Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение вычислительной техники

и автоматизированных систем»

**Отчёт**

по лабораторной работе № 08

по дисциплине ***«Объектно-ориентированные технологии программирования и стандарты проектирования»***

или

по дисциплине ***«Объектно-ориентированное программирование»***

тема: «Базовый синтаксис языка Java. Циклические конструкции. Итерационные алгоритмы»

Название бригады: «…»

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студенты группы 1070 |
| Преподаватель: |  | Иванченко Виктор Викторович |

2018-2019 учебный год

**ЛАБОРАТОРНАЯ РАБОТА № 08**

**БАЗОВЫЙ СИНТАКСИС ЯЗЫКА JAVA. ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ.**

**ИТЕРАЦИОННЫЕ АЛГОРИТМЫ**

Цель работы:

Изучить синтаксис циклических конструкций языка Java для программирования итерационных алгоритмов и закрепить их на примере разработки простейших интерактивных консольных Java-приложений.

**Задание А**

1. Написать программу, которая подсчитывает количество цифр заданного натурального числа.

**Задание B**

1. Разработайте программу, которая проверяет, что все цифры, которые входят в заданное натурального число, имеют одинаковую чётность, т.к. либо все чётные, либо все нечётные.

Задание C

1. Разработайте программу, которая проверяет, что заданное натуральное число читается одинаково слева направо и справа налево (т.е. является палиндромом). К примеру, число 1235321 – удовлетворяет условию, а число 12345321 – нет.

Задание D

1. Найти количество различных цифр у заданного натурального числа. К примеру, число 12345436 имеет шесть различных цифр, а число 121212 – только два.

Задание E

1. Найти конкретное число Фибоначчи по его номеру.

Требования к выполнению

1. Каждому участнику из команды нужно просмотреть все задания и выполнить по одному понравившемуся заданию из каждого раздела. Задания из соответствующих разделов внутри команды не должны повторяться.
2. Для некоторых вычислительных алгоритмов (решается самостоятельно) необходимо спроектировать блок-схему решения, которую необходимо поместить в отчёт или конспект.
3. Все алгоритмы должны быть решены с использованием итеративного подхода (однако, если есть желание, то можно также привести параллельно решение задания с использованием рекурсивного подхода).
4. На базе спроектированных алгоритмов разработать простейшее интерактивное консольное приложение с использование архитектурного шаблона проектирования ***Model-View-Controller****,* ***MVC***.
5. Создаваемые классы необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные стартовые пакеты: ***by.bntu.fitr.povt.nameofteam.javalabs.lab08****.*
6. При выполнения задания необходимо по максимуму пытаться разрабатывать универсальный, масштабируемый и легко поддерживаемый и читаемый код.
7. Также рекомендуется придерживаться ***Single Responsibility Principle****,* ***SRP*** (принципа единственной ответственности): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только одна причина изменить в дальнейшем соответствующий блок кода.
8. Если логически не подразумевается или в задании иного не указано, то входными и выходными данными являются вещественные числа (числа с плавающей запятой).
9. Все задания необходимо решать используя только базовые операции (простые операторы), определённые над примитивными типами данных в языке программирования Java, и условные и циклические конструкции (т.е. не нужно использовать массивы или любые другие контейнеры данных, операции над строковыми типами данных и т.д.).
10. В соответствующих компонентах бизнес-логики необходимо предусмотреть «защиту от дурака».
11. Для генерирования случайных чисел воспользуйтесь методами объекта класса ***java.util.Random***, а для реализации ввода данных с консоли (терминала) – соответствующими методами объекта класса ***java.util.Scanner***.
12. Программа должна обязательно быть снабжена комментариями, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчиков, название бригады (если есть), номер группы и дату разработки. Исходный текст классов и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.
13. Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем.
14. Интерфейс программы и комментарии должны быть на английском языке.
15. При проверки работоспособности приложения необходимо проверить все тестовые случаи.
16. При выполнении задания не рекомендуется использовать интегрированные средства разработки (*Integrated Development Environment, IDE*). Лучше задействовать любой текстовый редактор и основные компоненты Java (компилятор – ***javac***, утилиту для запуска *JVM* – ***java***).
17. При разработке программ придерживайтесь соглашений по написанию кода на *Java* (***Java Code-Convention****)* !!!

Результаты выполнения основного задания

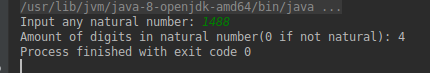


Рисунок 1 – Результат компиляции и выполнения

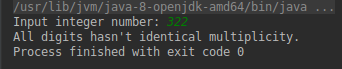


Рисунок 2 – Результат компиляции и выполнения

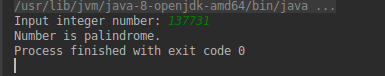


Рисунок 3 – Результат компиляции и выполнения

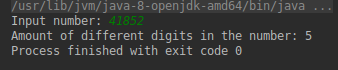


Рисунок 4 – Результат компиляции и выполнения

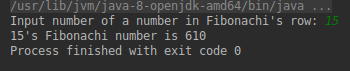


Рисунок 5 – Результат компиляции и выполнения

Что мы узнали нового в процессе выполнения лабораторной работы (выводы)

1) В JDK 8 в Java добавили элементы функционального языка, которые могут заменить в коде циклы, если требуется найти определенный элемент в базе данных.

2) Цикл со счетчиком с тремя пустыми блоками будет выполнять бесконечно

3) Блок итерации выполняется каждый раз при срабатывании массива

4) Блок условия в циклических конструкциях может содержать только выражение(переменную) типа boolean

5) В блоках инициализации и инкремента циклических конструкций со счетчиком возможно использование нескольких переменных одновременно

6) Нововведения в JDK 11:

* boolean isBlank(): возвращает true, если строка пуста или содержит только пробелы, иначе false.
* String repeat(int): возвращает строку, значение которой представляет собой конкатенацию этой строки, повторяющееся количество раз.
* String strip(): Возвращает строку, значение которой является этой строкой, при этом удаляются все пробелы в начале и в конце строки.
* Больше не нужно компилировать однофайловое приложение. Это сделает JDK

*ПРИЛОЖЕНИЕ А*

Листинг исходного кода класса(ов)