Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

Jaca Core, наследование
Тема

Преподаватель _____ А. С. Черниговский
подпись, дата — инициалы, фамилия

Студент КИ19-17/1б 031939175 _____ А. Д. Непомнящий
номер группы, зачетной книжки — подпись, дата — инициалы, фамилия

Красноярск 2021

1 Цель

Цель настоящей работы состоит в ознакомлении с механизмом наследования в языке Java и повторении основных языковых конструкций языка Java.

2 Задачи

Имеется набор из четырех сущностей. Необходимо выстроить иерархию наследования. В каждом классе (базовом и производных) должно быть минимум одно числовое и одно текстовое поле. При вводе числовых параметров обязательна проверка на число и на диапазон (даже если число может быть любое, проверку необходимо реализовать). Для всех классов должны быть реализованы конструкторы (по умолчанию, с параметрами), методы equals(), hashCode(), toString().

Реализовать консольное Java-приложение, которое имеет простейшее пользовательское меню, состоящее как минимум из следующих пунктов.

1) Добавить новый элемент. (Элементы должны добавляться в коллекцию элементов типа базового класса. Необходимо предусмотреть возможность добавления любого объекта производного класса в данную коллекцию.)
2) Удалить элемент по индексу.
3) Вывод всех элементов в консоль.
4) Сравнение двух элементов по индексу.
5) Завершение работы приложения.

3 Описание задания

Вариант 10. Сущности: корабль, пароход, парусник, корвет.

4 Ход выполнения

Ниже представлены листинги программы по заданию.

Листинг 1 – Код класса Vessel

```
package com.github.durakin.infsystems.lab1;

public abstract class Vessel {
    private final String name;
    private final int tonnage;

    public Vessel() {
        this("Unnamed vessel", 0);
    }

    public Vessel(String name, int tonnage) {
        this.name = name;
        this.tonnage = tonnage;
    }

    @Override
    public boolean equals(Object otherObject) {
        if (this == otherObject) return true;
        if (otherObject == null || getClass() != otherObject.getClass()) return
false;

        Vessel vessel = (Vessel) otherObject;

        if (tonnage != vessel.tonnage) return false;
        return name.equals(vessel.name);
    }

    @Override
    public int hashCode() {
        int result = name.hashCode();
        result = 31 * result + tonnage;
        return result;
    }

    @Override
    public String toString() {
        return "Vessel " + name + "\nTonnage " + tonnage +
                't';
    }
}
```

Листинг 2 – код класса Steamboat

```
package com.github.durakin.infsystems.lab1;

import java.util.Objects;

public class Steamboat extends Vessel {

    private final String fuelType;
    private final int fuelConsumption;

    public Steamboat() {
        super();
        this.fuelType = "Unknown";
        this.fuelConsumption = 0;
    }

    public Steamboat(String name, int tonnage, String fuelType, int
fuelConsumption) {
        super(name, tonnage);
        this.fuelType = fuelType;
        this.fuelConsumption = fuelConsumption;
    }

    @Override
    public boolean equals(Object otherObject) {
        if (this == otherObject) return true;
        if (otherObject == null || getClass() != otherObject.getClass()) return
false;
        if (!super.equals(otherObject)) return false;

        Steamboat steamboat = (Steamboat) otherObject;

        if (fuelConsumption != steamboat.fuelConsumption) return false;
        return Objects.equals(fuelType, steamboat.fuelType);
    }

    @Override
    public int hashCode() {
        int result = super.hashCode();
        result = 31 * result + (fuelType != null ? fuelType.hashCode() : 0);
        result = 31 * result + fuelConsumption;
        return result;
    }
```

Окончание листинга 2

```
    @Override
    public String toString() {

        return super.toString() + "\nType: Steamboat; Fuel type: " + fuelType +
"; Fuel consumption: " +
                fuelConsumption + ';';

    }
}
```

Листинг 3 – код класса Sailboat

```
package com.github.durakin.infsystems.lab1;

import java.util.Objects;

public class Sailboat extends Vessel {
    private final String sailsPlan;
    private final int gunsInstalled;

    public Sailboat() {
        super();
        this.sailsPlan = "Unknown";
        this.gunsInstalled = 0;
    }

    public Sailboat(String name, int tonnage, String sailsPlan, int
gunsInstalled) {
        super(name, tonnage);
        this.sailsPlan = sailsPlan;
        this.gunsInstalled = gunsInstalled;
    }

    @Override
    public boolean equals(Object otherObject) {
        if (this == otherObject) return true;
        if (otherObject == null || getClass() != otherObject.getClass()) return
false;
        if (!super.equals(otherObject)) return false;

        Sailboat sailboat = (Sailboat) otherObject;

        if (gunsInstalled != sailboat.gunsInstalled) return false;
```

Окончание листинга 3

```
        return Objects.equals(sailsPlan, sailboat.sailsPlan);
    }

    @Override
    public int hashCode() {
        int result = super.hashCode();
        result = 31 * result + (sailsPlan != null ? sailsPlan.hashCode() : 0);
        result = 31 * result + gunsInstalled;
        return result;
    }

    @Override
    public String toString() {
        return super.toString() + "\nType: Sailboat; sails plan: " + sailsPlan +
"; Guns installed: " + gunsInstalled +
                ';';
    }
}
```

Листинг 4 – код класса Corvette

```
package com.github.durakin.infsystems.lab1;

import java.util.Objects;

public class Corvette extends Vessel {
    private final String artillery;
    private final int speed;

    public Corvette() {
        super();
        this.artillery = "Unknown";
        this.speed = 0;
    }

    public Corvette(String name, int tonnage, String artillery, int speed) {
        super(name, tonnage);
        this.artillery = artillery;
        this.speed = speed;
    }

    @Override
    public boolean equals(Object o) {
```

Окончание листинга 4

```
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;

        Corvette corvette = (Corvette) o;

        if (speed != corvette.speed) return false;
        return Objects.equals(artillery, corvette.artillery);
    }

    @Override
    public int hashCode() {
        int result = super.hashCode();
        result = 31 * result + (artillery != null ? artillery.hashCode() : 0);
        result = 31 * result + speed;
        return result;
    }

    @Override
    public String toString() {
        return super.toString() + "\nType: Corvette; Installed artillery: " +
artillery + "; Maximal speed: " +
                speed + ';';
    }
}
```

Листинг 5 – код класса Main

```
package com.github.durakin.infsystems.lab1;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import java.util.function.Function;

public class Main {
    public static void main(String[] args) {
        final List<Vessel> vessels = new ArrayList<>();
        var reader = new BufferedReader(new InputStreamReader(System.in));
        boolean lastMenuIteration = false;
        while (!lastMenuIteration) {
            System.out.println("""
```

Продолжение листинга 5

```
                    1. Add new steamboat instance
                    2. Add new sailboat instance
                    3. Add mew corvette instance
                    4. Print all elements
                    5. Delete element by index
                    6. Check if two elements are equal by indexes
                    7. Quit
                    """);


            var cmdNumber = CheckedIntInput("Enter required command number (1-7)
", reader, Main::CommandNumberChecker);


            switch (cmdNumber) {
                case 1 -> {
                    var name = CheckedStringInput("Enter steamboat's name ",
reader, Main::StringChecker);
                    var tonnage = CheckedIntInput("Enter steamboat's tonnage (in
tonnes, integer) ", reader, Main::PositiveIntegerChecker);
                    var fuelType = CheckedStringInput("Enter steamboat's type of
fuel ", reader, Main::StringChecker);
                    var fuelConsumption = CheckedIntInput("Enter steamboat's
fuel consumption (in standard for this type measure, integer) ", reader,
Main::PositiveIntegerChecker);
                    vessels.add(new Steamboat(name, tonnage, fuelType,
fuelConsumption));
                }
                case 2 -> {
                    var name = CheckedStringInput("Enter sailboat's name ",
reader, Main::StringChecker);
                    var tonnage = CheckedIntInput("Enter sailboat's tonnage (in
tonnes, integer) ", reader, Main::PositiveIntegerChecker);
                    var sailsPlan = CheckedStringInput("Enter sailboat's sails
plan ", reader, Main::StringChecker);
                    var gunsInstalled = CheckedIntInput("Enter number of guns
installed on sailboat) ", reader, Main::PositiveIntegerChecker);
                    vessels.add(new Sailboat(name, tonnage, sailsPlan,
gunsInstalled));
                }
                case 3 -> {
                    var name = CheckedStringInput("Enter corvette's name ",
reader, Main::StringChecker);
                    var tonnage = CheckedIntInput("Enter corvette's tonnage (in
tonnes, integer) ", reader, Main::PositiveIntegerChecker);
                    var artillery = CheckedStringInput("Enter corvette's
artillery info ", reader, Main::StringChecker);
                    var speed = CheckedIntInput("Enter maximal speed of corvette
(in knots, integer) ", reader, Main::PositiveIntegerChecker);
                    vessels.add(new Corvette(name, tonnage, artillery, speed));
```

Продолжение листинга 5

```
                }
                case 4 -> {
                    for (var i : vessels) {
                        System.out.println(i.toString() + '\n');
                    }
                }
                case 5 -> {
                    var indexToDelete = CheckedIntInput("Enter index of element
to delete: ", reader, Main::PositiveIntegerChecker);
                    vessels.remove(indexToDelete);
                }
                case 6 -> {
                    var indexToCompare1 = CheckedIntInput("Enter index of first
element to compare: ", reader, Main::PositiveIntegerChecker);
                    var indexToCompare2 = CheckedIntInput("Enter index of other
element to compare: ", reader, Main::PositiveIntegerChecker);
                    try {

System.out.println(vessels.get(indexToCompare1).equals(vessels.get(indexToCompar
e2)) ? "Elements are equal" : "Elements are not equal");
                    } catch (IndexOutOfBoundsException exception) {
                        System.out.println("Index is out of bounds of list of
vessels");
                    }
                }
                case 7 -> lastMenuIteration = true;
            }
        }

        System.out.println();
    }



    private static int CheckedIntInput(String message, BufferedReader reader,
Function<Integer, Boolean> checker) {
        while (true) {
            try {
                System.out.print(message);
                var number = Integer.parseInt(reader.readLine());
                if (!checker.apply(number)) {
                    throw new Exception("Не в промежутке");
                }
                return number;
            } catch (Exception e) {
```

Окончание листинга 5

```
                System.err.println(e.getMessage());
            }
        }
    }

    private static String CheckedStringInput(String message, BufferedReader
reader, Function<String, Boolean> checker) {
        while (true) {
            try {
                System.out.print(message);
                var string = reader.readLine();
                if (!checker.apply(string)) {
                    throw new Exception("String is empty");
                }
                return string;
            } catch (Exception e) {
                System.err.println(e.getMessage());
            }
        }
    }

    private static boolean CommandNumberChecker(int number) {
        return 1 <= number && number <= 7;
    }

    private static boolean PositiveIntegerChecker(int number) {
        return 0 <= number && number < 10000;
    }

    private static boolean StringChecker(String string) {
        return string != null && !string.isBlank();
    }


}
```

5 Выводы

Была написана программа, соответствующая поставленным задачам. В ходе работы были повторены основные языковые конструкции языка программирования Java, механизмы наследования в языке Java.