**Main**

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or  
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.  
import HomeWork1.Ex1.MyArrayList;  
import HomeWork1.Ex1.MyHashSet;  
import HomeWork1.Ex2.Book;  
import HomeWork1.Ex2.Student;  
  
void main() {  
 CheckMyArrayList();  
 CheckMyHashSet();  
 List<Student> students = List.*of*(  
 new Student("Alex", List.*of*(  
 new Book("Java", 100, 1999),  
 new Book("Spring", 300, 2005),  
 new Book("Algorithms", 450, 2001),  
 new Book("Patterns", 350, 2003),  
 new Book("Docker", 200, 2020)  
 )),  
 new Student("Maria", List.*of*(  
 new Book("Java", 100, 1999),  
 new Book("C#", 150, 2007),  
 new Book("C", 300, 2005),  
 new Book("Networks", 400, 2018),  
 new Book("Docker", 200, 2020)  
 ))  
 );  
  
 students.stream()  
 .peek(System.*out*::println) // вывести студентов  
 .flatMap(s -> s.getBooks().stream()) // все книги  
 .sorted(Comparator.*comparingInt*(Book::getPages)) // сортировка по страницам  
 .distinct() // уникальные книги  
 .filter(b -> b.getYear() > 2000) // книги после 2000  
 .limit(3) // первые 3  
 .map(Book::getYear) // взять годы выпуска  
 .findFirst() // короткое замыкание -> Optional<Integer>  
 .ifPresentOrElse( // вывести значение  
 year -> System.*out*.println("Год выпуска: " + year),  
 () -> System.*out*.println("Книга не найдена")  
 );  
}  
  
void CheckMyArrayList(){  
 MyArrayList<String> List = new MyArrayList<String>(3);  
 List.add("a0");  
 List.add("a1");  
 List.add("a2");  
 List.add("a3");  
 List.add(2,"!!!");  
  
 List.remove(1);  
  
 String i = List.get(0);  
  
 ArrayList<String> Add = new ArrayList<String>(10);  
 Add.add("b0");  
 Add.add("b2");  
 Add.add(1,"b1");  
  
 List.addAll(3, Add);  
}  
void CheckMyHashSet(){  
 MyHashSet<String> states = new MyHashSet<String>();  
  
 states.add("Germany");  
 states.add("France");  
 states.add("Italy");  
 states.add("Italy");  
  
 states.remove("Italy");  
}

**Задание 1**

**MyArrayList**

package HomeWork1.Ex1;  
import java.util.Arrays;  
import java.util.Collection;  
  
public class MyArrayList<E> {  
 private Object[] data; // внутренний массив (выделенная память)  
 private static final int *DEFAULT\_CAPACITY* = 10;  
 private int size = 0; // количество элементов  
  
 //Конструкторы  
 public MyArrayList(){ //Если не задавать емкость (по умолчанию 10)  
 data = new Object[*DEFAULT\_CAPACITY*];  
 }  
 public MyArrayList(int Capacity){ //С указанием емкости  
 ensureCapacity(Capacity);  
 }  
  
 // увеличение емкости списка  
 private void ensureCapacity(int NewSize) {  
 if (data == null){data = new Object[NewSize];return;}  
 data = Arrays.*copyOf*(data, NewSize);  
 }  
  
  
 // добавить элемент в конец  
 public void add(E e) {  
 if (size >= data.length){ensureCapacity(size+1);}  
 data[size++] = e;  
 }  
  
 // добавить элемент с указанием индекса  
 public void add(int index, E e) {  
 checkIndexForAdd(index);  
 if (size >= data.length){ensureCapacity(size+1);}  
  
 // сдвигаем элементы вправо  
 for (int i = size - 1; i >= index; i--) {  
 data[i + 1] = data[i];  
 }  
  
 // вставляем новый элемент  
 data[index] = e;  
 size++;  
 }  
  
 // проверка индекса  
 private void checkIndex(int index) {  
 if (index < 0 || index >= size) {  
 throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + size);  
 }  
 }  
  
 // получить элемент по индексу  
 public E get(int index) {  
 checkIndex(index);  
 return (E) data[index];  
 }  
  
 // удалить по индексу  
 public void remove(int index) {  
 checkIndex(index);  
 for (int i = index; i < size - 1; i++) {  
 data[i] = data[i + 1];  
 }  
 data[--size] = null; // освобождаем последний элемент  
 }  
  
 private void checkIndexForAdd(int index) {  
 if (index < 0 || index > size) {  
 throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + size);  
 }  
 }  
 // добавить все элементы из другого списка  
 public void addAll(int index, Collection<? extends E> col) {  
 if (!(col == null || col.isEmpty())){  
 checkIndexForAdd(index); // проверка индекса для вставки (0 <= index <= size)  
  
 int newSize = size + col.size();  
 ensureCapacity(newSize); // увеличить массив, если нужно  
  
 // сдвигаем существующие элементы вправо  
 for (int i = size - 1; i >= index; i--) {  
 data[i + col.size()] = data[i];  
 }  
  
 // вставляем элементы коллекции  
 int j = index;  
 for (E : col) {  
 data[j++] = e;  
 }  
  
 size = newSize;  
 }  
 }  
}

**MyHashSet**

package HomeWork1.Ex1;  
  
import java.util.LinkedList;  
  
public class MyHashSet<E>  
{  
 private static final int *DEFAULT\_CAPACITY* = 16; //Размер хэш-таблицы  
 private LinkedList<E>[] MyHashTable; //Аналог хэш-таблицы  
  
 @SuppressWarnings("unchecked")  
 public MyHashSet(){  
 MyHashTable = (LinkedList<E>[]) new LinkedList[*DEFAULT\_CAPACITY*];  
 }  
  
 @SuppressWarnings("unchecked")  
 public MyHashSet(int Capacity){  
 MyHashTable = (LinkedList<E>[]) new LinkedList[Capacity];  
 }  
  
 // Хэш-функция (чтобы не выходить за пределы массива)  
 private int getIndex(E e) {  
 return (e == null) ? 0 : Math.*abs*(e.hashCode() % MyHashTable.length);  
 }  
 //Вставка  
 public void add(E e){  
 int index = getIndex(e);  
 if (MyHashTable[index] == null) {  
 MyHashTable[index] = new LinkedList<>();  
 }  
 if (!MyHashTable[index].contains(e)) { // проверка уникальности  
 MyHashTable[index].add(e);  
 }  
 }  
 // Удаление  
 public void remove(E e){  
 int index = getIndex(e);  
 if (MyHashTable[index] != null) {  
 MyHashTable[index].remove(e);  
 }  
 }  
}

**Задание 2**

**Student**

package HomeWork1.Ex2;  
import java.util.List;  
  
public class Student {  
 private String name;  
 private List<Book> books;  
  
 public Student(String name, List<Book> books) {  
 this.name = name;  
 this.books = books;  
 }  
  
 public List<Book> getBooks() { return books; }  
  
 @Override  
 public String toString() {  
 return name + " => " + books;  
 }  
}

**Book**

package HomeWork1.Ex2;  
  
import java.util.Objects;  
  
public class Book {  
 private String title;  
 private int pages;  
 private int year;  
  
 public Book(String title, int pages, int year) {  
 this.title = title;  
 this.pages = pages;  
 this.year = year;  
 }  
  
 public int getPages() { return pages; }  
 public int getYear() { return year; }  
  
 @Override  
 public String toString() {  
 return title + " (" + pages + " стр., " + year + " г.)";  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Book)) return false;  
 Book book = (Book) o;  
 return pages == book.pages &&  
 year == book.year &&  
 Objects.*equals*(title, book.title);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(title, pages, year);  
 }  
}