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

**Институт информационных технологий**

КАФЕДРА ИНСТРУМЕТНАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИиППО)

Практические РАБОТы

по дисциплине «Программирование на языка Джава»

Выполнил студент группы ИКБО-02-21 Семянников Н.С.

Принял старший преподаватель *Рачков А.В.*

Практические работы работа выполнены «\_\_\_»\_\_\_\_\_\_\_2022г.

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_2022г.

Москва 2022

Оглавление

[Практическая работа № 1 3](#_Toc114178655)

[Практическая работа № 2 8](#_Toc114178656)

[Практическая работа № 3 9](#_Toc114178657)

[Практическая работа № 18 10](#_Toc114178658)

[Практическая работа № 19 11](#_Toc114178659)

[Практическая работа № 20 12](#_Toc114178660)

[Используемая литература 13](#_Toc114178661)

# Практическая работа № 1

## Цель работы

Изучить работу с классами в Java

## Теоретическое введение

Класс представляет из себя шаблон, или прототип, который определяет и описывает статические свойства и динамическое поведение, общие для всех объектов одного и того же вида.

Экземпляр класса - реализация конкретного объекта типа класс.

Переменные: содержит статические атрибуты класса

Методы: описывают динамическое поведение класса.

Переменные и методы, входящие в состав класса, формально называется переменные-поля данных класса и методы класса.

Для ссылки на переменную-поле данных класса или метод, вы должны:

* сначала создать экземпляр класса, который вам нужен;
* затем, использовать оператор точка “.” чтобы сослаться на элемент класса.

Конструктор – это специальный метод класса, который используется для создания и инициализации всех переменных-полей данных класса.

Перегрузка методов означает, что несколько методов могут иметь то же самое имя метод, но сами методы могут иметь различные реализации. Тем не менее, различные реализации должны быть различимы по списку их аргументов

Контроль за доступом осуществляется с помощью модификатора, он может быть использован для управления видимостью класса или переменных – полей или методов внутри класса.

* public: класс / переменная / метод доступными для всех других объектов в системе
* private: класс / переменная / метод доступными в пределах только этого класса

## Выполнение лабораторной работы

### Задание:

Необходимо реализовать простейший класс на языке программирования Java. Не забудьте добавить метод toString() к вашему классу. Так-же в программе необходимо предусмотреть класс-тестер для тестирования класса и вывода информации об объекте.

### Решение:

#### Класс «Собака»

public class Dog {

private String name;

private int age;

public Dog(String n, int a) {

name = n;

age = a;

}

public Dog(String n) {

name = n;

age = 0;

}

public Dog() {

name = "Pup";

age = 0;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String toString() {

return this.name + ", age " + this.age;

}

public void intoHumanAge() {

System.out.println(name + "'s age in human years is " + age \* 7 + " years");

}

}

#### Класс-тестер «Собака»

public class TestDog {

public static void main(String[] args) {

Dog d1 = new Dog("Mike", 3);

Dog d2 = new Dog("Helen", 7);

Dog d3 = new Dog("Bob");

d3.setAge(1);

System.out.println(d1);

d1.intoHumanAge();

d2.intoHumanAge();

d3.intoHumanAge();

}

}

#### Результат тестирования класса «Собака» представлен на рисунке 1

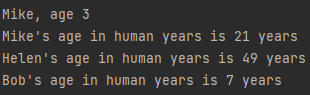


Рисунок 1 - Результат тестирования класса «Собака»

#### Класс «Мяч»

public class Circle {

private double radius;

private String color;

public Circle(double r, String cl) {

radius = r;

color = cl;

}

public Circle(double r) {

radius = r;

color = "Red";

}

public Circle() {

radius = 5;

color = "Red";

}

public double getRadius() {

return radius;

}

public void setRadius(double r) {

radius = r;

}

public String getColor() {

return color;

}

public void setColor(String cl) {

color = cl;

}

public String toString() {

return radius + " - " + color;

}

public void getArea() {

System.out.println(Math.PI \* Math.pow(radius, 2));

}

}

#### Класс-тестер «Мяч»

public class TestCircle {

public static void main(String[] args) {

Circle c1 = new Circle(3, "Green");

Circle c2 = new Circle(8);

Circle c3 = new Circle();

c1.setColor("Blue");

System.out.println(c1);

System.out.println(c2);

System.out.println(c3);

c2.getArea();

}

}

#### Результат тестирования класса «Мяч» представлен на рисунке 2

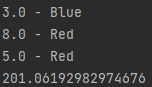


Рисунок 2 - Результат тестирования класса «Мяч»

#### Класс «Книга»

public class Book {

private int pages;

private String name;

public Book(int p, String n) {

pages = p;

name = n;

}

public Book(int p) {

pages = p;

name = "Lion The Fat";

}

public Book() {

pages = 200;

name = "Lion The Fat";

}

public int getPages() {

return pages;

}

public void setPages(int p) {

pages = p;

}

public String getName() {

return name;

}

public void setName(String n) {

name = n;

}

public String toString() {

return pages + " - " + name;

}

public void getWeight() {

System.out.println(pages \* 1.5 + " grams");

}

}

#### Класс-тестер «Книга»

public class TestBook {

public static void main(String[] args) {

Book b1 = new Book(500, "Slim Lion");

Book b2 = new Book(800);

Book b3 = new Book();

b2.setName("Happy Man");

System.out.println(b1);

System.out.println(b2);

System.out.println(b3);

b2.getWeight();

}

}

#### Результат тестирования класса «Книга» представлен на рисунке 3

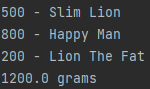


Рисунок 3 - Результат тестирования класса «Книга»

## Выводы по работе:

Изучил инкапсуляцию и работу с классами в Java.

# Практическая работа № 2

## Цель работы

Изучить обозначения и работу с UML диаграммами.

## Теоретическое введение

Средствами UML в виде диаграмм можно графически изобразить класс и экземпляр класса.

Графически представляем класс в виде прямоугольника, разделенного на три области – область именования класса, область инкапсуляции данных и область операций.

Имя: определяет класс.

Переменные: содержит статические атрибуты класса, или описывают свойства класса

Методы: описывают динамическое поведение класса.

На рисунке 4 приведен общий вид UML диаграммы класса.



Рисунок 4 – UML

## Выполнение лабораторной работы

### Задание:

Необходимо написать программу, которая состоит из двух классов Author и TestAuthor.

Написать программу, которая состоит из двух классов. Один из них Ball должен реализовывать сущность мяч, а другой с названием TestBall тестировать работу созданного класса.

### Решение:

#### Класс «Автор»

package T1;

public class Author {

private String name, email;

private char gender;

public Author(String name, String email, char gender) {

this.name = name;

this.email = email;

if (gender != 'm' && gender != 'f')

gender = 'u';

this.gender = gender;

}

public String getName() {

return name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public char getGender() {

return gender;

}

public String toString() {

return name + "(" + (gender == 'f' ? "ms" : gender) + ") at " + email;

}

}

#### Класс-тестер «Автор»

package T1;

public class TestAuthor {

public static void main(String[] args) {

Author a1 = new Author("Ivan Popov", "ivan@mail.ru", 'm');

System.out.println(a1.getName());

System.out.println(a1.getEmail());

System.out.println(a1.getGender());

a1.setEmail("ivanov@yandex.ru");

System.out.println(a1);

}

}

#### Результат тестирования класса «Автор» представлен на рисунке 5

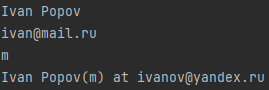


Рисунок 5 - Результат тестирования класса «Автор»

#### Класс «Мяч»

package T2;

public class Ball {

private double x = 0.0, y = 0.0;

public Ball(double x, double y) {

this.x = x;

this.y = y;

}

public Ball() {

x = 0;

y = 0;

}

public double getX() {

return x;

}

public void setX(double x) {

this.x = x;

}

public double getY() {

return y;

}

public void setY(double y) {

this.y = y;

}

public void setXY(double x, double y) {

this.x = x;

this.y = y;

}

public void move(double xDisp, double yDisp) {

x += xDisp;

y += yDisp;

}

public String toString() {

return "Lab2.T2.Ball @ (" + x + ", " + y + ")";

}

}

#### Класс-тестер «Мяч»

package T2;

public class TestBall {

public static void main(String[] args) {

Ball b = new Ball(3,0);

System.out.println(b);

b.move(10, 7);

System.out.println(b);

}

}

#### Результат тестирования класса «Мяч» представлен на рисунке 6



Рисунок 6 - Результат тестирования класса «Мяч»

## Выводы по работе:

Научился писать классы по UML диаграммам.

# Практическая работа № 3

## Цель работы

## Теоретическое введение

## Выполнение лабораторной работы

### Задание:

Если нет конкретного, сформулировать самостоятельно.

### Решение:

Здесь могут быть схемы. коды, скриншоты и т.п. Обращай те внимание на подписи таблиц (вверху по левому краю) и рисунков (снизу по центру). Перед их появление, в тексте должна быть ссылка. Весь отчет оформляется в соответствии с ГОСТ.

## Выводы по работе:

Делается кратко по каждой работе.

# Практическая работа № 18

## Цель работы

Целью работы является изучение работы циклов, ввода/вывода, создание массивов и методов в Java.

## Теоретическое введение

Язык Java — это объектно-ориентированный язык программирования.

Программы написанные на Java могут выполняться на различных операционных системах при наличии необходимого ПО - Java Runtime Environment.

Чтобы объявить переменную, необходимо указать тип переменной и её имя.

Массив — это конечная последовательность упорядоченных элементов одного типа, доступ к каждому элементу в которой осуществляется по его индексу.

Условие — это конструкция, позволяющая выполнять то или другое действие, в зависимости от логического значения, указанного в условии.

Цикл — это конструкция, позволяющая выполнять определенную часть кода несколько раз. В Java есть три типа циклов for, while, do while.

Для ввода данных используется класс Scanner из библиотеки пакетов.

Методы позволяют выполнять блок кода, из любого другого места, где это доступно. Методы определяются внутри классов.

Методы могут быть статическими (можно выполнять без создания экземпляра класса), не статическими (не могут выполняться без создания экземпляра класса). Методы могут быть открытыми(public), закрытыми(private). Закрытые методы могут вызываться только внутри того класса, в котором они определены. Открытые методы можно вызывать для объекта внутри других классов.

## Выполнение лабораторной работы

### Задание:

Сгенерировать массив целых чисел случайным образом, вывести его на экран, отсортировать его, и снова вывести на экран. (Вариант 4)

### Решение:

#### Класс «RandomArray»

import java.util.Arrays;

import java.util.Random;

import java.util.Scanner;

public class RandomArray {

public static void main(String[] args) {

int n;

System.out.print("Enter array size: ");

Scanner sc = new Scanner(System.in);

n = sc.nextInt();

Random rn = new Random();

int[] a = new int[n];

System.out.println("Original");

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

a[i] = rn.nextInt(1, 30);

System.out.print(a[i]+" ");

}

System.out.println();

Arrays.sort(a);

System.out.println("Sorted");

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

System.out.print(a[i]+" ");

}

}

}

#### Результат работы класса «RandomArray» представлен на рисунке 7.

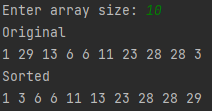


Рисунок 7 - Результат работы класса «RandomArray»

## Выводы по работе:

Изучил работу циклов и создание массива случайных чисел в Java.

# Практическая работа № 19

## Цель работы

Научиться создавать классы.

## Теоретическое введение

Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные.

Объект — это экземпляр класса.

Данные объекта скрыты от остальной программы.

Сокрытие данных называется инкапсуляцией.

Полиморфизм подтипов — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

Для того чтобы создать класс в языке Java необходимо создать файл с расширением java. Имя файла должно быть таким же, как и имя создаваемого класса.

В теле класса можно описать методы, переменные, константы, конструкторы класса.

Для того чтобы создать экземпляр класса необходимо объявить переменную, тип которой соответствует имени класса или суперкласса. После чего нужно присвоить этой переменной значение, вызвав конструктор создаваемого класса с помощью оператора new.

## Выполнение лабораторной работы

### Задание:

Создать класс, описывающий книгу (Book). В классе должны быть описаны нужные свойства книги (автор, название, год написания и т. д.) и методы для получения, изменения этих свойств. Протестировать работу класса в классе BookTest, содержащим метод статический main(String[] args).

### Решение:

#### Класс «Book»

package V3;

public class Book {

private String author, name;

private int year = 1999, pages = 100;

public Book(String author, String name, int year, int pages) {

this.author = author;

this.name = name;

this.year = year;

this.pages = pages;

}

public Book(String author, String name, int year) {

this.author = author;

this.name = name;

this.year = year;

}

public Book(String author, String name) {

this.author = author;

this.name = name;

}

public String getName() {

return name;

}

public int getPages() {

return pages;

}

public int getYear() {

return year;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public void setName(String name) {

this.name = name;

}

public void setPages(int pages) {

this.pages = pages;

}

public void setYear(int year) {

this.year = year;

}

@Override

public String toString() {

return "Book {" +

"author='" + author + '\'' +

", name='" + name + '\'' +

", year=" + year +

", pages=" + pages +

'}';

}

}

#### Класс-тестер «TestBook»

package V3;

public class TestBook {

public static void main(String[] args) {

Book b1 = new Book("John Smith", "Atlantis", 2005, 520);

Book b2 = new Book("John Smith", "Atlantis");

System.out.println("Pages: " + b2.getPages());

b2.setName("Capital");

System.out.println(b2 + "\n" + b1);

}

}

#### Результат тестирования класса «Book» представлен на рисунке 8

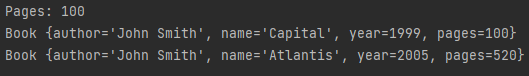


Рисунок 8 - Результат тестирования класса «Book»

## Выводы по работе:

Научился создавать классы, методы и назначать уровень доступа к методам/переменным

# Практическая работа № 20

## Цель работы

Изучить наследование в Java

## Теоретическое введение

Одним из ключевых аспектов объектно-ориентированного программирования является наследование. С помощью наследования можно расширить функционал уже имеющихся классов за счет добавления нового функционала или изменения старого.

Чтобы объявить один класс наследником от другого, надо использовать после имени класса-наследника ключевое слово extends, после которого идет имя базового класса.

С помощью ключевого слова super мы можем обратиться к любому члену базового класса - методу или полю, если они не определены с модификатором private.

Можно запретить наследование с помощью ключевого слова final.

Кроме обычных классов в Java есть абстрактные классы.

Абстрактный класс похож на обычный класс. В абстрактном классе также можно определить поля и методы, в то же время нельзя создать объект или экземпляр абстрактного класса. Абстрактные классы призваны предоставлять базовый функционал для классов- наследников. А производные классы уже реализуют этот функционал.

При определении абстрактных классов используется ключевое слово abstract.

Кроме обычных методов абстрактный класс может содержать абстрактные методы. Такие методы определяются с помощью ключевого слова abstract и не имеют никакого функционала. Производный класс обязан переопределить и реализовать все абстрактные методы, которые имеются в базовом абстрактном классе.

## Выполнение лабораторной работы

### Задание:

Создать абстрактный класс, описывающий мебель. С помощью наследования реализовать различные виды мебели. Также создать класс FurnitureShop, моделирующий магазин мебели. Протестировать работу классов

### Решение:

#### Абстрактный класс «Furniture»

abstract class Furniture {

protected String material, manufacturer;

protected int count;

protected double price, width, height, depth, weight;

public Furniture() {

material = "Wood";

manufacturer = "IKEA";

price = 1000;

count = 1;

width = 100;

height = 50;

depth = 100;

weight = 20;

}

public double getWidth() {

return width;

}

public void setWidth(double width) {

this.width = width;

}

public double getHeight() {

return height;

}

public void setHeight(double height) {

this.height = height;

}

public double getWeight() {

return weight;

}

public void setWeight(double weight) {

this.weight = weight;

}

public double getDepth() {

return depth;

}

public void setDepth(double depth) {

this.depth = depth;

}

public String getManufacturer() {

return manufacturer;

}

public void setManufacturer(String manufacturer) {

this.manufacturer = manufacturer;

}

public String getMaterial() {

return material;

}

public void setMaterial(String material) {

this.material = material;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public abstract String toString();

}

#### Класс «Table»

public class Table extends Furniture {

private int legs;

private boolean isAngled;

public Table() {

legs = 4;

isAngled = true;

}

public Table(int l, boolean isA) {

legs = l;

isAngled = isA;

}

public int getLegs() {

return legs;

}

public boolean isAngled() {

return isAngled;

}

public void setAngled(boolean angled) {

isAngled = angled;

}

public void setLegs(int legs) {

this.legs = legs;

}

@Override

public String toString() {

return "Table {" +

"legs=" + legs +

", width=" + width +

", height=" + height +

", depth=" + depth +

", weight=" + weight +

", angled=" + isAngled +

", material='" + material + '\'' +

", manufacturer='" + manufacturer + '\'' +

", count=" + count +

", price=" + price +

'}';

}

}

#### Класс «Chair»

public class Chair extends Furniture {

private int legs, maxWeight;

private String seatMaterial;

private boolean isRolling;

public Chair() {

legs = 5;

maxWeight = 150;

seatMaterial = "Leather";

isRolling = true;

}

public Chair(int l, int maxW, String sm, boolean isR) {

legs = l;

maxWeight = maxW;

seatMaterial = sm;

isRolling = isR;

}

public int getLegs() {

return legs;

}

public void setLegs(int legs) {

this.legs = legs;

}

public int getMaxWeight() {

return maxWeight;

}

public void setMaxWeight(int maxWeight) {

this.maxWeight = maxWeight;

}

public String getSeatMaterial() {

return seatMaterial;

}

public void setSeatMaterial(String seatMaterial) {

this.seatMaterial = seatMaterial;

}

public boolean isRolling() {

return isRolling;

}

public void setRolling(boolean rolling) {

isRolling = rolling;

}

@Override

public String toString() {

return "Chair {" +

"legs=" + legs +

", isRolling=" + isRolling +

", maxWeight=" + maxWeight +

", main material='" + material + '\'' +

", seat material='" + seatMaterial + '\'' +

", manufacturer='" + manufacturer + '\'' +

", count=" + count +

", price=" + price +

", width=" + width +

", height=" + height +

", depth=" + depth +

", weight=" + weight +

'}';

}

}

#### Класс «Sofa»

public class Sofa extends Furniture {

private int doors, sections;

private boolean hangBar;

public Sofa() {

doors = 2;

sections = 4;

hangBar = true;

}

public Sofa(int d, int s, boolean hB) {

doors = d;

sections = s;

hangBar = hB;

}

public int getDoors() {

return doors;

}

public void setDoors(int doors) {

this.doors = doors;

}

public int getSections() {

return sections;

}

public void setSections(int sections) {

this.sections = sections;

}

public boolean isHangBar() {

return hangBar;

}

public void setHangBar(boolean hangBar) {

this.hangBar = hangBar;

}

@Override

public String toString() {

return "Sofa {" +

"doors=" + doors +

", sections=" + sections +

", hangBar=" + hangBar +

", material='" + material + '\'' +

", manufacturer='" + manufacturer + '\'' +

", count=" + count +

", price=" + price +

", width=" + width +

", height=" + height +

", depth=" + depth +

", weight=" + weight +

'}';

}

}

#### Класс «FurnitureShop»

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class FurnitureShop {

public static class Defaults {

public double width = 100, height = 50, depth = 100, price = 1000;

public String manu = "IKEA", material = "Wood";

public int count = 1;

@Override

public String toString() {

return "Defaults {" +

"width=" + width +

", height=" + height +

", depth=" + depth +

", material='" + material + '\'' +

", manu='" + manu + '\'' +

", count=" + count +

", price=" + price +

'}';

}

}

public static void main(String[] args) {

Defaults dp = new Defaults(); //Для редактирования значений, которые в классе абстракции

List<Furniture> fn = new ArrayList<Furniture>();

Scanner sc = new Scanner(System.in);

String a; //а - выбор, n - для проверки на размер

while (true) {

System.out.println("1) Add new");

System.out.println("2) Show");

System.out.println("3) Change defaults");

System.out.println("4) Exit");

System.out.print("Choose action: ");

a = sc.next();

switch (a) {

case "1" -> {

while (a != "4")

a = AddFurniture(fn, sc, dp); //Костыль, чтобы создать бесконечный цикл для добавления

}

case "2" -> {

ShowFurniture(fn); //Вывод мебели

}

case "3" -> {

ChangeDefaults(dp, sc); //Изменить значения абстракции

}

case "4" -> {

System.exit(0); //Выход

}

}

}

}

private static void ChangeDefaults(Defaults dp, Scanner sc) {

System.out.println(dp);

System.out.print("Width: ");

dp.width = sc.nextDouble();

System.out.print("Height: ");

dp.height = sc.nextDouble();

System.out.print("Depth: ");

dp.depth = sc.nextDouble();

System.out.print("Material: ");

dp.material = sc.next();

System.out.print("Manu: ");

dp.manu = sc.next();

System.out.print("Count: ");

dp.count = sc.nextInt();

System.out.print("Price: ");

dp.price = sc.nextDouble();

}

private static void AddTable(List<Furniture> fn, Scanner sc) {

System.out.println("Legs(number), is angled(1-T, 0-F)");

fn.add(new Table(sc.nextInt(), sc.nextInt() == 1));

}

private static void AddChair(List<Furniture> fn, Scanner sc) {

System.out.println("Legs(n>2), max weight(n>50), seat material, is rolling(1-T, 0-F)");

fn.add(new Chair(sc.nextInt(), sc.nextInt(), sc.next(), sc.nextInt() == 1));

}

private static void AddSofa(List<Furniture> fn, Scanner sc) {

System.out.println("Doors(number), sections(number), has hang bar(1-T, 0-F)");

fn.add(new Sofa(sc.nextInt(), sc.nextInt(), sc.nextInt() == 1));

}

private static void ShowFurniture(List<Furniture> fn) {

for (Furniture furniture : fn) {

System.out.println(furniture);

}

}

static String AddFurniture(List<Furniture> fn, Scanner sc, Defaults dp) {

System.out.println("1) New table");

System.out.println("2) New chair");

System.out.println("3) New sofa");

System.out.println("4) Back");

System.out.print("Choose action: ");

switch (sc.next()) {

case "1" -> {

AddTable(fn, sc);

ApplyDefaults(fn.get(fn.size() - 1), dp);

}

case "2" -> {

AddChair(fn, sc);

ApplyDefaults(fn.get(fn.size() - 1), dp);

}

case "3" -> {

AddSofa(fn, sc);

ApplyDefaults(fn.get(fn.size() - 1), dp);

}

case "4" -> {

return "4";

}

}

return "1";

}

private static void ApplyDefaults(Furniture fn, Defaults dp) {

fn.setWidth(dp.width);

fn.setHeight(dp.height);

fn.setDepth(dp.depth);

fn.setMaterial(dp.material);

fn.setManufacturer(dp.manu);

fn.setCount(dp.count);

fn.setPrice(dp.price);

}

}

#### Результат:

Главное меню представлено на рисунке 9

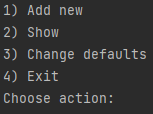


Рисунок 9 - Главное меню

Выбор добавляемого объекта представлен на рисунке 10

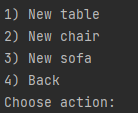


Рисунок 10 - Выбор добавления

Добавление стола представлено на рисунке 11

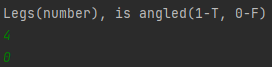


Рисунок 11 - Добавляем стол

Добавление стула представлено на рисунке 12

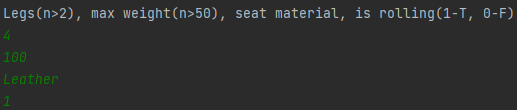


Рисунок 12 - Добавляем стул

Просмотр всех объектов представлен на рисунке 13

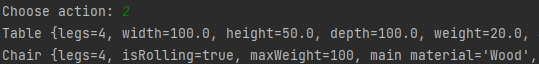


Рисунок 13 - Просмотр объектов

Изменение стандартных (абстрактных) значений представлено на рисунке 14

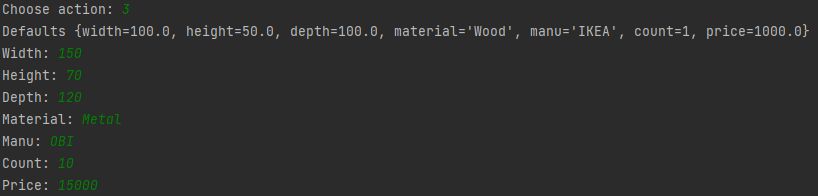


Рисунок 14 - Изменение стандартных значений

## Выводы по работе:

Изучил работу наследования и абстракции в Java.

# Используемая литература

*Список литературы тоже должен быть оформлен по ГОСТ. Первым пунктом можно указать:*

1. *Конспект лекций по дисциплине «Программирование на языке Джава», РТУ МИРЭА, лектор – старший преподаватель Зорина Н.В.*
2. *Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. –СПб.: Питер, 2013. – 240 с.*
3. *Фрэйн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств – Питер: 2016г. Режим доступа свободный: https://www.htbook.ru/kompjutery\_i\_seti/setevye\_tekhnologii/html5-i-css3-razrabotka-sajtov-dlja-ljubykh-brauzerov-i-ustrojstv;*
4. *Справочник по языку PHP [Электронный ресурс]:php.su— Режим доступа свободный: http://www.php.su;*
5. *И т.д.*