Министерство образования Республики Беларусь
Учреждение образования
«Витебский государственный технологический университет»

Кафедра «Информационные системы и технологии»

КУРСОВОЙ ПРОЕКТ

по дисциплине «Объектно-ориентированное проектирование и программирование»

на тему «система ЦВЕТОЧНИЦА»

Выполнил:

студент факультета ФИТР _____ Руммо В.Г.

группы ИТ-15 *подпись*

Проверил:

руководитель

ст.преп.каф. ИСиТ _____ Деркаченко П.Г

подпись

_____

отметка о допуске к защите

«___» _____ 2025 г._____

подпись

Витебск, 2025

Учреждение образования
«Витебский государственный технологический университет»
Факультет информационных технологий и робототехники
Кафедра «Информационные системы и технологии»

УТВЕРЖДАЮ

Заведующий кафедрой ИСиТ

_____ Казаков В.Е.

«____» _____ 2025 г.

ЗАДАНИЕ
на курсовое проектирование
по дисциплине «Объектно-ориентированное проектирование и программирование»

Студенту Руммо Валерии Геннадьевне группы ИТ-15

1. Тема курсовой работы Система ЦВЕТОЧНИЦА

2. Сроки сдачи студентом завершенного проекта: _____

3. Исходные данные к курсовой работе Создать класс **Цветок**. Создать объекты-цветы разного наименования и стоимости. Собрать букет из цветов на определённую сумму (задаётся пользователем с клавиатуры). Входные данные: стоимость букета (целое), выходные данные: букет - список цветов, с суммарной стоимостью, равной заданной. Цветы в букете могут повторяться. Если на заданную сумму подобрать букет невозможно, сообщить об этом пользователю и попросить ввести другую сумму. Провести сортировку цветов в букете по их стоимости; по их названиям (тип сортировки вводится пользователем). Найти количество цветов заданного типа в букете (розы, гвоздики и пр. – вводится пользователем).

4. Содержание расчетно-пояснительной записки:

Введение

1. Постановка задачи
2. Описание метода решения задачи
3. Разработка схемы классов, блок-схем алгоритмов задачи

4. Разработка классов для решения поставленной задачи
5. Разработка тестового примера
6. Заключение
7. Список использованных источников

5. Характеристика графического материала и/или презентации: _____

_____

6. Руководитель курсового проектирования:

Деркаченко Павел Григорьевич

7. Календарный график курсового проектирования

№	Содержание этапа работы	Сроки выполнения	Подпись или замечания руководителя
1	Введение, Постановка задачи. Описание метода решения задачи	12.09.2025 – 22.09.2025	
2	Разработка схемы (схем) классов, блок-схем алгоритмов задачи	23.09.2025 – 06.10.25	
3	Разработка классов для решения поставленной задачи	07.10.25 – 07.11.2025	
4	Разработка тестового примера	08.11.2025 – 28.11.25	
5	Выводы. Оформление курсовой работы	29.11.2025 – 11.12.2025	
6	Предоставление на окончательную проверку курсовой работы руководителю	12.12.2025 – 24.12.2025	
7	Защита курсовой работы	30.12.25	

Руководитель Деркаченко П.Г

Задание принял к исполнению «12» сентября 2025 г. _____

подпись

Проинформирован об обязанности выполнять курсовую работу самостоятельно, без привлечения третьих лиц, не допускать плагиата, некорректных заимствований, фальсификации и подлога материалов

_____

подпись

СОДЕРЖАНИЕ

Введение

В современном мире автоматизация процессов играет ключевую роль в повышении эффективности различных сфер деятельности. Цветочный бизнес - не исключение. Подбор букетов на определенную сумму является типичной задачей, с которой сталкиваются как флористы, так и клиенты цветочных магазинов. Ручной подбор комбинаций цветов для заданного бюджета является трудоемким и не всегда оптимальным процессом. Разработка автоматизированной системы для решения этой задачи позволяет существенно ускорить процесс формирования букетов.

Задача подбора букета на заданную сумму относится к классу комбинаторных задач о рюкзаке (knapsack problem), где требуется найти все возможные комбинации предметов (цветов) с определенными стоимостями, сумма которых равна заданному бюджету.

Разработка объектно-ориентированной программной системы "ЦВЕТОЧНИЦА", позволяющей:

- Формировать все возможные комбинации букетов на заданную сумму
- Ограничивать количество одинаковых цветов в букете для повышения разнообразия
- Сортировать состав букетов по различным критериям
- Анализировать состав букетов (поиск цветов определенного типа)

Задачи проектирования:

1. Проанализировать предметную область и формализовать требования
2. Разработать архитектуру системы с использованием принципов ООП
3. Реализовать алгоритм поиска всех комбинаций букетов на заданную сумму

4. Спроектировать пользовательский интерфейс для взаимодействия с системой
5. Протестировать работоспособность системы на различных входных данных

Используемые технологии и инструменты:

- Язык программирования: Java 11+
- Среда разработки: IntelliJ IDEA
- Парадигма программирования: Объектно-ориентированное программирование
- Принципы проектирования: SOLID, DRY, KISS
- Структуры данных: List, Map, Set
- Алгоритмы: рекурсивный поиск с возвратом (backtracking)

1. Постановка задачи

Система "ЦВЕТОЧНИЦА" предназначена для автоматизации процесса подбора цветочных букетов на заданный бюджет. Программа решает следующие задачи:

- Автоматический поиск всех возможных комбинаций цветов, суммарная стоимость которых равна заданному бюджету
- Ограничение максимального количества одинаковых цветов в букете для повышения разнообразия
- Сортировка цветов в букете по стоимости или названию
- Интерактивное взаимодействие с пользователем через консольный интерфейс

Основные функции системы:

1. Управление каталогом цветов:
    - Хранение информации о доступных цветах (название, стоимость)
    - Отображение доступных цветов с ценами

2. Поиск букетов:
    - Ввод бюджета пользователем
    - Ввод ограничения на максимальное количество одинаковых цветов
    - Поиск всех возможных комбинаций с помощью рекурсивного алгоритма
    - Исключение дубликатов комбинаций
3. Отображение результатов:
    - Постраничный вывод найденных букетов

4. Обработка букетов:
    - Сортировка цветов в букете по стоимости или названию
    - Поиск и подсчет цветов по части названия
    - Вывод детальной информации о букете

5. Обработка ошибок:
    - Валидация вводимых пользователем данных
    - Обработка некорректного ввода
    - Сообщения об ошибках в понятном формате

Ограничения и допущения

- Входные данные: бюджет (целое положительное число), ограничение на одинаковые цветы (целое неотрицательное число)
- Ограничение по данным: в системе предустановлен набор из 12 видов цветов с ценами от 5 до 15 рублей
- Ограничение по времени: поиск букетов для бюджета до 100 рублей выполняется за разумное время (до 5 секунд)
- Пользовательский интерфейс: консольный, с пошаговым руководством
- Допущения: все цены целочисленные, цветы могут повторяться в букете в пределах установленного лимита

2. Описание метода решения задачи

Система построена по модульному принципу с четким разделением ответственности между компонентами. Используется трехслойная архитектура:

1. Слой предметной области (Domain Layer):
   - Классы `Flower` и `Bouquet` представляют основные сущности системы
   - Инкапсулируют бизнес-логику и правила предметной области

2. Слой бизнес-логики (Business Logic Layer):
   - Классы `FlowerManagement` и `BouquetManagement` содержат логику работы с цветами и букетами
   - Реализуют алгоритмы поиска и обработки данных

3. Слой представления (Presentation Layer):
   - Класс `ConsoleUI` обеспечивает взаимодействие с пользователем
   - Отвечает за ввод данных и вывод результатов

При разработке системы применены следующие принципы ООП:

1. Инкапсуляция:
   - Все поля классов объявлены как private
   - Доступ к данным осуществляется через публичные методы (геттеры)
   - Внутренняя реализация скрыта от внешних классов
2. Наследование:
   - Класс `Flower` реализует интерфейс `Comparable<Flower>` для поддержки сортировки

- Использование наследования стандартных классов коллекций Java

3. Полиморфизм:
   - Использование интерфейса `Comparable` для различных стратегий сортировки
4. Абстракция:
   - Классы представляют абстракции реальных сущностей (цветок, букет)

Алгоритмическое решение:

Для поиска всех возможных комбинаций букетов используется рекурсивный алгоритм с возвратом (backtracking)
Структуры данных:

- `List<Flower>`: для хранения доступных цветов и временных комбинаций
- `Map<Flower, Integer>`: для хранения букетов (цветок → количество)
- `Set<String>`: для исключения дубликатов комбинаций
- `LinkedHashMap`: для сохранения порядка элементов при сортировке

Обоснование выбора метода

Рекурсивный поиск с возвратом выбран по следующим причинам:

1. Полнота решения: гарантирует нахождение всех возможных комбинаций
2. Гибкость: позволяет легко добавлять ограничения (лимит на одинаковые цветы)
3. Простота реализации: алгоритм интуитивно понятен и легко отлаживается

3. Разработка схемы классов, блок-схем алгоритмов задачи

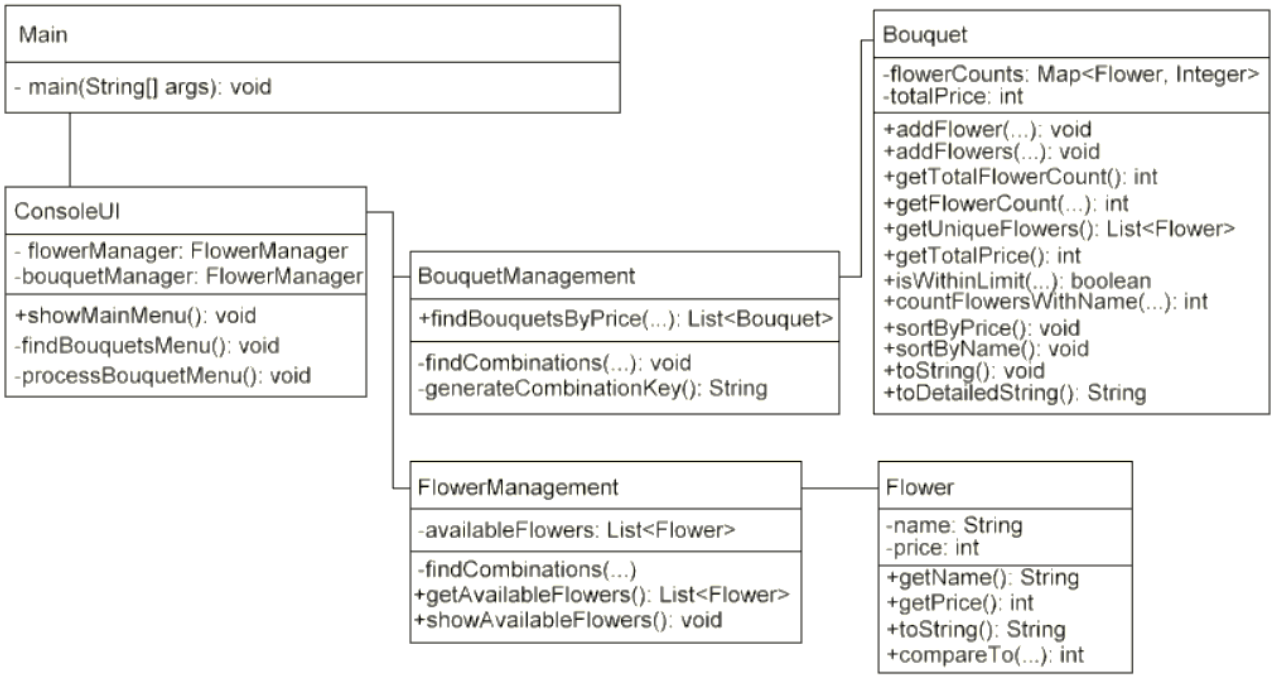

Рисунок 4.1 - Диаграмма классов UML

1. Композиция вместо наследования:
   - `Bouquet` содержит `Map<Flower, Integer>` (композиция)
   - `ConsoleUI` использует `FlowerManagement` и `BouquetManagement` (агрегация)
2. Использование интерфейсов:
   - `Flower` реализует `Comparable<Flower>` для поддержки сортировки
3. Разделение ответственности:
   - `Flower` - хранение данных о цветке
   - `Bouquet` - управление коллекцией цветов
   - `FlowerManagement` - управление каталогом цветов
   - `BouquetManagement` - алгоритмы поиска букетов
   - `ConsoleUI` - взаимодействие с пользователем

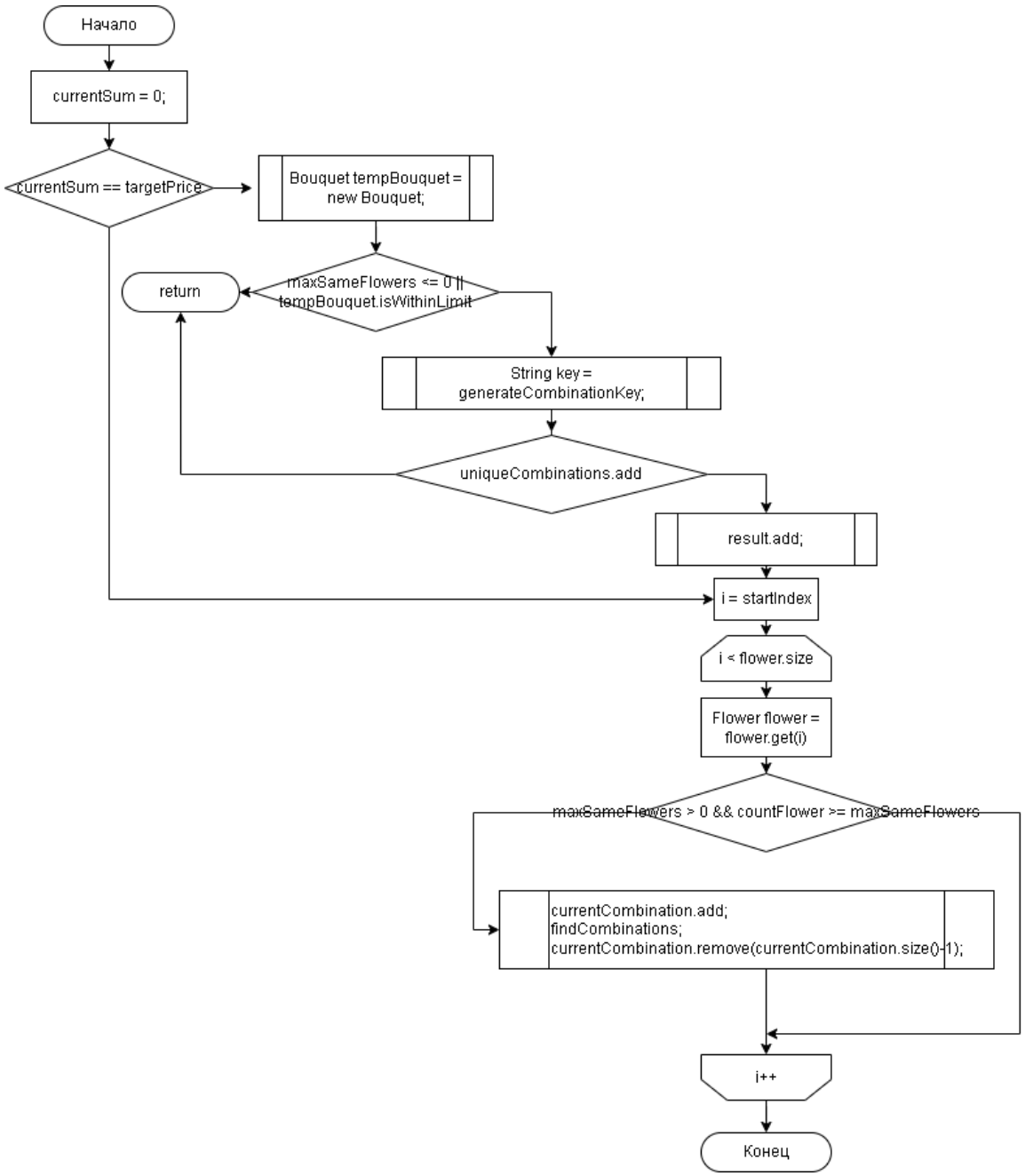

Рисунок 4.2 – блок-схема метода `findCombinations`

4. Разработка классов для решения поставленной задачи

GitHub: https://github.com/uwuNixx/KursProjectOOP

1. Пакет flower

Flower.java:

Поля:

- `private final String name` - название цветка.
- `private final int price` - цена цветка в рублях.

Методы:

- Конструктор `Flower(String name, int price)` - создает объект цветка с заданными параметрами.
- `public String getName()` - геттер для получения названия цветка.
- `public int getPrice()` - геттер для получения цены цветка.
- `public int compareTo(Flower other)` - реализация интерфейса `Comparable<Flower>` для сравнения цветов по цене (используется в сортировке).
- `public String toString()` - возвращает строковое представление цветка в формате "Название - цена руб.".
- `public boolean equals(Object obj)` *и* `public int hashCode()` - переопределены для корректной работы с коллекциями (HashMap, HashSet).

Ответственность:

- Хранение неизменяемых данных о цветке.
- Предоставление доступа к данным через геттеры.
- Поддержка сравнения и сортировки через `Comparable`.

FlowerManagement.java:

Поля:

- `private List<Flower> availableFlowers` - список всех доступных цветов. Использует ArrayList для упорядоченного хранения.

Методы:

Инициализация:

- `loadFlowersFromFile(String filename)` - загружает данные о цветах из текстового файла конфигурации. Конструктор по умолчанию `FlowerManagement()` автоматически вызывает загрузку данных.

Операции с данными:

- `public List<Flower> getAvailableFlowers()` - возвращает копию списка доступных цветов (защита от модификации извне).
- `public List<Flower> findFlowersByName(String namePart)` - возвращает цветы, содержащие в названии заданную подстроку (регистронезависимый поиск).

Представление:

- `public void showAvailableFlowers()` - выводит в консоль таблицу доступных цветов с форматированием.

Ответственность:

- Хранение и управление каталогом цветов.
- Предоставление доступа к данным с защитой от несанкционированной модификации.
- Поиск цветов по критериям.
-

2. Пакет bouquet

Bouquet.java:

Поля:

- `private Map<Flower, Integer> flowerCounts` - отображение "цветок → количество". Используется HashMap для эффективного

доступа. Ключ - объект `Flower`, значение - количество таких цветков в букете.

- `private int totalPrice` - общая стоимость букета. Рассчитывается динамически при добавлении цветов.

Методы:

Конструкторы:

- `Bouquet()` - создает пустой букет.
- `Bouquet(List<Flower> flowers)` - создает букет из списка цветов (удобно для тестирования).

Основные операции:

- `public void addFlower(Flower flower)` - добавляет один цветок в букет, увеличивая счетчик и общую стоимость.
- `public void addFlowers(List<Flower> flowers)` - добавляет несколько цветков.
- `public List<Flower> getFlowers()` - возвращает "развернутый" список всех цветов (для обратной совместимости).
- `public int getTotalPrice()` - геттер для общей стоимости.

Операции анализа:

- `public int getTotalFlowerCount()` - возвращает общее количество цветов в букете.
- `public int getFlowerCount(Flower flower)` - возвращает количество конкретного цветка в букете.
- `public List<Flower> getUniqueFlowers()` - возвращает список уникальных цветов в букете.
- `public int countFlowersWithName(String namePart)` - подсчитывает цветы, содержащие в названии заданную подстроку (нечеткий поиск).

Операции сортировки:

- `public void sortByPrice()` - сортирует уникальные цветы по цене (от дешевых к дорогим). Использует LinkedHashMap для сохранения порядка.
- `public void sortByName()` - сортирует уникальные цветы по названию (алфавитно).

Валидация:

- `public boolean isWithinLimit(int maxSameFlowers)` - проверяет, что ни один цветок не превышает заданный лимит количества.

Представление:

- `public String toString()` - возвращает форматированное строковое представление букета с группировкой одинаковых цветов.
- `public String toDetailedString()` - возвращает детальное представление (все цветы по отдельности, для отладки).

Ответственность:

- Управление композицией цветов с учетом их количества.
- Автоматический расчет общей стоимости.
- Сортировка и представление данных в удобном формате.

BouquetManagement.java:

Методы:

Основной алгоритм:

- `public List<Bouquet> findBouquetsByPrice(…)` - публичный метод для поиска всех букетов. Возвращает пустой список, если букеты не найдены.

Приватные вспомогательные методы:

- `private void findCombinations(…)` - рекурсивный алгоритм поиска с возвратом (backtracking). Основная логика:
    1. Рассчитывает текущую сумму.
    2. Проверяет превышение бюджета (ранний выход).

3. При достижении точной суммы создает букет и проверяет ограничения.
4. Рекурсивно добавляет цветки (разрешая повторения).
5. Использует backtracking для исследования всех комбинаций.

- `private String generateCombinationKey(List<Flower> flowers)` - генерирует уникальный ключ для комбинации цветов (используется для исключения дубликатов).

Ответственность:

- Реализация эффективного алгоритма поиска всех комбинаций.
- Обеспечение уникальности результатов.

3. Основной класс

Main.java:

Main содержит единственный статический метод main().

Архитектурные решения:

1. Принцип единственной ответственности: класс отвечает только за запуск приложения
2. Четкое разделение слоев: не смешивает логику запуска с бизнес-логикой
3. Простота и ясность: код минимален и самодокументируем

Взаимодействие: создает экземпляр ConsoleUI и делегирует ему управление программой, после чего управление полностью передается пользовательскому интерфейсу.

ConsoleUI.java:

Поля:

- `private FlowerManagement flowerManager` - ссылка на менеджер цветов.

- `private BouquetManagement bouquetManager` - ссылка на менеджер букетов.

Методы:

Управление программой:

- `public void showMainMenu()` - основной цикл программы, отображает главное меню и обрабатывает выбор пользователя.

Основные сценарии:

- `private void findBouquets(Scanner scanner)` - сценарий поиска букетов:
    1. Запрашивает у пользователя бюджет и ограничение.
    2. Вызывает `bouquetManager.findBouquetsByPrice()`.
    3. Реализует постраничный вывод результатов.
    4. Предлагает выбрать букет для детальной обработки.
- `private void processBouquet(Scanner scanner, Bouquet bouquet)` - сценарий обработки выбранного букета:
    1. Предлагает сортировку (по цене или названию).
    2. Выполняет поиск цветов по названию через `bouquet.countFlowersWithName()`.

Вспомогательные методы:

- `private void showBouquetPreview(Bouquet bouquet, int index)` - отображает краткую информацию о букете (количество цветов, видов, стоимость).
- Обработка пользовательского ввода с валидацией и сообщениями об ошибках.

Ответственность:

- Предоставление удобного интерфейса для взаимодействия с системой.
- Валидация и обработка пользовательского ввода.
- Организация постраничного вывода больших объемов данных.

5. Разработка тестового примера

Сценарий использования приложения "ЦВЕТОЧНИЦА"

Контекст использования

Пользователь: Мария, студентка, хочет купить букет для подруги на день рождения.

Бюджет: 50 рублей

Требования:

- Букет должен быть разнообразным
- Хотелось бы, чтобы в букете были розы
- Готова рассмотреть разные варианты композиций

Цель тестирования

Проверить работоспособность основных функций системы:

1. Загрузка данных о цветах из файла
2. Поиск всех возможных букетов на заданную сумму
3. Ограничение количества одинаковых цветов
4. Сортировка и анализ выбранного букета

Шаг 1: Запуск программы и просмотр каталога

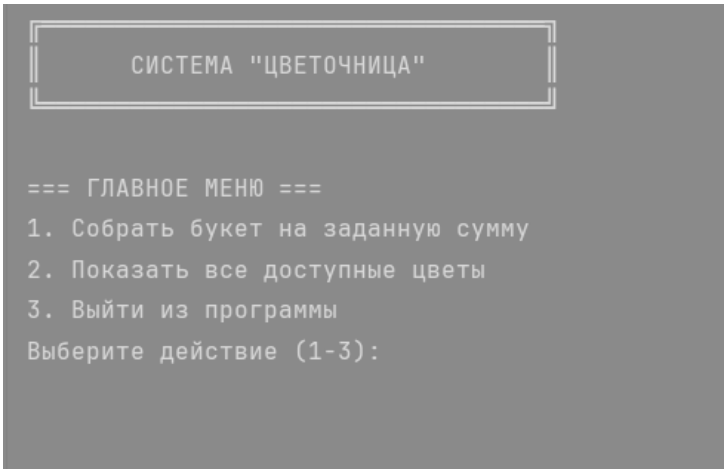

Рисунок 6.1 – Главное меню программы

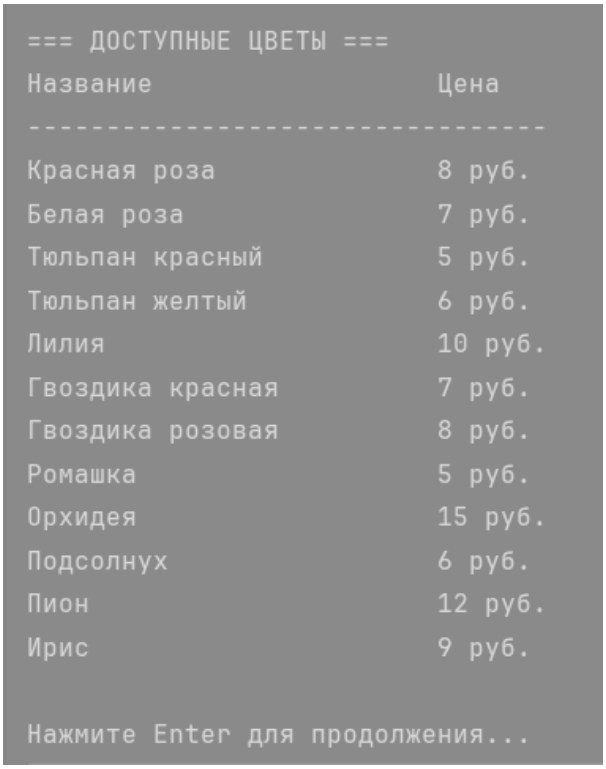

Рисунок 6.2 – Просмотр доступных цветов в виде списка

Шаг 2: Поиск букетов на 50 рублей с ограничениями

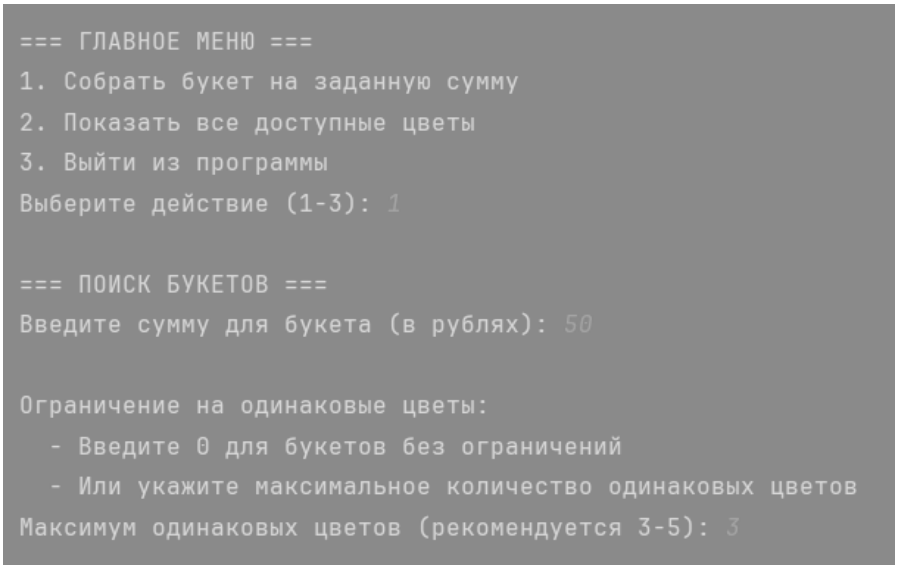

Рисунок 6.3 – Выбираем параметры букета (стоимость и ограничение на одинаковые цветы, для разнообразия букета)

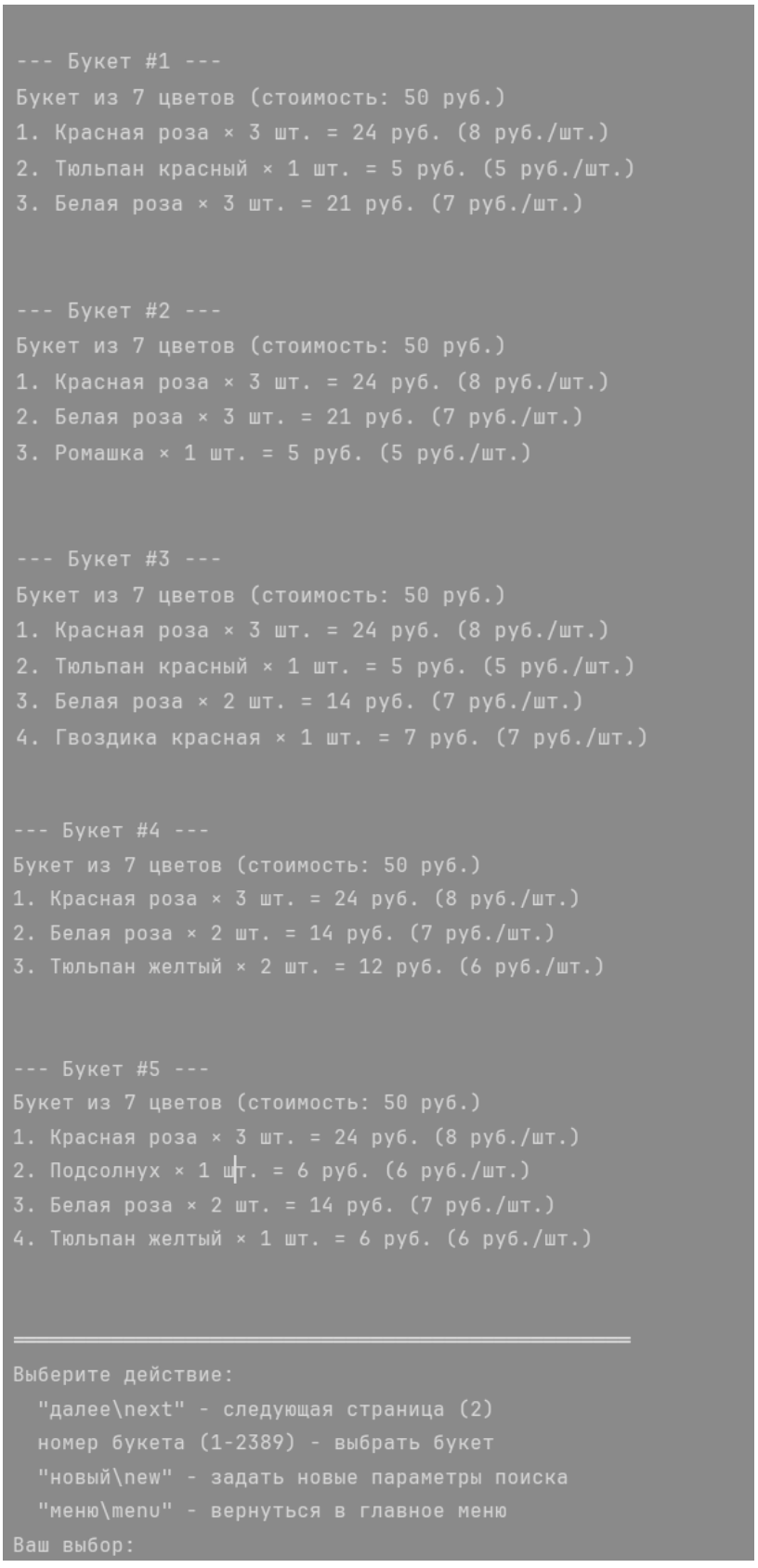

Рисунок 6.4 – Получаем список из букетов по 5 на одной странице, предполагаем, что ни один из данных нам не понравился, смотрим дальше

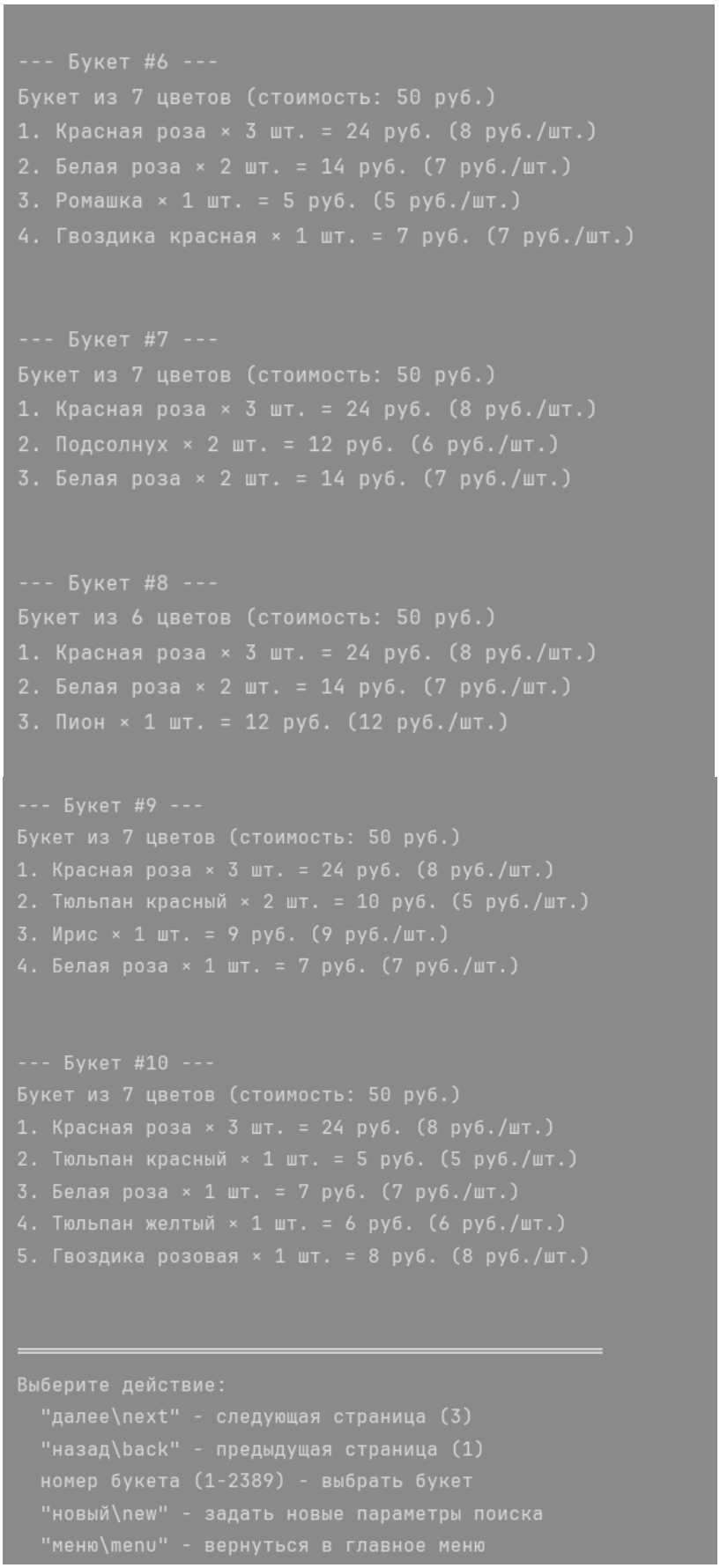

Рисунок 6.5 – Выбираем понравившийся букет, например #8, отсортируем его по цене и посмотрим, сколько всего в нем получилось роз

Шаг 3: Обработка выбранного букета

```
Выбран букет #8
Букет из 7 цветов (стоимость: 50 руб.)
1. Красная роза × 3 шт. = 24 руб. (8 руб./шт.)
2. Тюльпан красный × 2 шт. = 10 руб. (5 руб./шт.)
3. Ирис × 1 шт. = 9 руб. (9 руб./шт.)
4. Белая роза × 1 шт. = 7 руб. (7 руб./шт.)


=== ОБРАБОТКА БУКЕТА ===

Сортировка букета:
1. По стоимости цветов
2. По названиям цветов
Выберите тип сортировки (1-2): 1

Букет отсортирован по стоимости:
Букет из 6 цветов (стоимость: 50 руб.)
1. Белая роза × 2 шт. = 14 руб. (7 руб./шт.)
2. Красная роза × 3 шт. = 24 руб. (8 руб./шт.)
3. Пион × 1 шт. = 12 руб. (12 руб./шт.)


Найти цветы по названию (например: "роза") роза
Количество цветов с названием содержащим "роза": 5

Вернуться к просмотру букетов? (да/нет):
```

Рисунок 6.6 – Итог работы программы

Шаг 4: Тестирование обработки ошибок

```
=== ГЛАВНОЕ МЕНЮ ===
1. Собрать букет на заданную сумму
2. Показать все доступные цветы
3. Выйти из программы
Выберите действие (1-3): A
Ошибка: введите число от 1 до 3

=== ГЛАВНОЕ МЕНЮ ===
1. Собрать букет на заданную сумму
2. Показать все доступные цветы
3. Выйти из программы
Выберите действие (1-3): 1

=== ПОИСК БУКЕТОВ ===
Введите сумму для букета (в рублях): A
Ошибка: введите корректное число

=== ГЛАВНОЕ МЕНЮ ===
1. Собрать букет на заданную сумму
2. Показать все доступные цветы
3. Выйти из программы
Выберите действие (1-3): 1

=== ПОИСК БУКЕТОВ ===
Введите сумму для букета (в рублях): -4
Сумма должна быть положительной!
```

Рисунок 6.7 – Обработка ошибочного ввода

Анализ результатов

1. Корректность работы:
   - Программа находит все возможные комбинации для заданного бюджета
   - Ограничение на одинаковые цветы работает корректно
   - Сортировка выполняется согласно выбору пользователя
   - Поиск по названию учитывает частичные совпадения
2. Производительность:
   - Для бюджета 50 рублей поиск выполняется за 1-2 секунды
   - Ограничение на одинаковые цветы существенно сокращает время поиска
3. Удобство использования:
   - Постраничный вывод позволяет удобно просматривать большое количество вариантов
   - Группировка одинаковых цветов делает вывод компактным и информативным
   - Подсказки и сообщения об ошибках помогают пользователю
4. Ограничения:
   - При большом бюджете возрастает время поиска и результаты ограничиваются 100000 варинатами букетов (исключает переполнение памяти)
   - Количество вариантов может быть очень большим

В ходе выполнения курсового проекта была разработана программная система "ЦВЕТОЧНИЦА" для автоматизированного подбора цветочных букетов на заданный бюджет.

6. Заключение

Основные результаты

1. Анализ и формализация задачи:
   - Изучена предметная область и сформулированы требования к системе
   - Определены основные сущности (цветок, букет) и операции над ними
   - Проанализированы возможные алгоритмические подходы к решению задачи
2. Проектирование архитектуры:
   - Разработана модульная архитектура с четким разделением ответственности
   - Применены принципы объектно-ориентированного программирования
   - Спроектирована диаграмма классов, отражающая структуру системы
3. Реализация системы:
   - Разработаны классы для представления основных сущностей
   - Реализован рекурсивный алгоритм поиска всех комбинаций букетов
   - Создан консольный пользовательский интерфейс с постраничным выводом
   - Реализованы функции сортировки и анализа букетов
4. Тестирование и отладка:
   - Проведено тестирование системы на различных входных данных
   - Убедились в корректности работы алгоритмов
   - Проверили обработку некорректного ввода пользователя

Соответствие принципам ООП

1. Инкапсуляция: Все поля классов приватные, доступ через методы

2. Наследование: Использование интерфейса Comparable для поддержки сортировки
3. Полиморфизм: Параметрический полиморфизм через дженерики коллекций
4. Абстракция: Классы представляют абстракции реальных сущностей

Соответствие принципам SOLID

1. Single Responsibility: Каждый класс имеет одну ответственность
2. Open/Closed: Система открыта для расширения (добавление новых функций)
3. Liskov Substitution: Корректное использование наследования
4. Interface Segregation: Использование специализированных интерфейсов
5. Dependency Inversion: Зависимости от абстракций, а не конкретных реализаций

Практическая значимость

Разработанная система может быть использована:

- В цветочных магазинах для быстрого подбора букетов по бюджету клиента
- Для обучения основам комбинаторных алгоритмов и ООП
- Как основа для более сложных систем управления запасами цветов

Направления дальнейшего развития

1. Оптимизация производительности:
    - Внедрение алгоритма динамического программирования для больших бюджетов
2. Расширение функциональности:
    - Добавление базы данных для хранения каталога цветов
    - Добавление функции сохранения букетов в файл
3. Улучшение пользовательского опыта:
    - Добавление фильтров по типу цветов, цвету, сезонности
    - Реализация рекомендательной системы на основе предпочтений пользователя

7. Список использованных источников

1. Курсовое проектирование : методические указания методические указания по выполнению курсовых проектов по дисциплине «Объектно-ориентированное программирование» / сост. П. Г. Деркаченко. – Витебск : УО «ВГТУ», 2025. – 15 с.
2. Блок-схемы алгоритмов. ГОСТ. Примеры [Электронный ресурс] – Режим доступа: https://pro-prof.com/archives/1462 - Дата доступа: 23.12.2025
3. UML для самых маленьких: диаграмма классов [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/511798/ - Дата доступа: 23.12.2025
4. Фундаментальное руководство по пакетам в Java [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/755654/ - Дата доступа: 23.12.2025
5. Курс лекций «Объектно-ориентированное проектирование и программирование» / П.Г.Деркаченко