TestHashSet — копия.java

```
package src.main.java.test;

import src.main.java.base.MyEasyHashSet;
import src.main.java.base.MyHardHashSet;

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

        boolean flg;

        MyEasyHashSet<Integer> myEasyHashSet = new MyEasyHashSet<>();

        System.out.println("Массив:"+myEasyHashSet.getList());
// Пустой список                    #Массив:[]
        flg = myEasyHashSet.add(1);
        System.out.println("Массив:"+myEasyHashSet.getList()+" Результат
операции (add(1)):"+flg);    // Добавили единицу                 #Массив:[1]
Результат операции (add(1)):true
        flg = myEasyHashSet.add(2);
        System.out.println("Массив:"+myEasyHashSet.getList()+" Результат
операции (add(2)):"+flg);    // Добавили двойку                  #Массив:[1,
2] Результат операции (add(2)):true
        flg = myEasyHashSet.add(1);
        System.out.println("Массив:"+myEasyHashSet.getList()+" Результат
операции (add(1)):"+flg);    // Пробуем добавить единицу еще раз #Массив:[1,
2] Результат операции (add(1)):false
        flg = myEasyHashSet.remove(2);
        System.out.println("Массив:"+myEasyHashSet.getList()+" Результат
операции (remove(2)):"+flg); // Пробуем удалить двойку           #Массив:[1]
Результат операции (remove(2)):true
        flg = myEasyHashSet.remove(2);
        System.out.println("Массив:"+myEasyHashSet.getList()+" Результат
операции (remove(2)):"+flg); // Пробуем удалить двойку еще раз   #Массив:[1]
Результат операции (remove(2)):false
        flg = myEasyHashSet.remove(1);
        System.out.println("Массив:"+myEasyHashSet.getList()+" Результат
операции (remove(1)):"+flg); // Пробуем удалить двойку еще раз   #Массив:[]
Результат операции (remove(2)):true
        flg = myEasyHashSet.remove(1);
        System.out.println("Массив:"+myEasyHashSet.getList()+" Результат
операции (remove(1)):"+flg); // Пробуем удалить двойку еще раз   #Массив:[]
Результат операции (remove(2)):false

        System.out.println("\n\n");

        MyHardHashSet<Integer> myHardHashSet = new MyHardHashSet<>();

        System.out.println("Массив:"+myHardHashSet.strmas());
// Пустой список                    #Массив:[]
        flg = myHardHashSet.add(1);
        System.out.println("Массив:"+myHardHashSet.strmas()+" Результат
операции (add(1)):"+flg);    // Добавили единицу                 #Массив:[1]
Результат операции (add(1)):true
        flg = myHardHashSet.add(2);
        System.out.println("Массив:"+myHardHashSet.strmas()+" Результат
операции (add(2)):"+flg);    // Добавили двойку                  #Массив:[1,
2] Результат операции (add(2)):true
        flg = myHardHashSet.add(1);
        System.out.println("Массив:"+myHardHashSet.strmas()+" Результат
операции (add(1)):"+flg);    // Пробуем добавить единицу еще раз #Массив:[1,
2] Результат операции (add(1)):false
        flg = myHardHashSet.remove(2);
```

```
        System.out.println("Массив:"+myHardHashSet.strmas()+" Результат
операции (remove(2)):"+flg); // Пробуем удалить двойку           #Массив:[1]
Результат операции (remove(2)):true
        flg = myHardHashSet.remove(2);
        System.out.println("Массив:"+myHardHashSet.strmas()+" Результат
операции (remove(2)):"+flg); // Пробуем удалить двойку еще раз   #Массив:[1]
Результат операции (remove(2)):false
        flg = myHardHashSet.remove(1);
        System.out.println("Массив:"+myHardHashSet.strmas()+" Результат
операции (remove(1)):"+flg); // Пробуем удалить двойку еще раз   #Массив:[]
Результат операции (remove(2)):true
        flg = myHardHashSet.remove(1);
        System.out.println("Массив:"+myHardHashSet.strmas()+" Результат
операции (remove(1)):"+flg); // Пробуем удалить двойку еще раз   #Массив:[]
Результат операции (remove(2)):false



    }
}
```

TestLinkedList — копия.java

```
package src.main.java.test;

import java.util.List;
import src.main.java.base.MyLinkedList;

public class TestLinkedList {
    public static void main(String[] arg){

        boolean flg;
        MyLinkedList<Integer> myLinkedList = new MyLinkedList<>();
        List<Integer> list = List.of(7,8,9);
        List<Integer> list_empty = List.of();

        System.out.println(myLinkedList.strLinkedList());
//Пустой лист                       #null
        flg = myLinkedList.add(1);
        System.out.println(myLinkedList.strLinkedList()+" \t\t add 1\treturn:
"+flg);       //Добавили единицу                  #[1]->null
add 1  return: true
        flg = myLinkedList.add(2);
        System.out.println(myLinkedList.strLinkedList()+" \t\t add 2\treturn:
"+flg);       //Добавили два                      #[1]->[2]->null
add 2  return: true
        flg = myLinkedList.add(3);
        System.out.println(myLinkedList.strLinkedList()+" \t add 3\treturn:
"+flg);         //Добавили три                      #[1]->[2]->[3]->null
add 3  return: true
        flg = myLinkedList.add(1);
        System.out.println(myLinkedList.strLinkedList()+" add 1\treturn:
"+flg);            //Добавили единицу еще раз         #[1]->[2]->[3]->[1]-
>null        add 1  return: true
        flg = myLinkedList.remove(2);
        System.out.println(myLinkedList.strLinkedList()+" \t rem 2\treturn:
"+flg);         //Убрали двойку                     #[1]->[3]->[1]->null
rem 2  return: true
        flg = myLinkedList.remove(1);
        System.out.println(myLinkedList.strLinkedList()+" \t\t rem 1\treturn:
"+flg);       //Убрали единицу                    #[3]->[1]->null
rem 1  return: true
        flg = myLinkedList.remove(1);
        System.out.println(myLinkedList.strLinkedList()+" \t\t rem 1\treturn:
"+flg);       //Убрали вторую единицу             #[3]->null
rem 1  return: true
        flg = myLinkedList.remove(3);
        System.out.println(myLinkedList.strLinkedList()+" \t\t\t rem
3\treturn: "+flg);     //Убрали тройку                     #null
rem 3  return: true
        flg = myLinkedList.remove(3);
        System.out.println(myLinkedList.strLinkedList()+" \t\t\t rem
3\treturn:"+flg);     //Пробуем убрать тройку еще раз     #null
rem 3  return:false
        flg = myLinkedList.addAll(list);
        System.out.println(myLinkedList.strLinkedList()+" \t addAll 7,8,9
return:"+flg);  //Добавим массив из трех элементов  #[7]->[8]->[9]->null
addAll 7,8,9  return:true
        flg = myLinkedList.addAll(list_empty);
        System.out.println(myLinkedList.strLinkedList()+" \t addAll empty
return:"+flg);  //Пробуем добавить пустой массив    #[7]->[8]->[9]->null
addAll 7,8,9  return:false
    }
```

```
}
```

TestStudent — копия.java

```java
package src.main.java.test;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Random;
import src.main.java.base.Book;
import src.main.java.base.Student;

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


            Random rand = new Random();
            List<Student> sList = new ArrayList<>();
            for (int i=0,j=rand.nextInt(20)+20;i<j;i++){
                sList.add(new Student());
            }

            //Начало стрима

            sList.stream()
            .peek(student -> System.out.println(student))
//выведим имя студента и количество книг у данного студента(пока что 0)
            .peek(Student::add_books)
//выдадим каждому студенту список книг
            //.peek(student -> System.out.println(student))
//убедимся что книги получены
            .flatMap(student -> student.getBooklist().stream())
//берем все книги у всех студентов
            .distinct()
//оставляем только уникальные книги
            .sorted(Comparator.comparing(Book::getPages))
//сортируем книги по страницам
            .toList()
//складываем все книги в один List
            .stream()
//делаем стрим с новым классом Book
            .filter(book -> book.getYear()>2000)
//ограничиваем показ книг только для тех у кого год выпуска меньше 2000
            .limit(3)
//ограничиваем показ книг до трех
            //.peek(book -> System.out.println(book))
//можно посмотреть информацию о найденной книге
            .map(Book::getYear)
//насколько я понял, это метод короткого замыкания
            .findFirst()
//у первой найденной книги берем getYear
            .map(year -> "Год выпуска: " + year)
//воспринимаем результат getYear как year
            .ifPresentOrElse(System.out::println,() ->
System.out.println("Книга отсутствует")); //не знаю как проверить, но код
рабочий



//Пример вывода


//Student A books:0
```

```
//Student I books:0

//Student G books:0

//Student D books:0

//Student H books:0

//Student D books:0

//Student F books:0

//Student G books:0

//Student G books:0

//Student I books:0

//Student J books:0

//Student B books:0

//Student A books:0

//book3 Author: H.G Year: 2005 Pages: 20 Optional: Made in Russia Federation (строчка закоментирована, показывать не будет) //Optional добавил до того как прочитать про короткое замыкание

//Год выпуска: 2005




    }
}
```

MyHardHashSet — копия.java

```java
package src.main.java.base;

import java.util.Arrays;

/* Допустим что я не знаю что такое Node и не знаю что HashSet написан с помощью HashMap.
 * Попробуем реализовать коллекцию с помощью простого массива. Для этого добавим дополнительные
 * функции увеличения и уменьшения массива.
 */




public class MyHardHashSet<T>{

    private T[] mas = (T[]) new Object[0];

    private void resize(){
        mas = Arrays.copyOf(mas, mas.length+1);
    }

    private void decreasize()
    {
        mas = Arrays.copyOf(mas, mas.length-1);
    }

    public boolean add(T value){

        for (T val : mas) {
            if (val == value) {
                return false;
            }
        }
        resize();
        mas[mas.length-1] = value;
        return true;
    }

    public boolean remove(T value){
        for (int i = 0;i<mas.length;i++)
        {
            if (mas[i] == value){
                mas[i] = mas[mas.length-1];
                decreasize();
                return true;
            }
        }
        return false;
    }

    public StringBuilder strmas() {                        //Для тестов

        StringBuilder ret = new StringBuilder("[");

        if (mas.length>=1)
        {

        for (int i = 0;i<mas.length-1;i++)
```

```
            {
                ret.append(mas[i].toString()).append(", ");
            }
            ret.append(mas[mas.length-1].toString()).append("]");
            return ret;
            }
            else {
                ret.append("]");
                return ret;
            }
        }

}
```

MyEasyHashSet — копия.java

```java
package src.main.java.base;

import java.util.LinkedList;


/*
 * Set - неупорядоченный набор уникальных значений,
 *  добавим список в поле, который будем контролировать.
 *  Да, можно получить список и работать с ним как с списком,
 *  но изменения не будут влиять на сам объект MyEasyHashSet,
 *  так как создается новый объект LinkedList, поэтому
 *  сам объект можно воспринимать как аналог HashSet
 *
 *  Это простая версия аналога HashSet
 *
 */

public class MyEasyHashSet<T>{

    private final LinkedList<T> list = new LinkedList<>();

    public boolean add(T value){
        return list.contains(value)?false:list.add(value);
    }

    public boolean remove(T value){
        return list.contains(value)?list.remove(value):false;
    }

    public LinkedList<T> getList() {                    //Для тестов
        return new LinkedList<>(list);
    }

}
```

MyLinkedList — копия.java

```
package src.main.java.base;

import java.util.Collection;


public class MyLinkedList<T> {

    private Node<T> start = null;

    public boolean add(T value){
        if (start == null){
            start = new Node<>(value);
        }
        else{
            Node<T> curr = start;
            while (true) {
                if (curr.next == null){
                    curr.next = new Node<>(value);
                            break;
                }
                curr = curr.next;
            }
        }
        return true;
    }

    public boolean addAll(Collection<T> collection){
        if (collection.isEmpty()){
            return false;
        }
        for (T elem : collection) {
            if (!add(elem)){
                return false;
            }
        }
        return true;
    }


    public boolean remove(T value){
        Node<T> curr = start;
        if (start == null) return false;
        if (curr.value == value){
            this.start = curr.next;
            return true;
        }
        while (curr.next!=null)
        {
            if (curr.next.value == value){
                curr.next = curr.next.next;
                return true;
            }
            curr = curr.next;
        }
        return false;
    }

    public StringBuilder strLinkedList(){
        StringBuilder ret = new StringBuilder("");
        Node<T> curr = start;
```

```
        if (curr!=null)
        {
            ret.append("[").append(curr.value).append("]->");
        }
        else {
            ret.append("null");
            return ret;
        }
        while (curr.next!=null)
        {
            curr = curr.next;
            ret.append("[").append(curr.value).append("]").append("->");
        }
        ret.append("null");
        return ret;
    }

}

class Node<T>{

    protected T value;
    Node<T> next = null;

    Node(T value){
        this.value=value;
    }

}
```

Student — копия.java

```java
package src.main.java.base;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Student {
    String name;
    private List<Book> booklist = new ArrayList<>();

    public Student() {

        List<String> namelist =
List.of("A","B","C","D","E","F","G","H","I","J");
        Random rand = new Random();

        setName("Student "+namelist.get(rand.nextInt(namelist.size())));
    }

    public Student add_books(){
        Random rand = new Random();
        for (int i=0,j=rand.nextInt(5)+5;i<j;i++){
            booklist.add(new Book());
        }
        return this;
    }



    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public String toString(){
        return name+" books:"+booklist.size();
    }

    public List<Book> getBooklist() {
        return List.copyOf(booklist);
    }

}
```

Book — копия.java

```java
package src.main.java.base;

import java.util.List;
import java.util.Random;


public class Book {
    String name;
    String author;
    Integer year;
    Integer pages;
    String optional;


    Book(){

        Random rand = new Random();

        List<String> authorlist=
List.of("A","B","C","D","E","F","G","H","I","J");

        setName("book"+rand.nextInt(10));

setAuthor(authorlist.get(rand.nextInt(authorlist.size()))+"."+authorlist.get(
rand.nextInt(authorlist.size())));
        setPages(rand.nextInt(190)+10);

        int y = rand.nextInt(75)+1950;
        setYear(y);
        setOptional(y>=1991?"Made in Russia Federation":"Сделано в СССР");
    }

    public String getName() {
        return name;
    }

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

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public Integer getYear() {
        return year;
    }

    public void setYear(Integer year) {
        this.year = year;
    }

    public Integer getPages() {
        return pages;
    }
```

```
    public void setPages(Integer pages) {
        this.pages = pages;
    }

    public String getOptional() {
        return optional;
    }

    public void setOptional(String optional) {
        this.optional = optional;
    }

    @Override
    public String toString(){
        return name+" Author: "+author+" Year: "+year+" Pages: "+pages+"
Optional: "+optional;
    }
}
```

Ссылка на GitHub:
https://github.com/SokIvan/First_Homework