ФЕДЕРАЛЬНОЕ Государственное АВТОНОМНОЕ образовательное УЧРЕЖДЕНИЕ ВЫСШЕГО образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИРОВЫХ ТЕХНОЛОГИЙ

**Кафедра прикладной информатики и информационных технологий**

**Лабораторная работа 1**

**по дисциплине «Разработка программных приложений на языках высокого уровня»**

**студента очной формы обучения**

**направления подготовки 09.03.03. «Прикладная информатика»**

**3 курса группы 12002005**

**Морозова Максима Андреевича**

Научный руководитель:

ст. пр. Лифиренко Максим Вячеславович

|  |  |
| --- | --- |
| ***Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г.  \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись (расшифровка подписи) |  |

**БЕЛГОРОД, 2023**

Порядок выполнения работы - изучить теоретический материал; - напишите программы на языке Java:

1. Программа, в которой перебираются числа от 1 до 500 и выводятся на экран. Если число делится на 5, то вместо него выводится слово fizz, если на 7, то buzz. Если число делится на 5 и на 7, то выводить слово fizzbuzz. Примечание\*: остаток от деления в Java обозначается через символ %.

2. Программа, в которой все переданные во входную строку аргументы выводятся на экран в обратной порядке. Например, если было передано 2 аргумента – make install, то на экран должно вывестись llatsni ekam. Примечание\*: для разбора слова по буквам необходимо использовать функцию charAt(). Например, str.charAt(i) вернет символ с позиции i в слове, записанном в строковую переменную str. Команда str.length() возвращает длину слова str.

3. Создайте программу, вычисляющую числа Фибоначчи. Числа Фибоначчи – последовательность чисел, в котором каждое следующее число равно сумме двух предыдущих. Начало этой последовательности – числа 1, 1, 2, 3, 5, 8, 13…

4. Создайте программу, вычисляющую факториал целого числа.

Индивидуальное задание №8. Ввести с консоли 3 целых числа. На консоль вывести: Простые числа.

Результат работы программы 1:

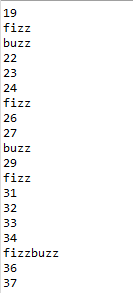


Рисунок 1 – результат работы программы

Результат работы программы 2:



Рисунок 2 – результат работы программы

Результат работы программы 3:

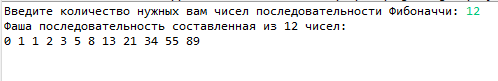


Рисунок 3 – результат работы программы

Результат работы программы 4:



Рисунок 4 – результат работы программы

Результат работы программы индивидуального задания №8:

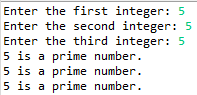


Рисунок 8 – результат работы программы

Ответы на контрольные вопросы:

1. В чем разница JDK и JRE?

1. JRE - это среда выполнения Java. Это пакет всего необходимого для

запуска скомпилированной Java-программы, включая виртуальную машину Java (JVM).JDK - полнофункциональный SDK для Java. В нем есть все, что есть в JRE, а также компилятор (javac) и инструменты (такие как javadoc и jdb). Он способен создавать и компилировать программы.

2. Назовите типы данных Java и классы-оболочки. Чем они отличаются?

2. Классы-оболочки (wrapper-классы) из пакета java.lang:

Boolean,Character, Integer, Byte, Short, Long, Float, Double. Объекты этих классов могут хранить те же значения, что и соответствующие им базовые типы. Объект любого из этих классов представляет собой полноценный экземпляр в динамической памяти, в котором хранится его неизменяемое значение. Значения базовых типов хранятся в стеке и не являются объектами.

Классы, соответствующие числовым базовым типам, находятся в библиотеке java.lang, являются наследниками абстрактного класса Number и реализуют интерфейс Comparable

Тип данныхКласс-обёртка

byteByte

shortShort

intInteger

longLong

charCharacter

floatFloat

doubleDouble

booleanBoolean

3. Для чего применяют документирование кода с помощью дескрипторов?

Перечислите основные дескрипторы.

3. Документирование кода необходимо для того, чтобы другие

программисты, работающие над кодом могли понять что и где было создано, как работает и куда передается.

@author – задает сведения об авторе;

@version – задает номер версии класса;

@exception – задает имя класса исключения;

@param – описывает параметры, передаваемые методу;

@return – описывает тип, возвращаемый методом;

@deprecated – указывает, что метод устаревший и у него есть более

совершенный аналог;

@since – определяет версию, с которой метод (член класса, класс)

присутствует;

@throws – описывает исключение, генерируемое методом;

@see – что следует посмотреть дополнительно.

4. Перечислите операторы управления, используемые в Java.

4. Java поддерживает:

Оператор if-then и if-then-else

Оператор switch

Оператор while

Оператор do-while

Оператор for

Оператор break

Оператор continue

Оператор return

**Листинг программы:**  
/\*\*

\* @author MaxM

\* @version 11010000.10011111.11010000.10110000.11010000.10111100.11010000.10110000.11010000.10111101.11010000.10111000.11010000.10111101

\*/public class FizzBuzz {

/\*

\*Главный метод программы, который выводит числа от 1 до 500 и, в зависимости от того,

\*делится ли число на 5, на 7 или на оба, выводит "fizz", "buzz", "fizzbuzz" или само число.

\*@param args массив строковых аргументов командной строки.

\*/

public static void main(String[] args) {

// Цикл от 1 до 500

for (int i = 1; i <= 500; i++) {

// Если число делится на 5 и 7 без остатка, выводим "fizzbuzz"

if (i % 5 == 0 && i % 7 == 0) {

System.out.println("fizzbuzz");

}

// Если число делится на 5 без остатка, выводим "fizz"

else if (i % 5 == 0) {

System.out.println("fizz");

}

// Если число делится на 7 без остатка, выводим "buzz"

else if (i % 7 == 0) {

System.out.println("buzz");

}

// В остальных случаях выводим само число

else {

System.out.println(i);

}

}

}

/\*\*

• Класс для вывода слов в обратном порядке. \*/ public class ReverseOrder {

/\*\*

• Метод для ввода текста с консоли, разделения его на слова и вывода этих слов в обратном порядке.

•

• @param args аргументы командной строки.

• @return void \*/

public static void main(String[] args) { // Создаем экземпляр класса Scanner для получения ввода пользователя

Scanner scanner = new Scanner(System.in); // Выводим сообщение, запрашивая у пользователя ввод текста

System.out.print("Введите текст: "); // Получаем ввод пользователя и сохраняем его в переменной

input String input = scanner.nextLine(); // Разбиваем ввод пользователя на отдельные слова и сохраняем их в массиве

words String[] words = input.split(" "); // Проходим по массиву слов в обратном порядке

for (int i = words.length - 1; i >= 0; i--) { // Получаем текущее слово

String word = words[i]; // Проходим по буквам текущего слова в обратном порядке

for (int j = word.length() - 1; j >= 0; j--) { // Выводим текущую букву

System.out.print(word.charAt(j)); } // Выводим пробел после каждого слова System.out.print(" "); }

}

}

import java.util.Scanner;

/\*\*

\* Класс Fibonacci генерирует последовательность чисел Фибоначчи на основе ввода пользователя.

\*/

public class Fibonacci {

/\*\*

\* Генерирует последовательность чисел Фибоначчи на основе ввода пользователя.

\*

\* @param numTerms количество чисел в последовательности Фибоначчи для генерации

\* @return sequence последовательность чисел Фибоначчи

\*/

public static int[] generateSequence(int numTerms) {

int[] sequence = new int[numTerms];

int firstTerm = 0, secondTerm = 1;

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

sequence[i] = firstTerm;

int nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

secondTerm = nextTerm;

}

return sequence;

}

/\*\*

\* Метод main запрашивает ввод пользователя, генерирует последовательность чисел Фибоначчи на

\* основе этого ввода и выводит последовательность в консоль

\* @param args аргументы командной строки

\* @return – результат выполнения работы;

\*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Введите количество чисел последовательности Фибоначчи для генерации: ");

int numTerms = input.nextInt();

int[] sequence = generateSequence(numTerms);

System.out.print("Ваша последовательность чисел Фибоначчи: ");

for (int term : sequence) {

System.out.print(term + " ");

}

}

}

/\*\*

Класс Factorial вычисляет факториал числа.

\*/

import java.util.Scanner;

public class Factorial {

/\*\*

\* Метод main является точкой входа в программу.

\* @param args параметры командной строки

\*/

public static void main(String[] args) {

// Создаем объект scanner для чтения ввода с клавиатуры

Scanner scanner = new Scanner(System.in);

// Просим пользователя ввести целое число

System.out.print("Введите целое число: ");

// Считываем введенное пользователем число

int n = scanner.nextInt();

// Вычисляем факториал числа n с помощью метода calculateFactorial

int factorial = calculateFactorial(n);

// Выводим результат на экран

System.out.println(n + "! = " + factorial);

}

/\*\*

\* Метод calculateFactorial вычисляет факториал числа n.

\* @param n целое число, для которого нужно вычислить факториал

\* @return факториал числа n

\*/

public static int calculateFactorial(int n) {

if (n == 0) { // если n равно 0, то возвращаем 1 (факториал 0 равен 1)

return 1;

} else { // в противном случае рекурсивно вызываем метод calculateFactorial для числа n-1 и умножаем результат на n

return n \* calculateFactorial(n - 1);

}

}

**import** java.util.Scanner;

/\*\*

Класс PrimeNumbers позволяет определить, являются ли введенные целые числа простыми числами.

\*/

import java.util.Scanner;

public class PrimeNumbers {

/\*\*

\* Метод main запускает программу и принимает на вход три целых числа.

\* Выводит на экран сообщение о том, является ли каждое число простым.

\* @param args массив строковых аргументов командной строки

\*/

public static void main(String[] args) {

//создаем объект класса Scanner

Scanner sc = new Scanner(System.in);

//выводим сообщение на экран и считываем первое число

System.out.print("Введите первое целое число: ");

int num1 = sc.nextInt();

//выводим сообщение на экран и считываем второе число

System.out.print("Введите второе целое число: ");

int num2 = sc.nextInt();

//выводим сообщение на экран и считываем третье число

System.out.print("Введите третье целое число: ");

int num3 = sc.nextInt();

// проверяем, является ли каждое введенное число простым

if (isPrime(num1)) {

System.out.println(num1 + " является простым числом.");

}

if (isPrime(num2)) {

System.out.println(num2 + " является простым числом.");

}

if (isPrime(num3)) {

System.out.println(num3 + " является простым числом.");

}

}

/\*\*

\* Метод isPrime проверяет, является ли введенное число простым.

\* @param num число, которое нужно проверить на простоту

\* @return возвращает true, если число является простым, и false, если нет

\*/

public static boolean isPrime(int num) {

if (num <= 1) {

return false;

}

// проверяем, есть ли у числа делители от 2 до корня из числа

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0) {

return false;

}

}

return true;

}