Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Информатика |
| кафедра |

**ОТЧЁТ О ПРАКТИЧЕСКОЙ** **РАБОТЕ №1**

|  |
| --- |
| Java Core, Наследование. Вариант 7 |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | А. С. Кузнецов |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ20-17/1Б, 032052038 |  |  |  | М. А. Кобелев |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2022

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc113263562)

[2 Описание варианта задания 3](#_Toc113263563)

[3 Исходный код 3](#_Toc113263564)

[4 Результат 16](#_Toc113263565)

[5 Вывод 17](#_Toc113263566)

1. Цель работы

Ознакомиться с механизмом наследования в языке Java. Повторить основные языковые конструкции языка Java.

1. Описание варианта задания

Сооружение, дом, гараж, квартира. 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.
2. Удалить элемент по индексу.
3. Вывод всех элементов в консоль.
4. Сравнение двух элементов на равенство (по индексам).
5. Завершение работы приложения.
6. Исходный код

В листинге 1 представлен код файла Main.java.

Листинг 1 – Вариант 7

package ru.sfu;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Main {  
 public static final int *ADD* = 1;  
 public static final int *REMOVE* = 2;  
 public static final int *SHOW* = 3;  
 public static final int *COMPARE* = 4;  
 public static final int *EXIT* = 5;  
  
 public static final int *ADD\_TRANSPORT* = 1;  
 public static final int *ADD\_CAR* = 2;  
 public static final int *ADD\_TRAIN* = 3;  
 public static final int *ADD\_EXPRESS* = 4;  
  
 public static final int *MIN\_WEIGHT\_TON* = 0;  
 public static final int *MIN\_COST\_DOLLAR* = 3000;  
 public static final int *MIN\_FLOOR* = 1;  
 public static final int *MIN\_WAGON* = 1;  
  
 *// Метод для ввода целого числа* public static int inputInt(String inputName) {  
 System.*out*.println("Введите " + inputName + " = ");  
 Scanner scan = new Scanner(System.*in*);  
 int num;  
  
 while (true) {  
 if (scan.hasNextInt()) {  
 num = scan.nextInt();  
 scan.nextLine();  
 return num;  
 }  
 else {  
 System.*out*.println("Вводится не число!");  
 scan.nextLine();  
 }  
 }  
 }  
  
 *// Метод для ввода строки* public static String inputStr(String inputName) {  
 System.*out*.println("Введите " + inputName + " = ");  
 Scanner scan = new Scanner(System.*in*);  
 return scan.nextLine();  
 }  
  
 public static void main(String[] args) {  
 ArrayList<Transport> list = new ArrayList<Transport>();  
  
 exit:  
 while (true) {  
 System.*out*.println("\n1-добавить объект\n" +  
 "2-удалить объект\n" +  
 "3-вывести все объекты\n" +  
 "4-сравнить 2 объекта(по индексам)\n" +  
 "5-выход\n");  
  
 int amountIndexObjects = list.size() - 1;  
  
 int choice = *inputInt*("выбор");  
 if (choice < 1 || choice > 5) System.*out*.println("Такого выбора нет!");  
 switch (choice) {  
  
 case *ADD*:  
 System.*out*.println("1-добавить Transport\n" +  
 "2-добавить Car\n" +  
 "3-добавить Train\n" +  
 "4-добавить Express\n");  
  
 choice = *inputInt*("выбор");  
 if (choice < 1 || choice > 4) System.*out*.println("Такого выбора нет!");  
  
 String color = null;  
 int weight = 0;  
  
 switch (choice) {  
  
 case *ADD\_TRANSPORT*:  
 color = *inputStr*("цвет");  
 weight = *inputInt*("вес в тоннах");  
 if (weight < *MIN\_WEIGHT\_TON*) {  
 System.*out*.println("Не бывает таких лёгких транспортов!");  
 break;  
 }  
  
 Transport transport = new Transport(color, weight);  
 list.add(transport);  
 System.*out*.println("Добавлено!");  
 break;  
  
 case *ADD\_CAR*:  
 color = *inputStr*("цвет");  
 weight = *inputInt*("вес в тоннах");  
 if (weight < *MIN\_WEIGHT\_TON*) {  
 System.*out*.println("Не бывает таких лёгких транспортов!");  
 break;  
 }  
  
 String brand = *inputStr*("бренд");  
 int cost = *inputInt*("цену в долларах");  
 if (cost < *MIN\_COST\_DOLLAR*) {  
 System.*out*.println("Не бывает таких дешёвых машин!");  
 break;  
 }  
  
 Car car = new Car(color, weight, brand, cost);  
 list.add(car);  
 System.*out*.println("Добавлено!");  
 break;  
  
 case *ADD\_TRAIN*:  
 color = *inputStr*("цвет");  
 weight = *inputInt*("вес в тоннах");  
 if (weight < *MIN\_WEIGHT\_TON*) {  
 System.*out*.println("Не бывает таких лёгких транспортов!");  
 break;  
 }  
  
 String nameFactory = *inputStr*("название завода-изготовителя");  
 int amountWagons = *inputInt*("количество вагонов");  
 if (amountWagons < *MIN\_WAGON*) {  
 System.*out*.println("Не бывает так мало вагонов!");  
 break;  
 }  
  
 Train train = new Train(color, weight, nameFactory, amountWagons);  
 list.add(train);  
 System.*out*.println("Добавлено!");  
 break;  
  
 case *ADD\_EXPRESS*:  
 color = *inputStr*("цвет");  
 weight = *inputInt*("вес в тоннах");  
 if (weight < *MIN\_WEIGHT\_TON*) {  
 System.*out*.println("Не бывает таких лёгких транспортов!");  
 break;  
 }  
  
 nameFactory = *inputStr*("название завода-изготовителя");  
 amountWagons = *inputInt*("количество вагонов");  
 if (amountWagons < *MIN\_WAGON*) {  
 System.*out*.println("Не бывает так мало вагонов!");  
 break;  
 }  
  
 String name = *inputStr*("название экспресса");  
 int amountFloor = *inputInt*("количество этажей");  
 if (amountFloor < *MIN\_FLOOR*) {  
 System.*out*.println("Не бывает так мало этажей!");  
 break;  
 }  
  
 Express express = new Express(color, weight, nameFactory, amountWagons, name, amountFloor);  
 list.add(express);  
 System.*out*.println("Добавлено!");  
 break;  
 }  
 break;  
  
 case *REMOVE*:  
 if (list.size() == 0) {  
 System.*out*.println("Список и так пуст.");  
 break;  
 }  
  
 System.*out*.println("Введите индекс элемента, который хотите удалить" +  
 "(от 0 до " + amountIndexObjects + ")");  
  
 int removeIndex = *inputInt*("выбор");  
 if (removeIndex < 0 || removeIndex > amountIndexObjects) {  
 System.*out*.println("Индекс выходит за границу.");  
 } else {  
 list.remove(removeIndex);  
 System.*out*.println("Элемент удалён!");  
 }  
 break;  
  
 case *SHOW*:  
 for (int i = 0; i < list.size(); i++) {  
 System.*out*.println(i + ". " + list.get(i));  
 }  
 if (list.size() == 0) System.*out*.println("Список пуст.");  
 break;  
  
 case *COMPARE*:  
 if (list.size() < 2) {  
 System.*out*.println("Недостаточно элементов в списке для сравнения.");  
 break;  
 }  
  
 System.*out*.println("Индекс должен быть от 0 до " + amountIndexObjects);  
  
 int compareNum1 = *inputInt*("индекс 1 объекта для сравнения");  
 if (compareNum1 < 0 || compareNum1 > amountIndexObjects) {  
 System.*out*.println("Индекс вышел за границу.");  
 break;  
 }  
  
 int compareNum2 = *inputInt*("индекс 2 объекта для сравнения");  
 if (compareNum2 < 0 || compareNum2 > amountIndexObjects) {  
 System.*out*.println("Индекс вышел за границу.");  
 break;  
 }  
  
 if (list.get(compareNum1).equals(list.get(compareNum2))) System.*out*.println("Элементы равны");  
 else System.*out*.println("Элементы не равны.");  
  
 break;  
  
 case *EXIT*:  
 break exit;  
 }  
 }  
 }  
}

В листинге 2 представлен код файла Transport.java.

Листинг 2 – Вариант 7

package ru.sfu;  
  
import java.util.Objects;  
  
public class Transport {  
  
 private String color;  
 private int weight;  
  
 public Transport() {  
 this.weight = 0;  
 this.color = null;  
 }  
  
 public Transport(String brand, int weight) {  
 this.color = brand;  
 this.weight = weight;  
 }  
  
 public String getColor() {  
 return this.color;  
 }  
  
 public int getWeight() {  
 return this.weight;  
 }  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
  
 public void setWeight(int weight) {  
 this.weight = weight;  
 }  
  
 @Override  
 public boolean equals(Object otherObject) {  
 if (this == otherObject) return true;  
 if (otherObject == null) return false;  
 if (this.getClass() != otherObject.getClass()) return false;  
  
 Transport transport = (Transport) otherObject;  
 return this.weight == transport.weight && this.color.equals(transport.color);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(this.color, this.weight);  
 }  
  
 @Override  
 public String toString() {  
 return "Transport{color=" + this.color + ";weight=" + this.weight + ";}";  
 }  
}

В листинге 3 представлен код файла Car.java.

Листинг 3 – Вариант 7

package ru.sfu;  
  
import java.util.Objects;  
  
public class Car extends Transport {  
 private String brand;  
 private int cost;  
  
 public Car() {  
 super();  
 this.brand = null;  
 this.cost = 0;  
 }  
  
 public Car(String color, int weight, String brand, int cost) {  
 super(color, weight);  
 this.brand = brand;  
 this.cost = cost;  
 }  
  
 public String getBrand() {  
 return this.brand;  
 }  
  
 public int getCost() {  
 return this.cost;  
 }  
  
 @Override  
 public boolean equals(Object otherObject) {  
 if (this == otherObject) return true;  
 if (otherObject == null) return false;  
 if (this.getClass() != otherObject.getClass()) return false;  
  
 if (!super.equals(otherObject)) return false;  
  
 Car car = (Car) otherObject;  
 return this.brand.equals(car.brand) && this.cost == car.cost;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(this.getColor(), this.getWeight(), this.brand, this.cost);  
 }  
  
 @Override  
 public String toString() {  
 return "Car{" +  
 "color=" + this.getColor() + ";" +  
 "weight=" + this.getWeight() + ";" +  
 "brand=" + this.brand + ";" +  
 "cost=" + this.cost + ";}";  
 }  
}

В листинге 4 представлен код файла Train.java.

Листинг 4 – Вариант 7

package ru.sfu;  
  
import java.util.Objects;  
  
public class Train extends Transport {  
 private String nameFactory;  
 private int amountWagons;  
  
 public Train() {  
 super();  
 this.nameFactory = null;  
 this.amountWagons = 0;  
 }  
  
 public Train(String color, int weight, String nameFactory, int amountWagons) {  
 super(color, weight);  
 this.nameFactory = nameFactory;  
 this.amountWagons = amountWagons;  
 }  
  
 public String getNameFactory() {  
 return this.nameFactory;  
 }  
  
 public int getAmountWagons() {  
 return this.amountWagons;  
 }  
  
 @Override  
 public boolean equals(Object otherObject) {  
 if (this == otherObject) return true;  
 if (otherObject == null) return false;  
 if (this.getClass() != otherObject.getClass()) return false;  
  
 if (!super.equals(otherObject)) return false;  
  
 Train train = (Train) otherObject;  
 return this.nameFactory.equals(train.nameFactory) && this.amountWagons == train.amountWagons;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(this.getColor(), this.getWeight(), this.nameFactory, this.amountWagons);  
 }  
  
 @Override  
 public String toString() {  
 return "Train{" +  
 "color=" + this.getColor() + ";" +  
 "weight=" + this.getWeight() + ";" +  
 "nameFactory=" + this.nameFactory + ";" +  
 "amountWagons=" + this.amountWagons + ";}";  
 }  
}

В листинге 5 представлен код файла Express.java.

Листинг 5 – Вариант 7

package ru.sfu;  
  
import java.util.Objects;  
  
public class Express extends Train {  
 private String name;  
 private int amountFloor;  
  
 public Express() {  
 super();  
 this.name = null;  
 this.amountFloor = 0;  
 }  
  
 public Express(String color, int weight, String nameFactory, int amountWagons, String name, int amountFloor) {  
 super(color, weight, nameFactory, amountWagons);  
 this.name = name;  
 this.amountFloor = amountFloor;  
 }  
  
 public String getName() {  
 return this.name;  
 }  
  
 public int getAmountFloor() {  
 return this.amountFloor;  
 }  
  
 public void setName(String name) { this.name = name; }  
  
 public void setAmountFloor(int amountFloor) { this.amountFloor = amountFloor; }  
  
 @Override  
 public boolean equals(Object otherObject) {  
 if (this == otherObject) return true;  
 if (otherObject == null) return false;  
 if (this.getClass() != otherObject.getClass()) return false;  
  
 if (!super.equals(otherObject)) return false;  
  
 Express express = (Express) otherObject;  
 return this.name.equals(express.name) && this.amountFloor == express.amountFloor;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(this.getColor(), this.getWeight(), this.name, this.amountFloor);  
 }  
  
 @Override  
 public String toString() {  
 return "Express{" +  
 "color=" + this.getColor() + ";" +  
 "weight=" + this.getWeight() + ";" +  
 "nameFactory=" + this.getNameFactory() + ";" +  
 "amountWagons=" + this.getAmountWagons() + ";" +  
 "name=" + this.name + ";" +  
 "amountFloor=" + this.amountFloor + ";}";  
 }  
}

1. Результат

Результат работы программы можно увидеть на рисунке 1, 2.

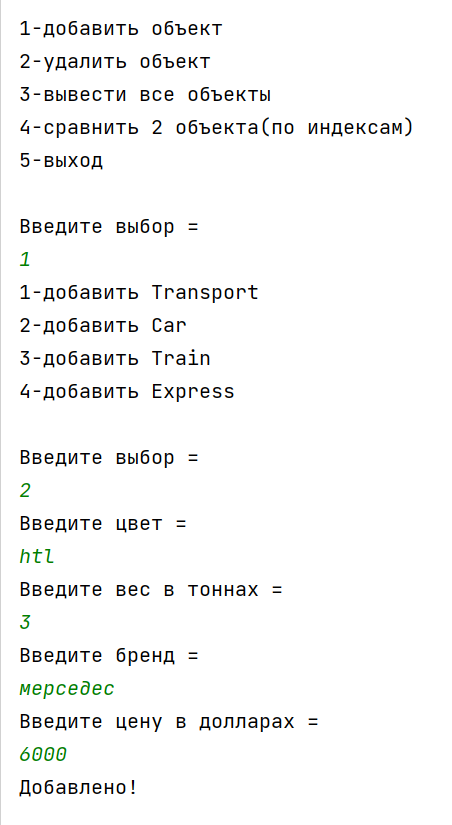


Рисунок 1 – Результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Результат работы программы

1. Вывод

В ходе работы повторил механизм наследования в языке Java, а также повторил основные языковые конструкции языка Java.