; j++){
            if(arr[j] == x) return j;
        }
        return -1;
    }
}
```

2 Implementacja stosu i kolejki

2.1 Stos

```
class MyStack{
    public Integer[] quess;

    public MyStack(int value){
        Integer[] val = new Integer[1];
        val[0] = value;
        this.quess = val;
    }
    public void add(int value){
        Integer[] val = new Integer[quess.length + 1];
        for(int j = 0; j < quess.length; j++){
            val[j] = quess[j];
        }
        val[quess.length] = value;
        this.quess = val;
    }

    public void remove(){
        Integer[] val = new Integer[quess.length - 1];
        for(int j = 0; j < quess.length - 1; j++){
            val[j] = quess[j];
        }
        this.quess = val;
    }

    public void show(){
        for(int j = 0; j < quess.length; j++){
```

```
            System.out.println(quess[j] + ", ");
        }
    }
}
```

2.2 Kolejka

```
class Ques{
    public Integer[] quess;

    public Ques(int value){
        Integer[] val = new Integer[1];
        val[0] = value;
        this.quess = val;
    }
    public void add(int value){
        Integer[] val = new Integer[quess.length + 1];
        for(int j = 0; j < quess.length; j++){
            val[j] = quess[j];
        }
        val[quess.length] = value;
        this.quess = val;
    }

    public void remove(){
        Integer[] val = new Integer[quess.length - 1];
        for(int j = 1; j < quess.length; j++){
            val[j - 1] = quess[j];
        }
        this.quess = val;
    }

    public void show(){
        for(int j = 0; j < quess.length; j++){
            System.out.println(quess[j] + ", ");
        }
    }
}
```

3 Implementacja dodatkowych metod do listy

3.1 Dodawanie elementu

```
public void add(int value){
        Integer[] val = new Integer[quess.length + 1];
        for(int j = 0; j < quess.length; j++){
            val[j] = quess[j];
        }
        val[quess.length] = value;
        this.quess = val;
    }
```

3.2 Odwracanie listy

```
public void reverse(){
        int temp;
        for(int j = 0; j < arr.length/2; j++){
            temp = arr[j];
            arr[j] = arr[arr.length - (j + 1)];
            arr[arr.length - (j + 1)] = temp;
        }
}
```

3.3 Odwracanie wszystkich kluczy o wartości key

```
public void removeKey(int key){
        for(int j = 0; j < arr.length; j++){
            if(arr[j] == key){
                Integer[] val = new Integer[arr.length - 1];
                for(int i = 0, k = 0; i < arr.length - 1; i++, k++){
                    if(i == j){
                        k++;
```

```
                    }
                    val[i] = arr[k];
                }
                this.arr = val;
            }
        }
}
```

3.4 Wstawianie elementu w określone miejsce na liście

```
public void insertKeyByIndex(int key, int index){
        Integer[] val = new Integer[arr.length + 1];
        for(int j = 0, k = 0; j < arr.length; j++, k++){
            if(j == index){
                val[k] = key;
                k++;
            }
            val[k] = arr[j];
        }
        this.arr = val;
}
```

4 Praca na zbiorach

4.1 Czy a należy do zbioru A

```
public boolean isAnElementOf(ArrOne arrOne){
        if(arr.length > arrOne.getNumberOfElements()) return false;
        boolean ps = false;
        for(int j = 0; j < arr.length; j++){
            ps = false;
            for(int i = 0; i < arrOne.getNumberOfElements(); i++){
                if(arrOne.getElementByIndex(i) == arr[j]) ps = true;
            }
            if(!ps) return false;
        }
        return true;
}
```

4.2 Tworzenie listy składającej się z części wspólnej zbiorów a i A

```
public Integer[] intersectionWith(ArrOne arrOne){
        Integer[] val = new Integer[0];
        for(int j = 0; j < arr.length; j++){
            for(int i = 0; i < arrOne.getNumberOfElements(); i++){
                if(arr[j] == arrOne.getElementByIndex(i)){
                    Integer[] temp = new Integer[val.length + 1];
                    for(int k = 0; k < val.length; k++){
                        temp[k] = val[k];
                    }
                    temp[val.length] = arr[j];
                    val = temp;
                }
            }
        }
        return val;
}
```

4.3 Tworzenie listy będącej sumą zbiorów a i A

```
public Integer[] sumWith(ArrOne arrOne){
        Integer[] val = new Integer[arr.length + arrOne.getNumberOfElements()];
        for(int j = 0; j < arr.length; j++){
            val[j] = arr[j];
        }
        for(int j = arr.length, k = 0; j < arr.length + arrOne.getNumberOfElements(); j++, k++){
            val[j] = arrOne.getElementByIndex(k);
        }
        return val;
}
```

4.4 Sprawdzanie czy a jest podzbiorem zbioru A

```
public boolean isASubsetOf(ArrOne arrOne){
        if(arr.length > arrOne.getNumberOfElements()) return false;
        int number = 0;
        for(int j = 0; j < arr.length; j++){
            for(int i = 0; i < arrOne.getNumberOfElements(); i++){
                if(arr[j] == arrOne.getElementByIndex(i)){
                    number++;
                }
            }
        }
        if(number == arr.length) return true;
        else return false;
}
```