Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

|  |
| --- |
| институт Высоких технологий |
| наименование института  кафедра Автоматизированных систем |
| наименование кафедры |

ОТЧЁТ

к лабораторной работе № 4 по дисциплине

|  |
| --- |
| Объектно-ориентированное программирование |
| Разработка классов с использованием механизмов наследования, полиморфизма и инкапсуляции. |
| наименование темы |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИСТб-16-1 |  |  |  | М.Е. Винзовский |
|  | Шифр |  | Подпись |  | И.О. Фамилия |
| Проверил |  |  |  |  | В.Л. Аршинский |
|  |  |  | Подпись |  | И.О. Фамилия |
|  |  |  |  |  |  |

Иркутск 2018 г.

Оглавление

[1 Постановка задачи 3](#_Toc510548137)

[1.1 Цель работы 3](#_Toc510548138)

[1.2 Требования к выполнению лабораторной работы 3](#_Toc510548139)

[1.3 Содержание задания на выполнение лабораторной работы 3](#_Toc510548140)

[1.4 Вариант индивидуального задания 4](#_Toc510548141)

[2 Описание структуры пользовательского меню 5](#_Toc510548142)

[3 Проектирование классов 8](#_Toc510548143)

[4 Описание методов классов и спецификация локальных переменных 9](#_Toc510548144)

[5 Таблица тестов 13](#_Toc510548145)

[6 Результаты тестирования 15](#_Toc510548146)

[7 Исходный код 18](#_Toc510548147)

[Список использованных источников 29](#_Toc510548148)

# 1 Постановка задачи

## 1.1 Цель работы

1. Изучение механизмов наследования, полиморфизма и инкапсуляции на практике.
2. Получение практических навыков описания иерархии классов и организации работы с объектами-членами классов-«групп».
3. Получение практических навыков использования абстрактных классов и переопределения унаследованных методов.

## 1.2 Требования к выполнению лабораторной работы

1. В ходе выполнения лабораторной работы студент должен на практике изучить механизмы наследования, полиморфизма и инкапсуляции.
2. Студент должен реализовать иерархию классов в соответствии с индивидуальным заданием.
3. В зависимости от задания некоторые из классов иерархии (как минимум один) являются абстрактными и служат для выделения общих данных и поведения для других классов.
4. Абстрактный класс должен содержать как минимум один абстрактный метод, реализация которого у его наследников должна различаться.
5. Разработанная иерархия классов должны быть дополнена соответствующим классом-«группой» для организации работы с объектами этих классов.
6. Класс-«группа» должен содержать методы для добавления объектов разработанной иерархии классов, удаления, поиска и вывода на экран.
7. В главном классе реализовать демонстрацию работы с классом-«группой»: добавление объектов разработанной иерархии классов в группу, удаление, поиск, вызов и выполнение общего для всех членов группы метода.
8. Все поля и методы должны иметь соответствующие модификаторы доступа.
9. Программный код должен быть оформлен в соответствии с установленными правилами и содержать необходимые комментарии.
10. По результатам выполнения оформляется отчет в соответствии с установленными требованиями.

## 1.3 Содержание задания на выполнение лабораторной работы

В соответствии с индивидуальным заданием описать иерархию классов, для каждого класса описать поля и соответствующие методы доступа к ним. В зависимости от задания некоторые из этих классов(как минимум один) являются абстрактными и служат для выделения общих данных и поведения для других классов. Абстрактный класс должен содержать как минимум один абстрактный метод, реализация которого у его наследников должна различаться. Помимо этого в общую часть задания входит разработка класса группирующего объекты описанных в соответствии с заданием классов. Для первого варианта индивидуального задания это может быть класс «кафедра». Для выполнения задания необходимо создать некоторое количество объектов, добавить их в группу используя предусмотренные методы класса-«группы» и для каждого из них вызвать унаследованный метод.

## 1.4 Вариант индивидуального задания

Вариант задания был выбран самостоятельно. Разрабатывается программа "Завод по производству роботов", которая содержит в себе:

* абстрактный класс Robots;
* классы, которые являются его потомками, такие как класс Military, класс Household и класс Toy;
* группирующий класс FactoryRobots;
* класс Menu для реализации пользовательского меню;
* главный класс lab4, в котором вызывается реализация.

Все классы, являющиеся типами прототипов роботов, а также группирующий класс, собраны в отдельный пакет java для упорядочения.

Программа должна выполнять:

Добавление нового завода вручную;

Добавление нового завода по умолчанию;

Добавление нового прототипа робота любого из вышеперечисленных типов;

Удаление роботов по типу и по модели;

Поиск роботов по моделям и типам;

Взаимодействие с роботами в зависимости от типа;

# 2 Описание структуры пользовательского меню

Меню в данной работе представляет собой вывод пунктов меню, каждому из которых присвоен порядковый номер. Выбор пункта меню осуществляется вводом этого номера. При вводе неверного номера или вообще других данных, выводится сообщение об ошибке и предлагается ещё раз ввести номер пункта меню.

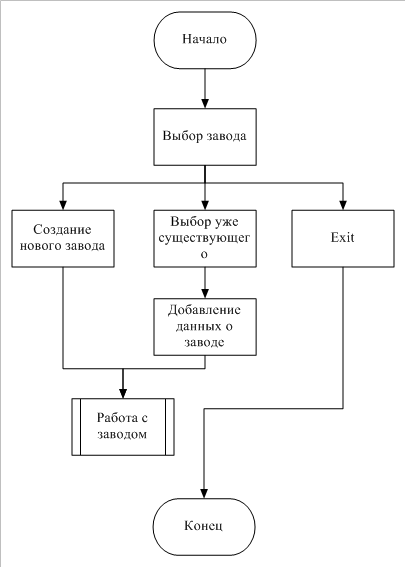


Рис. 1 – схема пользовательского меню программы

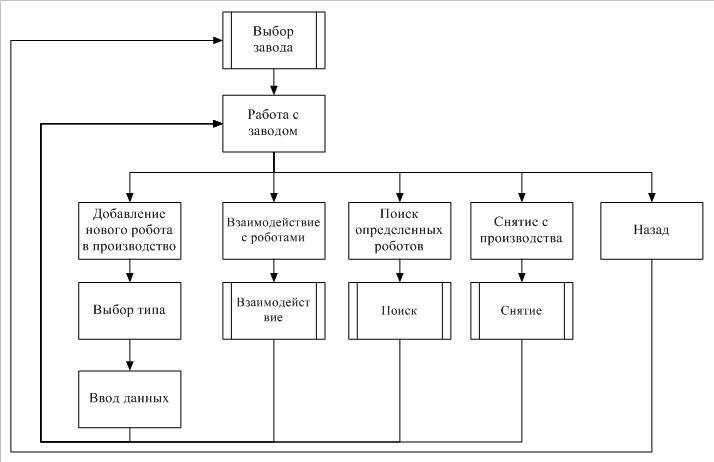


Рис. 2 – схема пользовательского меню для работы с данными завода

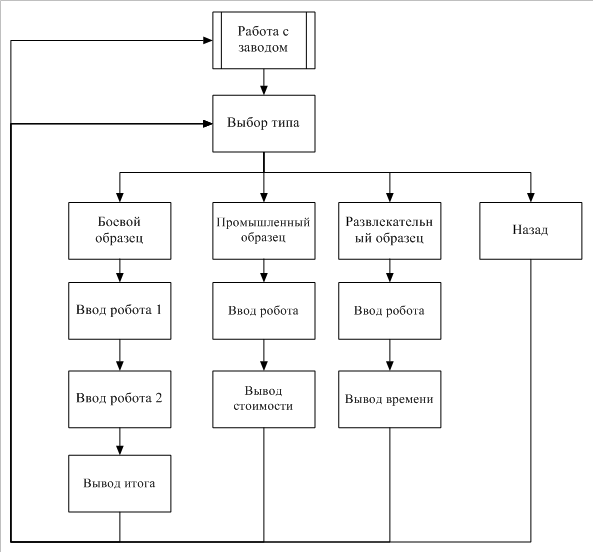


Рис. 3 – схема пользовательского меню для работы с определенными типами роботов

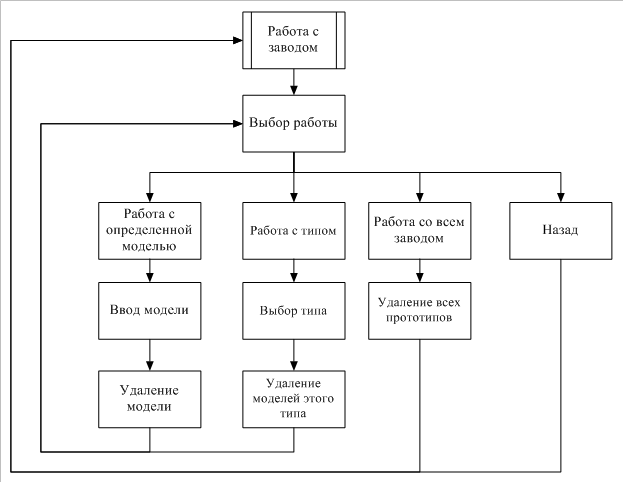


Рис. 4 – схема пользовательского меню для удаления прототипов

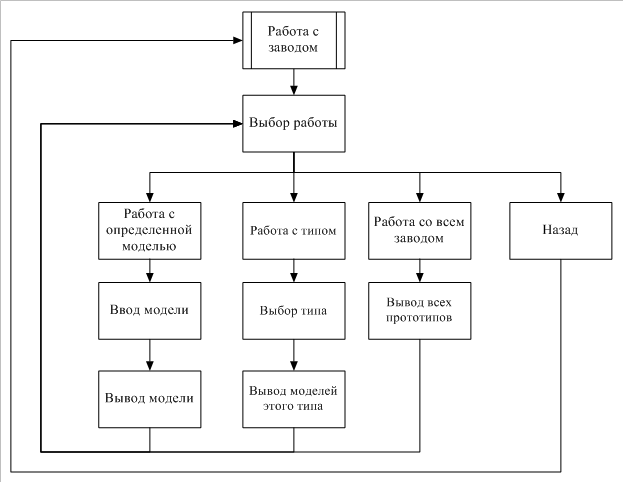


Рис. 5 – схема пользовательского меню для вывода прототипов

# 3 Проектирование классов

В процессе разработки программы были выделены 4 класса:

* класс Robots – абстрактный класс роботов;
* класс Military – класс боевых роботов;
* класс Household – класс промышленных роботов;
* класс Toy – класс развлекательных роботов;
* класс FactoryRobots – класс завода, группирующий класс;
* класс Lab4 – основной класс
* класс Menu–класс, в котором реализуется работа с пользователем;

Таблица 1 - Поля класса Robots

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | **ОДЗ** |
| ***Класс Robots*** | | | | |
| 1 | model | String | Модель робота | - |
| 2 | mass | int | Масса | [0...2147483647] |
| 3 | energy | int | Энергия | [0...2147483647] |

Таблица 2 - Поля класса Military

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | **ОДЗ** |
| ***Класс Military*** | | | | |
| 4 | force | int | Сила в ст. ед. | [0...2147483647] |
| 5 | durability | int | Прочность в ст. ед. | [0...2147483647] |

Таблица 3 - Поля класса Household

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | **ОДЗ** |
| ***Класс Household*** | | | | |
| 6 | zone | int | Зона работы | [0...2147483647] |
| 7 | speed | int | Скорость работы | [0...2147483647] |

Таблица 4 - Поля класса Toy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | **ОДЗ** |
| ***Класс Toy*** | | | | |
| 8 | age | int | Минимальный возраст | [0...2147483647] |
| 9 | detail | int | Кол-во деталей | [0...2147483647] |

Таблица 5 - Поля класса FactoryRobots

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | **ОДЗ** |
| ***Класс FactoryRobots*** | | | | |
| 10 | robotsPrototipe | ArrayList<Robots> | Список прототипов | - |

Таблица 6 - Поля класса Menu

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | | **ОДЗ** | |
| ***Класс Menu*** | | | | | | |
| 11 | fr | FactoryRobots | | Завод | | - |

На рисунке 6 изображена UML диаграмма классов

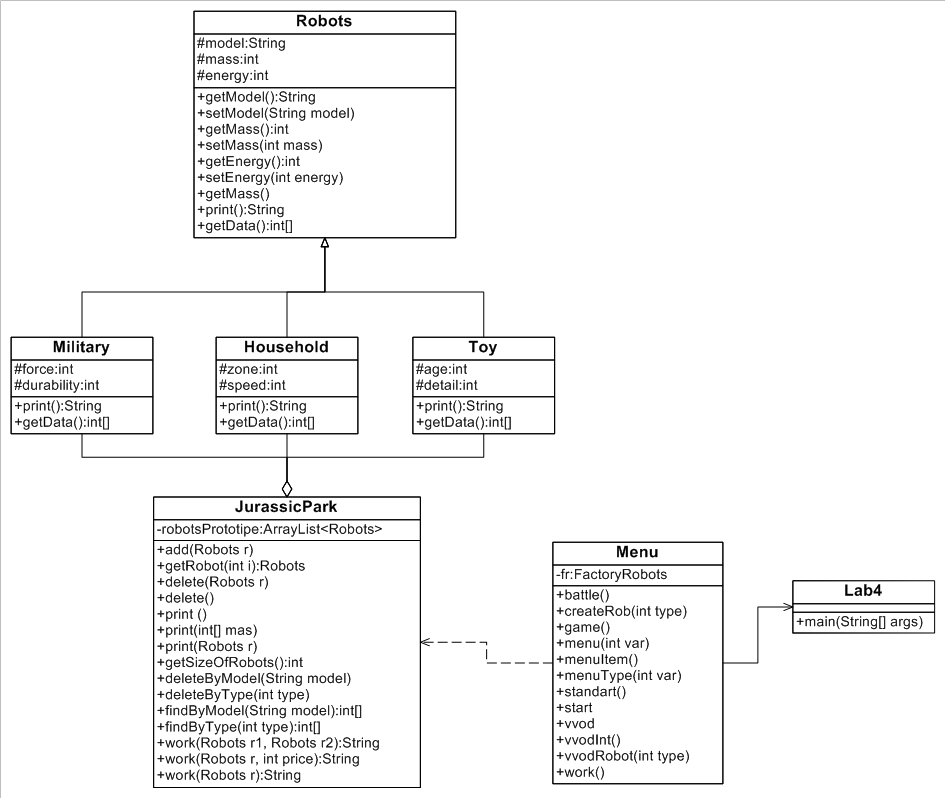


Рис. 6 – Диаграмма классов

# 4 Описание методов классов и спецификация локальных переменных

Таблица 7 – Методы классов и спецификация локальных переменных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | **ОДЗ** |
| **Класс Lab4** | | | | |
| ***public static void main(String[] args) – точка входа в программу*** | | | | |
| **class Menu** | | | | |
| ***public static void start() – Инициализация меню*** | | | | |
| 1 | num | Вводимый элемент | int | [0;2147483647] |
| 2 | uslovie | Переменная для проверки данных | boolean | {1;0} |
| 3 | vibor | Переменная для выбора команды | String | - |
| ***public static void standart() – Создание стандартного завода*** | | | | |
| ***public static void menuItem()– Инициализация меню*** | | | | |
| 4 | uslovie | Переменная для проверки данных | boolean | {1;0} |
| 5 | vibor | Переменная для выбора команды | String | - |
| ***public static void menuType(int var) – Инициализация меню*** | | | | |
| 6 | uslovie | Переменная для проверки данных | boolean | {1;0} |
| 7 | vibor | Переменная для выбора команды | String | - |
| 8 | n | Массив роботов | int[] | [0;2147483647] |
| ***public static void menu(int var) – Инициализация меню*** | | | | |
| 9 | uslovie | Переменная для проверки данных | boolean | {1;0} |
| 10 | vibor | Переменная для выбора команды | String | - |
| 11 | n | Массив роботов | int[] | [0;2147483647] |
| ***public static void battle() – Взаимодействие с боевыми роботами*** | | | | |
| 12 | n | Массив роботов | int[] | [0;2147483647] |
| 13 | robs | Выбранные роботы | int[] | [0;2147483647] |
| ***public static void work() – Взаимодействие с промышленными роботами*** | | | | |
| 14 | n | Массив роботов | int[] | [0;2147483647] |
| 15 | rob | Выбранный робот | int | [0;2147483647] |
| 16 | price | Цена электроэнергии | int | [0;2147483647] |
| ***public static void game() – Взаимодействие с развлекательными роботами*** | | | | |
| 17 | n | Массив роботов | int[] | [0;2147483647] |
| 18 | rob | Выбранный робот | int | [0;2147483647] |
| ***public static void createRob(int type) – добавление нового робота*** | | | | |
| 19 | text | Просьба для ввода | String | - |
| 20 | model | Модель робота | String | - |
| 21 | mass | Масса робота | int | [0;2147483647] |
| 22 | energy | Энергия робота | int | [0;2147483647] |
| 23 | data1 | Первая доп хар-ка робота | int | [0;2147483647] |
| 24 | data2 | Вторая доп хар-ка робота | int | [0;2147483647] |
| ***public static String vvod() – ввод текста*** | | | | |
| 25 | stdin | Объект для ввода данных с клавиатуры | BufferedReader | - |
| 26 | text | Введенный текст | String | - |
| ***public static int vvodInt() – ввод чисел*** | | | | |
| 27 | stdin | Объект для ввода данных с клавиатуры | BufferedReader | - |
| 28 | count | Введенное число | int | [0;2147483647] |
| ***public static int vvodRobot(int type) – поиск робота определенного типа*** | | | | |
| 29 | n | Массив роботов | int[] | [0;2147483647] |
| 30 | m | Введенный робот | int | [0;2147483647] |
| 31 | rob | Выбранный робот | int | [0;2147483647] |
| 32 | usl | Проверка условия | boolean | {1;0} |
| 33 | r | Модель робота | String | - |
| **class FactoryRobots** | | | | |
| ***FactoryRobots()– Конструктор*** | | | | |
| ***public void add(Robots r) – Метод добавления*** | | | | |
| ***public Robots getRobot(int i) - Передача определенного робота*** | | | | |
| ***public void delete(Robots r)– Метод удаления*** | | | | |
| ***public int getSizeOfRobots()– Метод для того, чтобы узнать количество прототипов на заводе*** | | | | |
| ***void print()– Метод для вывода списка роботов*** | | | | |
| 34 | i | Счетчик | int | [0;2147483647] |
| ***public void print(int[] mas) – Метод для вывода списка определенных роботов*** | | | | |
| 35 | i | Счетчик | int | [0;2147483647] |
| ***public void print(Robots r) – Метод для вывода робота*** | | | | |
| ***public int[] findByModel(String model) – Метод для поиска по модели*** | | | | |
| 36 | mas | Массив роботов | int[] | [0;2147483647] |
| 37 | mas2 | Сортированный массив роботов | int[] | [0;2147483647] |
| 38 | b | Количество найденных роботов | int | [0;2147483647] |
| 39 | usl | Проверка условия | boolean | {1;0} |
| 40 | i | Счетчик | int | [0;2147483647] |
| ***public int[] findByType(int type) – Метод для поиска по типу*** | | | | |
| 41 | mas | Массив роботов | int[] | [0;2147483647] |
| 42 | mas2 | Сортированный массив роботов | int[] | [0;2147483647] |
| 43 | b | Количество найденных роботов | int | [0;2147483647] |
| 44 | usl | Проверка условия | boolean | {1;0} |
| 45 | i | Счетчик | int | [0;2147483647] |
| ***public void deleteByModel(String model) – Метод для удаления по модели*** | | | | |
|  | usl | Проверка условия | boolean | {1;0} |
| 46 | i | Счетчик | int | [0;2147483647] |
| ***public void deleteByType(int type) – Метод для удаления по типу*** | | | | |
| 47 | usl | Проверка условия | boolean | {1;0} |
| 48 | i | Счетчик | int | [0;2147483647] |
| ***public String work(Robots r1, Robots r2) – Метод для взаимодействия боевых роботов*** | | | | |
| 49 | character1 | Доп. характеристики робота 1 | int[] | [0;2147483647] |
| 50 | character2 | Доп. характеристики робота 2 | int[] | [0;2147483647] |
| 51 | text | Текст для вывода |  |  |
| ***public String work(Robots r, int price) – Метод для взаимодействия производственных роботов*** | | | | |
| 52 | character | Доп. характеристики робота | int[] | [0;2147483647] |
| 53 | priceTotal | Цена работы | float | - |
| 54 | text | Текст для вывода | String | - |
| ***public String work(Robots r) – Метод для взаимодействия развлекательных роботов*** | | | | |
| 55 | character | Доп. характеристики робота | int[] | [0;2147483647] |
| 56 | time | Время сборки | int |  |
| 57 | text | Текст для вывода | String | - |
| **class Robots** | | | | |
| ***public Robots(String model, int mass, int energy) – Конструктор с вводом данных*** | | | | |
| ***public abstract String print() -Метод для вывода информации на экран, переопределяемый в классах наследниках*** | | | | |
| ***public abstract int[] getData()- Метод для вывод доп. данных, переопределяемый в классах наследниках*** | | | | |
| **class Military** | | | | |
| ***public Military(String model, int mass, int energy, int force, int durability) – Конструктор с вводом данных*** | | | | |
| ***public String print() - Метод для вывода информации на экран*** | | | | |
| ***public abstract int[] getData()- Метод для вывод доп. данных*** | | | | |
| **Class Household** | | | | |
| ***public Household(String model, int mass, int energy, int zone, int speed)– Конструктор с вводом данных*** | | | | |
| ***public String print() - Метод для вывода информации на экран*** | | | | |
| ***public abstract int[] getData()- Метод для вывод доп. данных*** | | | | |
| **Class Toy** | | | | |
| ***public Toy(String model, int mass, int energy, int age, int detail) – Конструктор с вводом данных*** | | | | |
| ***public String print() - Метод для вывода информации на экран*** | | | | |
| ***public abstract int[] getData()- Метод для вывод доп. данных*** | | | | |

# 5 Таблица тестов

Таблица 8 – Выделенные классы эквивалентности для основного метода программы (эти классы эквивалентности (тесты тоже) подходят для любого метода, суть которого заключается в выборе целого числа в заданном интервале)

Таблица 8 – Выделенные классы эквивалентности для основного метода программы

|  |  |  |
| --- | --- | --- |
| Входные условия | Правильные классы эквивалентности | Неправильные классы эквивалентности |
| Введенная строка содержит только цифры там, где требуется ввод цифр | Содержит только цифры. (1) | Содержит другие символы.(2) |
| Число не в ОДЗ | Число от 0 до 2, целое(3) | Число меньше 0 (4)  Число больше 2 (5) |
| Введенная строка не найдена | Строка найдена(6) | Строка не найдена(7) |

Таблица 9 – таблица тестов для основного метода программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входные данные | Выходные данные |
| 1 | Введено целое правильное число  (классы эквивалентности 1,3) | 2 | Сообщение с запросом данных, соответствующих задаче. |
| 2 | Введено целое неподходящее число  (классы эквивалентности 1, 4 или 5) | 5 | "Нет такого варианта" |
| 3 | Введено не число (классы эквивалентности 2) | r | "Нет такого варианта" |
| 4 | Введена строка (классы эквивалентности 6) | "RD1" | Вывод информации о сотруднике |
| 5 | Введена строка (классы эквивалентности 7) | "Gid" | "Нет такой модели" |

# 6 Результаты тестирования

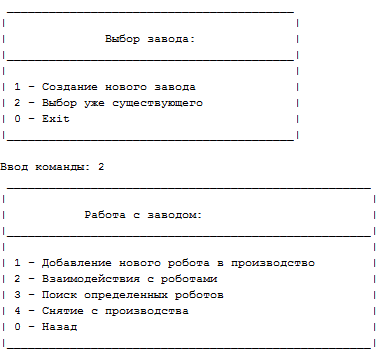


Рисунок 7 - результат тестирования основного метода программы

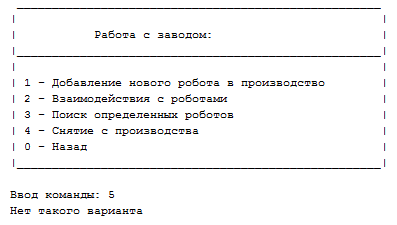


Рисунок 8- результат тестирования основного метода программы

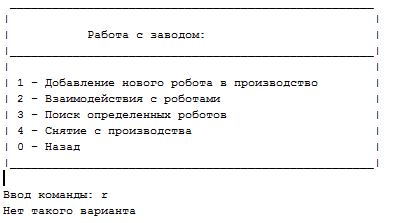


Рисунок 9- результат тестирования основного метода программы

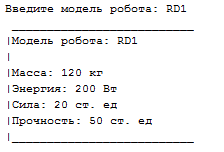


Рисунок 10- результат тестирования основного метода программы



Рисунок 11- результат тестирования основного метода программы

# 7 Исходный код

7.1 Исходный код класса Lab4

package lab4;

import java.io.IOException;

public class Lab4 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

try {

Menu.start();

} catch (IOException ex) {

}

}

}

7.2 Исходный код класса Robots

package Roboting;

public abstract class Robots {

protected String model;

protected int mass;

protected int energy;

public Robots(String model, int mass, int energy) {

this.model = model;

this.mass = mass;

this.energy = energy;

}

public String getModel() {

return model;

}

public void setModel(String model) {

this.model = model;

}

public int getMass() {

return mass;

}

public void setMass(int mass) {

this.mass = mass;

}

public int getEnergy() {

return energy;

}

public void setEnergy(int energy) {

this.energy = energy;

}

public abstract String print();

public abstract int[] getData();

}

7.3 Исходный код класса FactoryRobots

package Roboting;

import java.util.ArrayList;

public class FactoryRobots {

private ArrayList<Robots> robotsPrototipe;

public FactoryRobots() {

robotsPrototipe = new ArrayList<Robots>();

}

public void add(Robots r) {

robotsPrototipe.add(r);

}

public Robots getRobot(int i) {

return robotsPrototipe.get(i);

}

public void delete(Robots r) {

robotsPrototipe.remove(r);

}

public void delete() {

robotsPrototipe.removeAll(robotsPrototipe);

System.out.println("Удаление завершено!");

}

public void print() {

for (int i = 0; i < robotsPrototipe.size(); i++) {

print(robotsPrototipe.get(i));

}

}

public void print(int[] mas) {

for (int i = 0; i < mas.length; i++) {

print(robotsPrototipe.get(mas[i]));

}

}

public void print(Robots r) {

System.out.println(r.print());

}

public int getSizeOfRobots() {

return robotsPrototipe.size();

}

public void deleteByModel(String model) {

boolean usl = true;

for (int i = 0; i < robotsPrototipe.size(); i++) {

if (robotsPrototipe.get(i).model.equals(model)) {

robotsPrototipe.remove(i);

usl = false;

}

}

if (usl) {

System.out.println("Нет такой модели!");

} else {

System.out.println("Удаление завершено!");

}

}

public void deleteByType(int type) {

boolean usl = true;

for (int i = 0; i < robotsPrototipe.size(); i++) {

if (type == 1) {

{

if (robotsPrototipe.get(i) instanceof Military) {

robotsPrototipe.remove(i--);

usl = false;

}

}

}

if (type == 2) {

if (robotsPrototipe.get(i) instanceof Household) {

robotsPrototipe.remove(i--);

usl = false;

}

}

if (type == 3) {

if (robotsPrototipe.get(i) instanceof Toy) {

robotsPrototipe.remove(i--);

usl = false;

}

}

}

if (usl) {

System.out.println("Данный тип прототипов отстутсвует!");

} else {

System.out.println("Удаление завершено!");

}

}

public int[] findByModel(String model) {

int[] mas = new int[robotsPrototipe.size()];

int b = 0;

boolean usl = true;

for (int i = 0; i < robotsPrototipe.size(); i++) {

if (robotsPrototipe.get(i).model.equals(model)) {

usl = false;

mas[b++] = i;

}

}

if (usl) {

System.out.println("Нет такой модели!");

mas = null;

return mas;

}

int[] mas2 = new int[b];

for (int i = 0; i < b; i++) {

mas2[i] = mas[i];

}

return mas2;

}

public int[] findByType(int type) {

int[] mas = new int[robotsPrototipe.size()];

int b = 0;

boolean usl = true;

for (int i = 0; i < robotsPrototipe.size(); i++) {

if (type == 1) {

{

if (robotsPrototipe.get(i) instanceof Military) {

mas[b++] = i;

usl = false;

}

}

}

if (type == 2) {

if (robotsPrototipe.get(i) instanceof Household) {

mas[b++] = i;

usl = false;

}

}

if (type == 3) {

if (robotsPrototipe.get(i) instanceof Toy) {

mas[b++] = i;

usl = false;

}

}

}

if (usl) {

System.out.println("Данный тип прототипов отстутсвует!");

mas = null;

return mas;

}

int[] mas2 = new int[b];

for (int i = 0; i < b; i++) {

mas2[i] = mas[i];

}

return mas2;

}

public String battle(Robots r1, Robots r2) {

String text = "В результате проверки:\n";

int[] character1 = r1.getData();

int[] character2 = r2.getData();

do {

character1[1] -= character2[0];

character2[1] -= character1[0];

} while (character1[1] > 0 && character2[1] > 0);

if (character1[1] > 0) {

text += "Победил робот: " + r1.getModel() + "!\nОставшаяся прочность: " + character1[1];

} else if (character2[1] > 0) {

text += "Победил робот: " + r2.getModel() + "!\nОставшаяся прочность: " + character2[1];

} else {

text += "Оба робота уничтожены!";

}

return text;

}

public String work(Robots r, int price) {

String text;

int[] character = r.getData();

float priceTotal = (r.energy \* price \* 8 \* character[1] \* 30) / character[0];

text = "Цена работы робота в месяц(при 8 часах работы в день): " + priceTotal;

return text;

}

public String game(Robots r) {

String text = "";

int[] character = r.getData();

int time = character[1] \* 5;

text = "Среднее время сборки робота: " + time + " мин.";

return text;

}

}

7.4 Исходный код класса Menu

package lab4;

import Roboting.Military;

import Roboting.FactoryRobots;

import Roboting.Toy;

import Roboting.Household;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Menu {

private static FactoryRobots fr = new FactoryRobots();

public static void start() throws IOException {

boolean uslovie = true;

while (uslovie) {

fr.delete();

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

System.out.println("| |");

System.out.println("| Выбор завода: |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.println("| |");

System.out.println("| 1 - Создание нового завода |");

System.out.println("| 2 - Выбор уже существующего |");

System.out.println("| 0 - Exit |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.print("\nВвод команды: ");

String vibor = vvod();

switch (vibor) {

case "1":

menuItem();

break;

case "2":

standart();

menuItem();

break;

case "0":

uslovie = false;

break;

default:

System.out.println("Нет такого варианта");

break;

}

}

}

public static void standart() {

fr.add(new Military("RD2", 100, 200, 30, 20));

fr.add(new Military("RD1", 120, 200, 20, 50));

fr.add(new Military("Ski-fi", 100, 200, 20, 50));

fr.add(new Household("Gth42", 5, 30, 40, 2));

fr.add(new Household("LK682", 30, 780, 10, 3));

fr.add(new Toy("Dog-1334", 30, 50, 5, 10));

fr.add(new Toy("Cat-643", 10, 305, 10, 35));

}

public static void menuItem() throws IOException {

boolean uslovie = true;

while (uslovie) {

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

System.out.println("| |");

System.out.println("| Работа с заводом: |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.println("| |");

System.out.println("| 1 - Добавление нового робота в производство |");

System.out.println("| 2 - Взаимодействия с роботами |");

System.out.println("| 3 - Поиск определенных роботов |");

System.out.println("| 4 - Снятие с производства |");

System.out.println("| 0 - Назад |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.print("\nВвод команды: ");

int[] n;

String vibor = vvod();

switch (vibor) {

case "1":

menuType(1);

break;

case "2":

if (fr.getSizeOfRobots() == 0) {

System.out.println("На заводе нет схем ни одного робота!");

} else {

menuType(2);

}

break;

case "3":

if (fr.getSizeOfRobots() == 0) {

System.out.println("На заводе нет схем ни одного робота!");

} else {

menu(3);

}

break;

case "4":

if (fr.getSizeOfRobots() == 0) {

System.out.println("На заводе нет схем ни одного робота!");

} else {

menu(4);

}

break;

case "0":

uslovie = false;

break;

default:

System.out.println("Нет такого варианта");

break;

}

}

}

public static void menuType(int var) throws IOException {

boolean uslovie = true;

while (uslovie) {

if (fr.getSizeOfRobots() == 0 && var != 1) {

break;

}

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

System.out.println("| |");

System.out.println("| Выберете тип: |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.println("| |");

System.out.println("| 1 - Боевой образец |");

System.out.println("| 2 - Промышленный образец |");

System.out.println("| 3 - Развлекательный образец |");

System.out.println("| 0 - Назад |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.print("\nВвод команды: ");

int[] n;

String vibor = vvod();

switch (vibor) {

case "1":

if (var == 1) {

createRob(1);

}

if (var == 2) {

battle();

}

if (var == 3) {

n = fr.findByType(1);

if (n != null) {

fr.print(n);

}

}

if (var == 4) {

fr.deleteByType(1);

}

break;

case "2":

if (var == 1) {

createRob(2);

}

if (var == 2) {

work();

}

if (var == 3) {

n = fr.findByType(2);

if (n != null) {

fr.print(n);

}

}

if (var == 4) {

fr.deleteByType(2);

}

break;

case "3":

if (var == 1) {

createRob(3);

}

if (var == 2) {

game();

}

if (var == 3) {

n = fr.findByType(3);

if (n != null) {

fr.print(n);

}

}

if (var == 4) {

fr.deleteByType(3);

}

break;

case "0":

uslovie = false;

break;

default:

System.out.println("Нет такого варианта");

break;

}

}

}

public static void menu(int var) throws IOException {

boolean uslovie = true;

while (uslovie) {

if (fr.getSizeOfRobots() == 0) {

break;

}

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

System.out.println("| |");

System.out.println("| Выбор работы! |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.println("| |");

System.out.println("| 1 - Работа с определенной моделью |");

System.out.println("| 2 - Работа с определенным типом |");

System.out.println("| 3 - Работа со всем заводом |");

System.out.println("| 0 - Назад |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.print("\nВвод команды: ");

int[] n;

String vibor = vvod();

switch (vibor) {

case "1":

System.out.print("Введите модель робота: ");

vibor = vvod();

if (var == 3) {

n = fr.findByModel(vibor);

if (n != null) {

fr.print(n);

}

}

if (var == 4) {

fr.deleteByModel(vibor);

}

break;

case "2":

menuType(var);

break;

case "3":

if (var == 3) {

fr.print();

}

if (var == 4) {

fr.delete();

}

break;

case "0":

uslovie = false;

break;

default:

System.out.println("Нет такого варианта");

break;

}

}

}

public static void battle() throws IOException {

int[] n;

n = fr.findByType(1);

if (n != null) {

if (n.length > 1) {

System.out.println("Битвы роботов!");

System.out.println("Текущие прототипы создаваемые на заводе:");

fr.print(n);

int[] robs = new int[2];

System.out.println("\nБоевой робот 1: ");

robs[0] = vvodRobot(1);

System.out.println("\nБоевой робот 2: ");

robs[1] = vvodRobot(1);

System.out.println(fr.battle(fr.getRobot(robs[0]), fr.getRobot(robs[1])));

} else {

System.out.println("Прототип боевого робота всего 1, невозможно устроить битву!");

}

}

}

public static void work() throws IOException {

int[] n;

n = fr.findByType(2);

if (n != null) {

System.out.println("Вычисление стоимости работы робта за месяц!");

fr.print(n);

System.out.println("\nМодель промышленного робота:");

int rob = vvodRobot(2);

System.out.println("Цена за 1 вт/ч:");

int price = vvodInt();

System.out.println(fr.work(fr.getRobot(rob), price));

}

}

public static void game() throws IOException {

int[] n;

n = fr.findByType(3);

if (n != null) {

System.out.println("Вычисление среднего времени сборки робота!");

fr.print(n);

System.out.println("\nМодель развлекательного робота:");

int rob = vvodRobot(3);

System.out.println(fr.game(fr.getRobot(rob)));

}

}

public static void createRob(int type) throws IOException {

String text = "";

String model;

int mass, energy, data1, data2;

System.out.println("Введите модель робота: ");

model = vvod();

System.out.println("Введите массу робота: ");

mass = vvodInt();

System.out.println("Введите энергию робота: ");

energy = vvodInt();

if (type == 1) {

text = "Введите силу в ст. ед: ";

}

if (type == 2) {

text = "Введите зону действия в м2: ";

}

if (type == 3) {

text = "Введите минимальный возраст: ";

}

System.out.println(text);

data1 = vvodInt();

if (type == 1) {

text = "Введите прочность в ст. ед: ";

}

if (type == 2) {

text = "Введите скорость работы: ";

}

if (type == 3) {

text = "Введите количество деталей: ";

}

System.out.println(text);

data2 = vvodInt();

if (type == 1) {

fr.add(new Military(model, mass, energy, data1, data2));

}

if (type == 2) {

fr.add(new Household(model, mass, energy, data1, data2));

}

if (type == 3) {

fr.add(new Toy(model, mass, energy, data1, data2));

}

}

public static String vvod() throws IOException {

String text;

BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));

text = stdin.readLine();

return text;

}

public static int vvodInt() throws IOException {

BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));

int count = 0;

while (true) {

try {

System.out.print("Введите число: ");

count = Integer.parseInt(stdin.readLine()); // из строки в число

if (count > 0) {

break;

} else {

System.out.println("Число должно быть больше 0!");

}

} catch (NumberFormatException e) {

System.out.println("Ошибка ввода!");

}

}

return count;

}

public static int vvodRobot(int type) throws IOException {

int[] n;

n = fr.findByType(type);

int[] m;

int rob;

boolean usl = false;

while (true) {

System.out.print("Введите модель робота: ");

String r = vvod();

m = fr.findByModel(r);

if (m != null) {

for (int i = 0; i < n.length; i++) {

for (int j = 0; j < m.length; j++) {

if (n[i] == m[j]) {

usl = true;

}

}

}

}

if (usl) {

rob = m[0];

break;

} else if (m == null); else {

System.out.println("Робот не относится к выбранному типу!");

}

}

return rob;

}

}

7.5 Исходный код класса Military

package Roboting;

public class Military extends Robots {

protected int force;

protected int durability;

public Military(String model, int mass, int energy, int force, int durability) {

super(model, mass, energy);

this.force = force;

this.durability = durability;

}

@Override

public String print() {

String text;

text = " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n|Модель робота: " + this.model + "\n|\n|Масса: "

+ this.mass + " кг\n|Энергия: " + this.energy + " Вт\n|Сила: " + this.force + " ст. ед\n|Прочность: " + this.durability + " ст. ед\n|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

return text;

}

@Override

public int[] getData() {

int[] mil = {this.force, this.durability};

return mil;

}

}

7.6 Исходный код класса Household

package Roboting;

public class Household extends Robots {

protected int zone;

protected int speed;

public Household(String model, int mass, int energy, int zone, int speed) {

super(model, mass, energy);

this.zone = zone;

this.speed = speed;

}

@Override

public String print() {

String text;

text = " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n|Модель робота: " + this.model + "\n|\n|Масса: "

+ this.mass + " кг\n|Энергия: " + this.energy + " Вт\n|Зона действия: " + this.zone

+ " м2\n|Скорость работы: " + this.speed + "\n|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

return text;

}

@Override

public int[] getData() {

int[] mil = {this.zone, this.speed};

return mil;

}

}

7.7 Исходный код класса Toy

package Roboting;

public class Toy extends Robots {

protected int age;

protected int detail;

public Toy(String model, int mass, int energy, int age, int detail) {

super(model, mass, energy);

this.age = age;

this.detail = detail;

}

@Override

public String print() {

String text;

text = " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n|Модель робота: " + this.model + "\n|\n|Масса: "

+ this.mass + " кг\n|Энергия: " + this.energy + " Вт\n|Минимальный возраст: " + this.age + " лет\n|Количество деталей: " + this.detail + " ед\n|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

return text;

}

@Override

public int[] getData() {

int[] mil = {this.age, this.detail};

return mil;

}

}

Заключение

Программа была написана при помощи среды разработки NetBeans IDE. В ходе разработки программы были рассмотрены и применены на практике понятия класса и объекта, исследованы принципы инкапсуляции, наследования и полиморфизма.

После выполнения программы был составлен отчет, который соответствует всем требованиям.

# Список использованных источников

1. Объектно-ориентированное программирование. Метод.указания по выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во ИРНИТУ, 2015. – 23 c.
2. Программирование на Java для детей, родителей, дедушек и бабушек/ Файн Я.