|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА** – **Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт ИКБ | |
| Кафедра защиты информации | |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4** | |
| **по дисциплине «Технологии и методы программирования»** | |
|  | |
| Выполнила студент группы БИСО-02-21 | Одинцова Ю.А. |
| Проверил | Лесько С.А. |

Москва 2023

**Практическая работа №4**

Ссылка:

https://github.com/just-student-19/tmp\_5\_sem/tree/практика\_4

# Итератор

Итератор — это поведенческий паттерн проектирования, который даёт возможность последовательно обходить элементы составных объектов, не раскрывая их внутреннего представления.

Листинг программы:

class Iterator {

constructor(container) {

this.\_container = container;

this.\_index = 0;

}

next() {

this.\_index += 1;

}

value() {

return this.\_container[this.\_index];

}

hasMore() {

return this.\_index < this.\_container.length;

}

}

// Main

const vec = [167, 310, 844, 953, 569];

const it = new Iterator(vec);

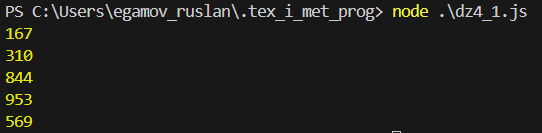
while (it.hasMore()) {

console.log(it.value(), " ");

it.next();

}

Скриншот выполнения программы:



# Посетитель

Посетитель — это поведенческий паттерн проектирования, который позволяет добавлять в программу новые операции, не изменяя классы объектов, над которыми эти операции могут выполняться

Листинг программы:

class Visitor {

visitElementA(element) {

}

visitElementB(element) {

}

}

class Element {

accept(visitor) {

}

}

class ElementA extends Element {

accept(visitor) {

visitor.visitElementA(this);

}

operationA() {

console.log('Операция с элементом №1');

}

}

class ElementB extends Element {

accept(visitor) {

visitor.visitElementB(this);

}

operationB() {

console.log('Операция с элементом №2');

}

}

class SomeVisitor extends Visitor {

visitElementA(element) {

element.operationA();

}

visitElementB(element) {

element.operationB();

}

}

// Main

const elementA = new ElementA();

const elementB = new ElementB();

const visitor = new SomeVisitor();

elementA.accept(visitor);

elementB.accept(visitor);

Скриншот выполнения программы:

