**Лабораторная работа №1**

**Тема:** Основы Java и консольное приложение

**Цель:** Целью данной лабораторной работы является разработка консольного приложения для управления задачами на языке Java с использованием объектно-ориентированных подходов и коллекций. Необходимо реализовать функционал создания, удаления и вывода задач через консоль, что позволит закрепить знания синтаксиса Java и работы с базовыми структурами данных.

**Ход работы**  
Работа началась с инициализации проекта на Maven, который обеспечивает управление зависимостями и облегчает сборку проекта. После этого был разработан основной функционал приложения для управления задачами.

В первую очередь был создан класс Task, содержащий поля для идентификатора задачи (id), названия (title) и описания (description). Эти данные позволяют структурировать информацию о каждой задаче. Для удобства управления задачами в приложении был разработан класс TaskManager, который использует коллекцию ArrayList для хранения объектов Task.

Класс TaskManager включает методы для:

* Добавления новой задачи в список.
* Удаления задачи по её идентификатору (индексу).
* Вывода списка всех задач в консоль.

Для взаимодействия пользователя с программой в методе main был реализован консольный интерфейс. Программа предлагает пользователю несколько команд:

* Создать новую задачу, где необходимо ввести название и описание.
* Вывести на экран список всех задач с указанием их идентификаторов.
* Удалить задачу по её идентификатору.

Команды обрабатываются с помощью условий и циклов, что позволяет программе реагировать на ввод пользователя и выполнять соответствующие действия.

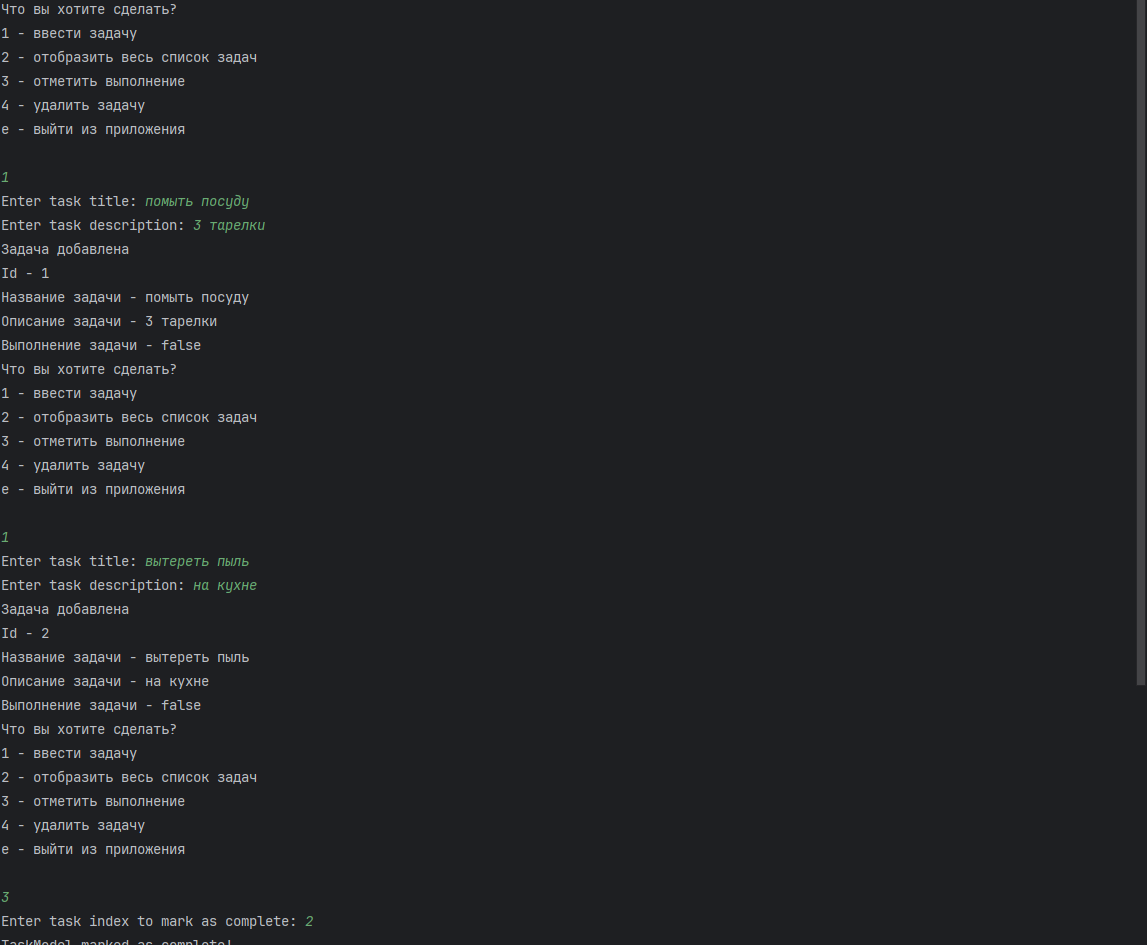


Рисунок 1 – Результат работы программы

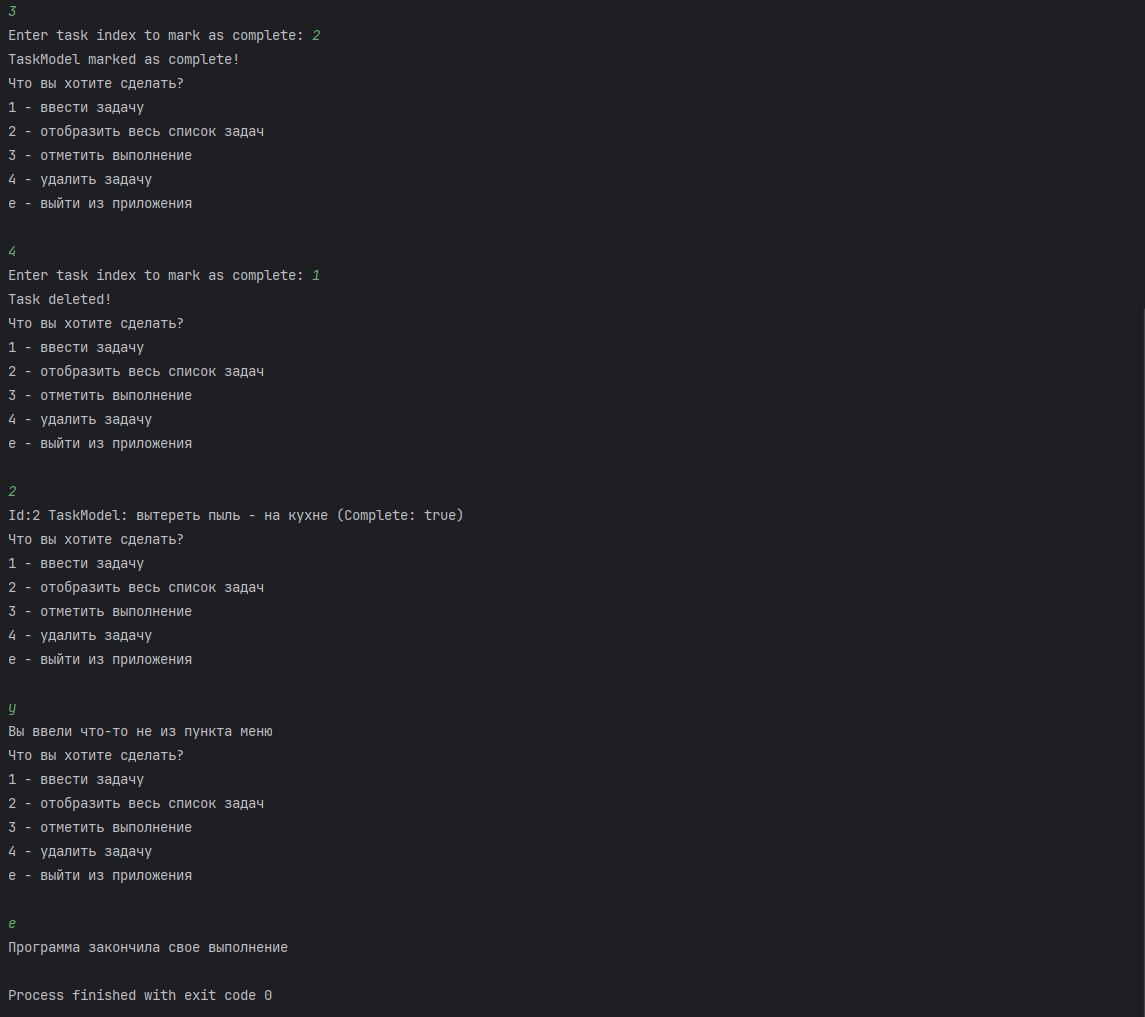


Рисунок 2 – Результат работы программы

**Заключение:**

Результатом лабораторной работы стало создание консольного приложения, которое успешно выполняет задачи по управлению списком дел. Пользователь может добавлять новые задачи с их названием и описанием, удалять задачи по идентификатору и просматривать список всех задач. Программа корректно реагирует на команды пользователя, поддерживает работу с динамическими коллекциями и использует основные элементы синтаксиса Java, такие как классы, методы, циклы и условия.

**Вывод:** в ходе выполнения лабораторной работы были достигнуты все поставленные цели. Я изучил основы синтаксиса Java, включая создание классов и методов, а также использование коллекций для хранения данных. Программа работает стабильно и выполняет все необходимые функции для управления задачами. Полученные знания и навыки будут полезны для дальнейшего изучения объектно-ориентированного программирования и создания более сложных приложений на Java.

**Приложение 1**

Main.java

package com.myapp;

import com.myapp.controller.TaskController;

import java.nio.charset.Charset;

import java.util.Scanner;

public class Main {

public static void menuSelector(){

Scanner in = new Scanner(System.in);

TaskController taskController = new TaskController();

int id = 1;

while (true){

System.out.println("""

Что вы хотите сделать?

1 - ввести задачу

2 - отобразить весь список задач

3 - отметить выполнение

е - выйти из приложения

""");

switch (in.nextLine()){

case "1":

// Ввод задачи

taskController.createTask(id);

id++;

break;

case "2":

taskController.listTasks();

break;

case "3":

taskController.completeTask();

break;

case "e":

// Закрываем Scanner

in.close();

return;

default:

System.out.println("Вы ввели что-то не из пункта меню");

break;

}

}

}

public static void main(String[] args) {

System.out.println("Текущая кодировка: " + Charset.defaultCharset().displayName());

menuSelector();

System.out.println("Программа закончила свое выполнение");

}

}

TaskController.java

package com.myapp.controller;

import com.myapp.model.Task;

import com.myapp.sevice.TaskService;

import java.util.Scanner;

public class TaskController {

private TaskService taskService;

private Scanner scanner;

public TaskController() {

this.taskService = new TaskService();

this.scanner = new Scanner(System.in);

}

// ИЗБЫТОЧНО НО МОЖЕТ ПРИГОДИТСЯ ПРИ ПОЯВЛЕНИИ ДРУГИХ ПОЛЕЙ

// Универсальный метод для ввода значений

public Object inputValue(String prompt, Class<?> type) {

System.out.print(prompt);

String input = scanner.nextLine(); // Ввод всех данных как строки

// Преобразуем строку в нужный тип данных

if (type == String.class) {

return input; // Если это строка, просто возвращаем введённую строку

} else if (type == int.class) {

return Integer.parseInt(input); // Преобразование строки в целое число

} else {

return null; // Для других типов можно добавить обработку

}

}

public void createTask(int id) {

String title = (String) inputValue("Enter task title: ", String.class); // Ввод строки

String description = (String) inputValue("Enter task description: ", String.class); // Ввод строки

// Создаём и возвращаем объект Task

taskService.addTask(new Task(id, title, description));

}

public void listTasks() {

for (Task task : taskService.getAllTasks()) {

System.out.println(task.getTaskInfo());

}

}

public void completeTask() {

System.out.print("Enter task index to mark as complete: ");

int index = scanner.nextInt();

taskService.markTaskAsComplete(index);

System.out.println("Task marked as complete!");

}

}

TaskModel.java

package com.myapp.model;

public class TaskModel {

private final int id;

private String title;

private String description;

private boolean completed;

// Конструктор с параметрами для задания начальных значений

public TaskModel(int id, String title, String description) {

this.id = id; // Присваиваем значение полю id при создании объекта

this.title = title; // Присваиваем значение полю title при создании объекта

this.description = description; // Присваиваем значение полю description

this.completed = false; // По умолчанию задача считается не выполненной

}

// Сеттеры для возможности изменения значений полей позже

public void setTitle(String title) {

this.title = title;

}

public void setDescription(String description) {

this.description = description;

}

public void setCompleted(boolean completed) {

this.completed = completed;

}

// Геттеры для доступа к значениям полей

public String getTitle() {

return title;

}

public String getDescription() {

return description;

}

public boolean getCompleted() {

return completed;

}

public int getId() {

return id;

}

public String getTaskInfo() {

return "Id:" + id + " TaskModel: " + title + " - " + description + " (Complete: " + completed + ")";

}

}

TaskService.java

package com.myapp.sevice;

import com.myapp.model.TaskModel;

import java.util.ArrayList;

import java.util.List;

public class TaskService {

private List<TaskModel> tasks;

public TaskService(){

this.tasks = new ArrayList<>();

}

public void addTask(TaskModel task) {

tasks.add(task);

System.out.println("Задача добавлена");

System.out.println("Id - "+task.getId());

System.out.println("Название задачи - " + task.getTitle());

System.out.println("Описание задачи - " + task.getDescription());

System.out.println("Выполнение задачи - " + task.getCompleted());

}

public List<TaskModel> getAllTasks() {

return tasks;

}

public void markTaskAsComplete(int id) {

if (id >= 0) {

for (TaskModel i : tasks){

if(i.getId() == id){

i.setCompleted(true);

break;

}

}

}

}

public void deleteTask(int id) {

if (id >= 0) {

for (TaskModel i : tasks){

if(i.getId() == id){

tasks.remove(i);

break;

}

}

}

}

}