Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных Технологий и Анализа Данных

Кафедра вычислительной техники

**Название работы** – Разработка классов с использованием механизмов наследования, полиморфизма и инкапсуляции.

Отчет по лабораторной работе № 4

Вариант 9

по дисциплине Объектно-ориентированное программирование

Выполнил

Студент, номер группы ИСМб-19-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Д. Солопов

Дата: \_\_\_\_\_\_\_\_\_\_

Принял

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Маланова

Дата: \_\_\_\_\_\_\_\_\_\_

Иркутск 2020

Содержание

[1 Постановка задачи 3](#_Toc54718772)

[2 Проектирование классов 6](#_Toc54718773)

[3 Описание структуры пользовательского меню 8](#_Toc54718774)

[4 Таблица тестов 10](#_Toc54718775)

[5 Листинг исходного кода 14](#_Toc54718776)

[6 Заключение 19](#_Toc54718777)

[7 Список использованных источников 20](#_Toc54718778)

1 Постановка задачи

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) являют-ся абстрактными и служат для выделения общих данных и поведения для дру-гих классов. Абстрактный класс должен содержать как минимум один абст-рактный 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 за-дания необходимо создать некоторое количество объектов, добавить их в груп-пу используя предусмотренные методы класса-«группы» и для каждого из них вызвать унаследованный метод.

**Вариант индивидуального задания**

9. хлебо-булочное изделие, товар, молочный продукт, магазин;

**Вопросы по работе:**

1. Какие проблемы могут возникнуть, если не написать @Override?

**Ответ:**

Среда разработки выдаст ошибку, программа не будет компилироваться. Это произойдёт из-за того, что абстрактные методы класса не будут переопределены, а переопределение абстрактных методов класса обязательно.

2. Почему класс Product определен как абстрактный?

**Ответ:**

Класс Product содержит общие элементы для обоих классов BakeryProduct и MilkProduct соответственно. Чтобы не повторять методы и поля в каждом классе, имеет смысл их обобщить и вывести в отдельный класс, от которого уже будут наследоваться классы с общими характеристиками.

3. Как нужно обращаться к полям класса-предка?

**Ответ:**

С помощью ключевого слова super. Данное ключевое слово это аналог ключевого слова this, однако для базового класса. Ключевое слово super позволяет обращаться к полям и методам(в т.ч. и конструкторам) базового класса.

4. В чем заключается полиморфизм? и как он проявляется в Вашей программе?

**Ответ:**

Полиморфизм заключается в переопределении поведения объектов классов, которые схожи между собой, но имеют внутреннюю уникальную реализацию. Например, коммуникация между животными. “Выдача” голоса животным предполагает разную реализацию, в силу того, что животные могут по разному произносить звуки, однако общее поведение у всех животных есть – они могут произносить звук.

В моей программе полиморфизм реализуется с помощью переопределения методов объекта базового класса, для каждого унаследованного объекта уникальным образом, относительно своего контекста и смысловой нагрузки.

5. В каком отношении находятся классы Product и Store, и как это отношение реализуется в Вашей программе?

**Ответ:**

Классы Product и Store не имеют никаких между собой отношений, однако класс Store находится в отношениях “имеет”(т.е. ассоциативности) с классами BakeryProduct и MilkProduct, у которых свои особенности реализации и свои отличительные черты между собой. Взаимодествие с методами и полями данных классов не уместно производить с помощью объектов класса Product, в силу того, что каждый класс (BakeryProduct и MilkProduct) имеют свою уникальную внутреннюю реализацию, а базовый класс эту реализацию не видит, поэтому при приведении объекта типа Bakery/MilkProduct к типу Product получится “урезанная” реализация этих классов, что будет влиять на интерфейс программы, а, следовательно, ассоциативные отношения класса Store с Bakery/MilkProduct вполне могут быть оправданы. Также, стоит учесть, что класс Product абстрактный, а значит объект этого класса не может быть создан.

6. В каком отношении находятся классы Product и MilkProduct как это отношение реализуется в Вашей программе?

**Ответ:**

Абстрактный класс Product содержит общие элементы для классов BakeryProduct и MilkProduct, а следовательно, Product – базовый класс по отношению к классу MilkProduct.

7. Разложите классы по отдельным пакетам по какому-либо признаку классификации

**Ответ(в виде кода и некоторых пояснений):**

На основе собственной программы я разложил в разные пакеты разные классы. В пакет bakery я внёс производный класс BakeryProduct, наследуемый от Product

package bakery;

import com.company.Product;

public class BakeryProduct extends Product {

//…

В пакет milk я внёс производный класс MilkProduct, наследуемый от Product

package milk;

import com.company.Product;

public class MilkProduct extends Product {

//…

В пакете com.company я оставил класс Product и класс Main, по содержимому файлов MilkProduct и BakeryProduct можно судить об этом из-за импорта класса Product из пакета com.company.

8. Ответьте на теоретические вопросы:

-Абстрактный класс, абстрактные методы. Переопределение метода.

-Реализация интерфейса, множественное наследование (покажите пример множественного наследования на основе Вашей программы).

**Ответ:**

**Абстрактный класс** – базовый класс, который не предполагает создания экземпляров.

**Абстрактные методы** – методы абстрактного класса, не содержащие блок кода реализации, обязательно должны быть переопределены в производных классах. Однако, можно не переопределять абстрактные методы, но тогда производный класс должен быть абстрактным.

**Переопределение метода** – переопределение метода виртуальной / абстрактной функции базового класса в производном классе, с целью обеспечения производного класса уникальным(или не уникальным) для него поведением.

**Множественное наследование** — свойство, поддерживаемое частью объектно-ориентированных языков программирования, когда класс может иметь более одного суперкласса.

Пример реализации множественного наследования на основе моей программы:

Класс Product:

package com.company;

public abstract class Product {

//…

Класс MilkProduct(наследник от Product):

package milk;

import com.company.Product;

public class MilkProduct extends Product {

//...

Класс Kefir(наследник от MilkProduct):

package milk;

public class Kefir extends MilkProduct{

//...

2 Проектирование классов

Программа содержит 4 класса, один из которых – Main, является точкой входа в программу.

Описание классов:

1. Main – точка входа в программу
2. Product – абстрактный класс содержащий общие атрибуты и методы работы с товарами.
3. BakeryProduct – класс хлебо-булочных изделий наследующий атрибуты и методы абстрактного класса Product.
4. MilkProduct – класс молочных продуктов наследующий атрибуты и методы абстрактного класса Product
5. Store – класс представляющий интерфейс управления магазином, который позволяет добавлять молочные или хлебо-булочные изделия в магазин и просматривать все или определённые товары.

Класс Main работает только с классом Store.

Словесное описание класса Main:

Класс Main содержит в себе один статический метод main(), который является точкой входа в программу.

Словесное описание класса Product:

Класс абстрактный. Класс содержит одно поле типа double - price, с спецификатором доступа protected, которое указывает на цену определённого товара. Класс содержит два поля типа String: nameProduct и compositionProduct, которые идентифицируют товар по названию и составу соответственно. Оба поля имеют спецификатор доступа protected. Класс содержит следующие абстрактные методы: setPrice() – для задания продукта, getPrice() – для получения цены продукта, setNameProduct() – для задания имени продукта, getNameProduct() – для получения имени продукта, setCompositionProduct() – для задания состава продукта и getCompositionProduct() – для получения цены продукта. Также, абстрактный класс содержит конструктор без параметров, задающий начальные значения атрибутам.

Словесное описание класса BakeryProduct:

Класс не абстрактный, наследник абстрактного класса Product. Класс содержит одно поле типа String – form, с спецификатором доступа private, обозначающее форму продукта(форму хлебо-булочного изделия). Класс содержит перегрузки всех абстрактных методов класса Product, а также содержит методы для задания/получения формы продукта: setFormProduct() и getFormProduct(). Класс содержит конструктор без параметров, в котором вызывается конструктор суперкласса, т.е. конструктор класса Product.

Словесное описание класса MilkProduct:

Класс не абстрактный, наследник абстрактного класса Product. Класс содержит одно поле типа String – storageForm, с спецификатором доступа private, обозначающее форму хранения молочного продукта(форму в которой хранится молочный продукт). Класс содержит перегрузки всех абстрактных методов класса Product, а также содержит методы для задания/получения формы хранения продукта: setStorageFormProduct() и getStorageFormProduct().Класс содержит конструктор без параметров, в котором вызывается конструктор суперкласса, т.е. конструктор класса Product.

Словесное описание класса Store:

Класс не абстрактный. Класс содержит два поля с типом автоматически расширяемого массива ArrayList(один из видов списка) для объектов класса BakeryProduct и MilkProduct: bakeries и milks соответственно. Оба поля класса имеют спецификатор доступа private. Класс содержит в себе реализацию пользовательского меню и группирует все классы, согласно заданию. Класс содержит следующие методы: display() – для вывода пользовательского меню, outputBakeryProduct() – для вывода информации о всех хлебо-булочных изделий, outputMilksProduct() – для вывода информации о всех молочных продуктах, addBakeryProduct() – для добавления хлебо-булочного изделия, addMilkProduct – для добавления молочного продукта.

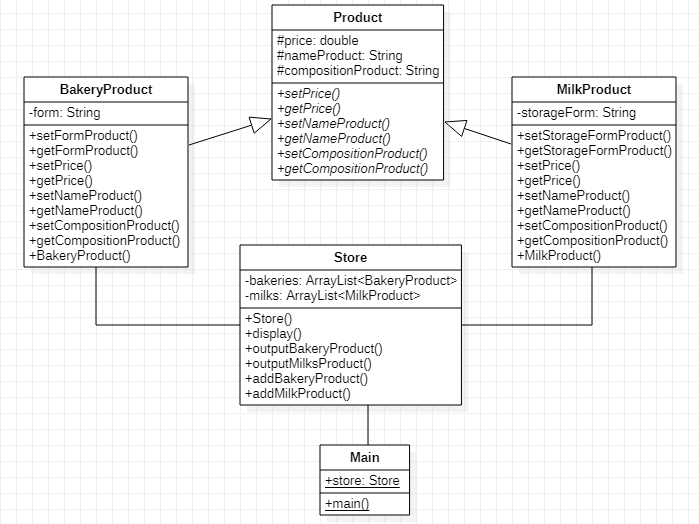


Рисунок 1 – Диаграмма классов UML

3 Описание структуры пользовательского меню

Пользовательское меню реализует класс Store. Весь процесс взаимодействия пользователя с меню осуществляется с помощью функции display(). Пользовательское меню содержит 6 пунктов:

1. Вывод всех продуктов
2. Вывод хлебо-булочных продуктов
3. Вывод молочных продуктов
4. Добавить хлебо-булочный продукт
5. Добавить молочный продукт
6. Выход

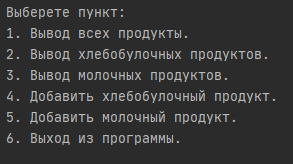


Рисунок 2 – Пункты пользовательского меню

В случае, если пользователь ввёл число не попадающее в диапазон [1; 6], программа предупреждает об этом пользователя и пользователь снова вводит число.

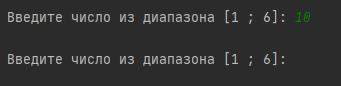


Рисунок 3 – Введен не корректный номер пункта

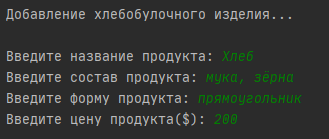


Рисунок 4 – Пример добавления хлебо-булочного изделия

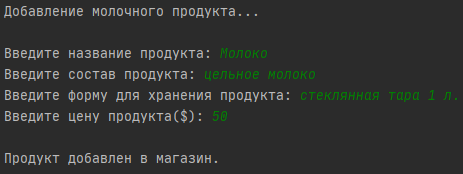


Рисунок 5 – Пример добавления молочного продукта

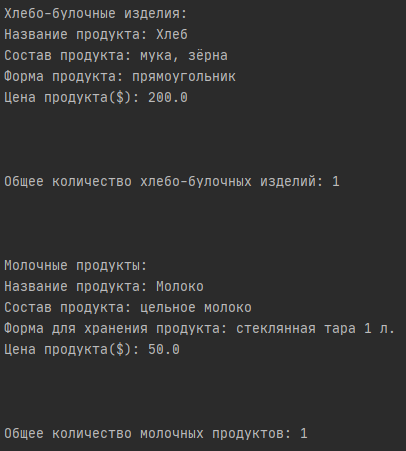


Рисунок 6 – Пример вывода информации о всех продуктах в магазине

4 Таблица тестов

Таблица 1 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Добавление молочного продукта и вывод молочных продуктов | (рис. 7) | (рис. 8) |
| 2 | Добавление хлебо-булочного продукта и вывод хлебо-булочных продуктов | (рис. 9) | (рис. 10) |
| 3 | Добавление хлебо-булочного продукта и вывод молочных продуктов(когда их не было добавлено) | (рис. 11) | (рис. 12) |
| 4 | Добавление молочного продукта и вывод всех продуктов | (рис. 13) | (рис. 14) |
| 5 | Добавление хлебо-булочного и молочного продукта, с выводом всех продуктов | (рис. 15-16) | (рис. 17) |

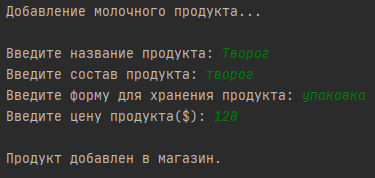


Рисунок 7– Добавление молочного продукта(тест 1 из таблицы тестов 1)

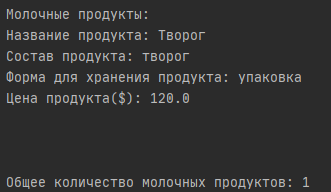


Рисунок 8 – Результат теста 1 из таблицы тестов 1

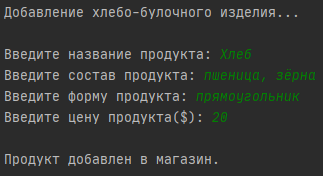


Рисунок 9 - Добавление хлебо-булочного продукта(тест 2 из таблицы тестов 1)

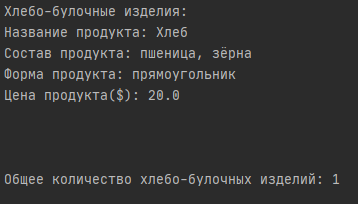


Рисунок 10 – Результат теста 2 из таблицы тестов 1

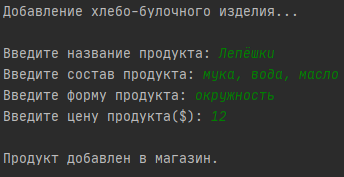


Рисунок 11 - Добавление хлебо-булочного продукта(тест 3 из таблицы тестов 1)



Рисунок 12 – Результат теста 3 из таблицы тестов 1

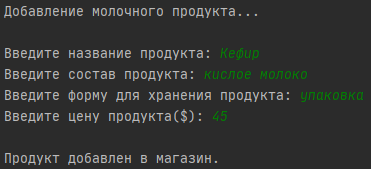


Рисунок 13 - Добавление молочного продукта(тест 4 из таблицы тестов 1)

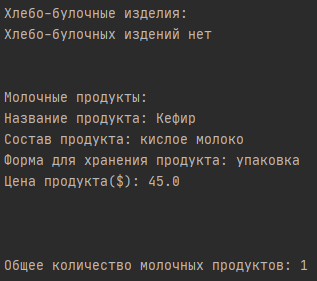


Рисунок 14 – Результат теста 4 из таблицы тестов 1

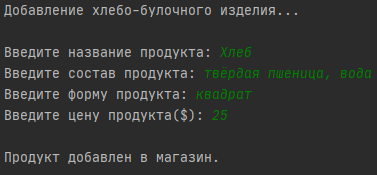


Рисунок 15 - Добавление хлебо-булочного продукта(тест 5 из таблицы тестов 1)

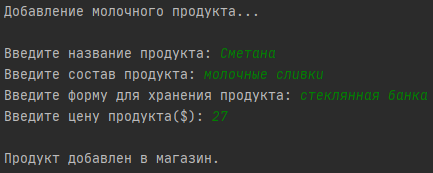


Рисунок 16 - Добавление молочного продукта(тест 5 из таблицы тестов 1)

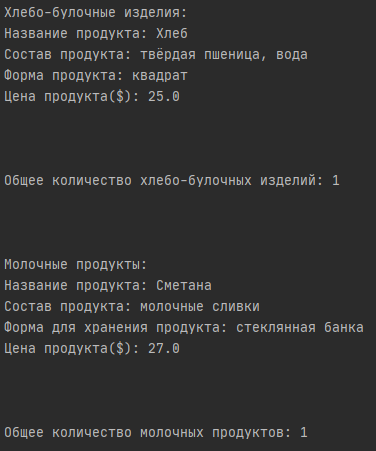


Рисунок 17 – Результат теста 5 из таблицы тестов 1

5 Листинг исходного кода

Класс Product:

package com.company;

public abstract class Product {

protected double price; //цена продукта

protected String nameProduct; //название продукта

protected String compositionProduct; //состав продукта

public Product(){

price = 0;

nameProduct = "";

compositionProduct = "";

}

public abstract boolean setPrice(double val);

public abstract double getPrice();

public abstract void setNameProduct(String n);

public abstract String getNameProduct();

public abstract void setCompositionProduct(String c);

public abstract String getCompositionProduct();

}

Класс BakeryProduct:

package com.company;

public class BakeryProduct extends Product{

private String form; //форма хлебо-булочного продукта

BakeryProduct(){

super(); //вызов базового конструктора

form = "";

}

public void setFormProduct(String f){

this.form = f;

}

public String getFormProduct(){

return this.form;

}

@Override

public boolean setPrice(double val) {

if(val <= 0)

return false;

super.price = val;

return false;

}

@Override

public double getPrice() {

return super.price;

}

@Override

public void setNameProduct(String n) {

super.nameProduct = n;

}

@Override

public String getNameProduct() {

return super.nameProduct;

}

@Override

public void setCompositionProduct(String c) {

super.compositionProduct = c;

}

@Override

public String getCompositionProduct() {

return compositionProduct;

}

}

Класс MilkProduct:

package com.company;

public class MilkProduct extends Product{

public String storageForm; //форма для хранения молочных продуктов

public MilkProduct(){

super();

storageForm = "";

}

public void setStorageFormProduct(String f){

this.storageForm = f;

}

public String getStorageFormProduct(){

return this.storageForm;

}

@Override

public boolean setPrice(double val) {

if(val <= 0)

return false;

super.price = val;

return false;

}

@Override

public double getPrice() {

return super.price;

}

@Override

public void setNameProduct(String n) {

super.nameProduct = n;

}

@Override

public String getNameProduct() {

return super.nameProduct;

}

@Override

public void setCompositionProduct(String c) {

super.compositionProduct = c;

}

@Override

public String getCompositionProduct() {

return compositionProduct;

}

}

Класс Store:

package com.company;

import java.util.ArrayList;

import java.util.Scanner;

//Магазин

public class Store {

private ArrayList<BakeryProduct> bakeries = null; //хлебо-булочные изделия

private ArrayList<MilkProduct> milks = null; //молочные продуктов

public Store(){

bakeries = new ArrayList<>();

milks = new ArrayList<MilkProduct>();

}

public void display(){

Scanner sc = new Scanner(System.in);

int number = 0;

do{

System.out.print("\n\nВыберете пункт:\n");

System.out.print("1. Вывод всех продукты.\n");

System.out.print("2. Вывод хлебо-булочных продуктов.\n");

System.out.print("3. Вывод молочных продуктов.\n");

System.out.print("4. Добавить хлебо-булочный продукт.\n");

System.out.print("5. Добавить молочный продукт.\n");

System.out.print("6. Выход из программы.\n");

number = sc.nextInt();

while((number < 1) || (number > 6)){

System.out.print("\nВведите число из диапазона [1 ; 6]: ");

number = sc.nextInt();

}

switch(number){

case 1:{

this.outputBakeryProduct();

this.outputMilksProduct();

break;

}

case 2:{

this.outputBakeryProduct();

break;

}

case 3:{

this.outputMilksProduct();

break;

}

case 4:{

this.addBakeryProduct();

break;

}

case 5:{

this.addMilkProduct();

break;

}

}

}while(number != 6);

}

public void outputBakeryProduct(){

System.out.print("\nХлебо-булочные изделия: \n");

if(bakeries.size() == 0){

System.out.print("Хлебо-булочных издений нет\n\n");

return;

}

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

System.out.println("Название продукта: " + bakeries.get(i).getNameProduct());

System.out.println("Состав продукта: " + bakeries.get(i).getCompositionProduct());

System.out.println("Форма продукта: " + bakeries.get(i).getFormProduct());

System.out.println("Цена продукта($): " + bakeries.get(i).getPrice());

System.out.print("\n\n");

}

System.out.println("\nОбщее количество хлебо-булочных изделий: " + bakeries.size() + "\n\n");

}

public void outputMilksProduct(){

System.out.print("\nМолочные продукты: \n");

if(milks.size() == 0){

System.out.print("Молочных продуктов нет\n\n");

return;

}

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

System.out.println("Название продукта: " + milks.get(i).getNameProduct());

System.out.println("Состав продукта: " + milks.get(i).getCompositionProduct());

System.out.println("Форма для хранения продукта: " + milks.get(i).getStorageFormProduct());

System.out.println("Цена продукта($): " + milks.get(i).getPrice());

System.out.print("\n\n");

}

System.out.println("\nОбщее количество молочных продуктов: " + milks.size() + "\n\n");

}

public void addBakeryProduct(){

System.out.print("\n\nДобавление хлебо-булочного изделия... \n\n");

BakeryProduct product = new BakeryProduct();

Scanner sc = new Scanner(System.in);

System.out.print("Введите название продукта: ");

product.setNameProduct(sc.nextLine());

System.out.print("Введите состав продукта: ");

product.setCompositionProduct(sc.nextLine());

System.out.print("Введите форму продукта: ");

product.setFormProduct(sc.nextLine());

System.out.print("Введите цену продукта($): ");

product.setPrice(sc.nextDouble());

bakeries.add(product);

System.out.print("\nПродукт добавлен в магазин.");

}

public void addMilkProduct(){

System.out.print("\n\nДобавление молочного продукта... \n\n");

MilkProduct product = new MilkProduct();

Scanner sc = new Scanner(System.in);

System.out.print("Введите название продукта: ");

product.setNameProduct(sc.nextLine());

System.out.print("Введите состав продукта: ");

product.setCompositionProduct(sc.nextLine());

System.out.print("Введите форму для хранения продукта: ");

product.setStorageFormProduct(sc.nextLine());

System.out.print("Введите цену продукта($): ");

product.setPrice(sc.nextDouble());

milks.add(product);

System.out.print("\nПродукт добавлен в магазин.");

}

}

Класс Main:

package com.company;

public class Main {

public static Store store = new Store();

public static void main(String[] args){

store.display();

}

}

6 Заключение

Были изучены механизмы наследования, полиморфизма, инкапсуляции на практике. Были получены практические навыки описания иерархии классов и организации работы с объектами-членам классов-“групп”. Получены практические навыки использования абстрактных классов и переопределения унаследованных методов.

7 Список использованных источников

1. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: <http://developer.alexanderklimov.ru/android/java/java.php> (дата обращения 12.09.20)

2. API Specification for the Java 7 SE. [официальный сайт] URL: <http://docs.oracle.com/javase/7/docs/api/> (дата обращения 12.09.20)

3. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированны анализ и проек-тирование с примерами приложений. Третье издание. М.: "Вильямс", 2010, -720 с.

4. Хабибуллин И.Ш. Java 7: для программистов / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ–Петербург, 2014. – 768 с.

5. Васильев А. Н. Java. Объектно-ориентированное программирование: для магистров и бакалавров. Базовый курс по объектно-ориентированному программированию / А. Н. Васильев . – СПб.: Питер, 2012. – 395 с.