

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №1 по дисциплине «Автоматизация тестирования»

Обучающийся группы 6232-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Александров А.А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанков А.А.

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146881081)

[2 Результаты работы 4](#_Toc146881082)

[2.1 Результаты задания 1 4](#_Toc146881083)

[2.2 Результаты задания 2 5](#_Toc146881084)

[2.3 Результаты задания 3 6](#_Toc146881085)

[3 Вывод 8](#_Toc146881086)

[ПРИЛОЖЕНИЕ А Листинг программы 9](#_Toc146881087)

1. Постановка задачи

Задание 1

Реализовать простое консольное приложение калькулятор, позволяющее производить следующие арифметические операции в 4-х системах счисления: сложение, вычитание, умножение, деление.

Задание 2

Реализовать несколько параметризованных тестов для проверки каждой

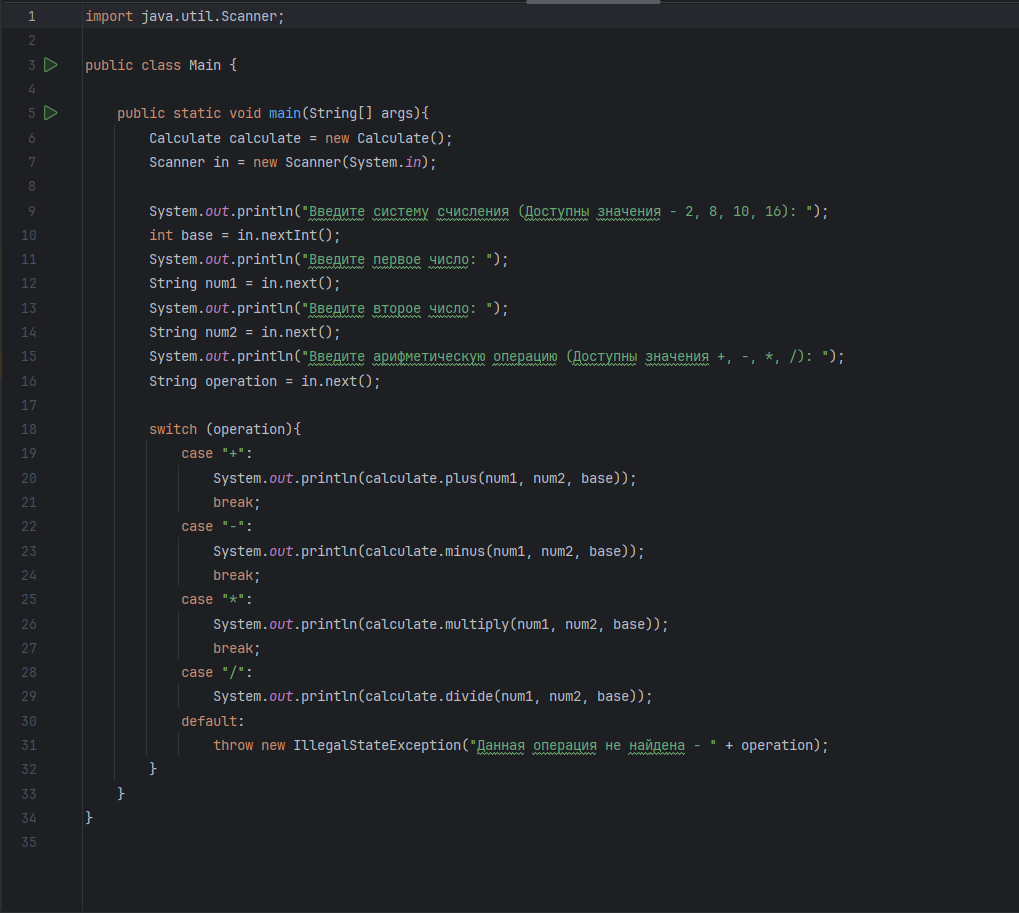
арифметической операции по каждой системе счисления. Должно быть, как минимум, по одному тесту, которые принимают в качестве параметров csv и источник данных. Отдельно (assertThrows) при попытке деления на 0.

Задание 3

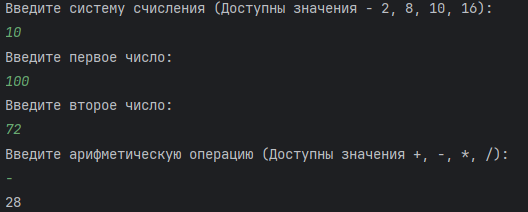
Подготовить yaml или csv файл с данными для тестирования. Реализовать динамические тесты на основе данных из подготовленного файла для одной из арифметической операции в каждой системе счисления. У каждого теста должно быть читаемое имя.

1. Результаты работы
   1. Результаты задания 1

Реализованная программа представляет собой простое консольное приложение, написанное на языке Java, выполняющее заданные арифметические операции для чисел в заданной системе исчисления.

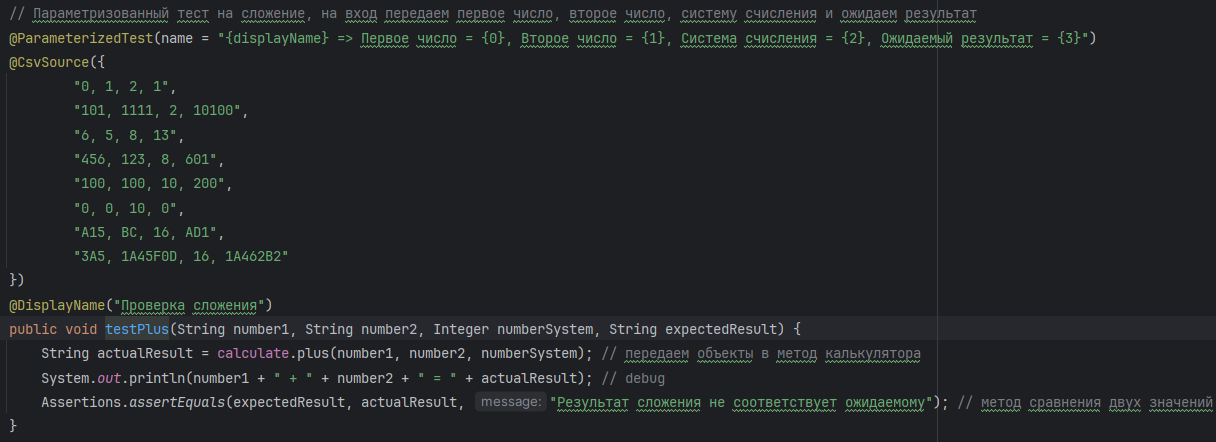


Вывод на консоль:



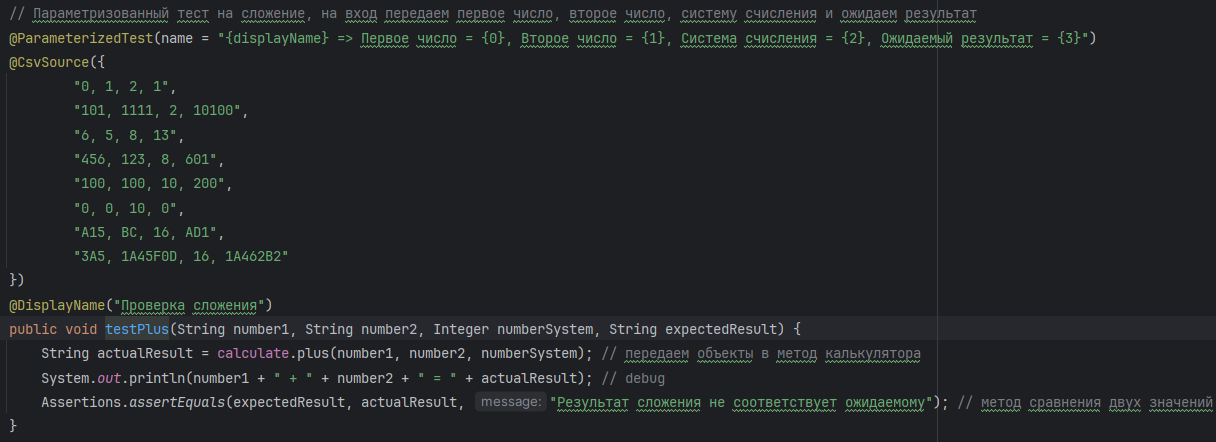
* 1. Результаты задания 2

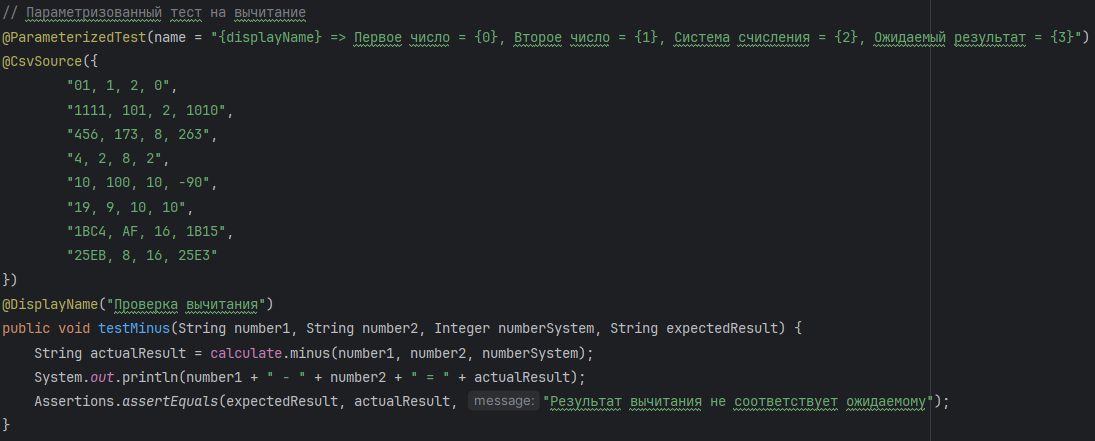
Параметризированные тесты. Значения параметров задаются конструктором атрибута, встроенного JUnit5.

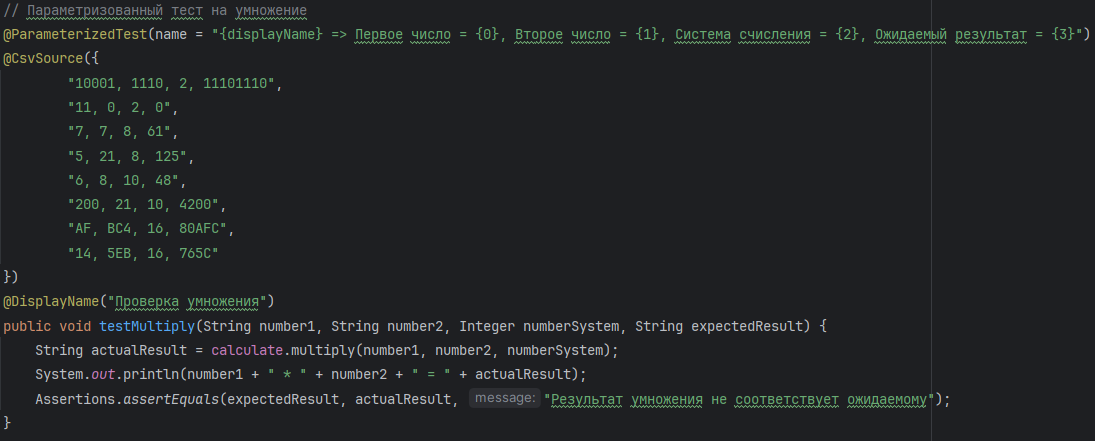


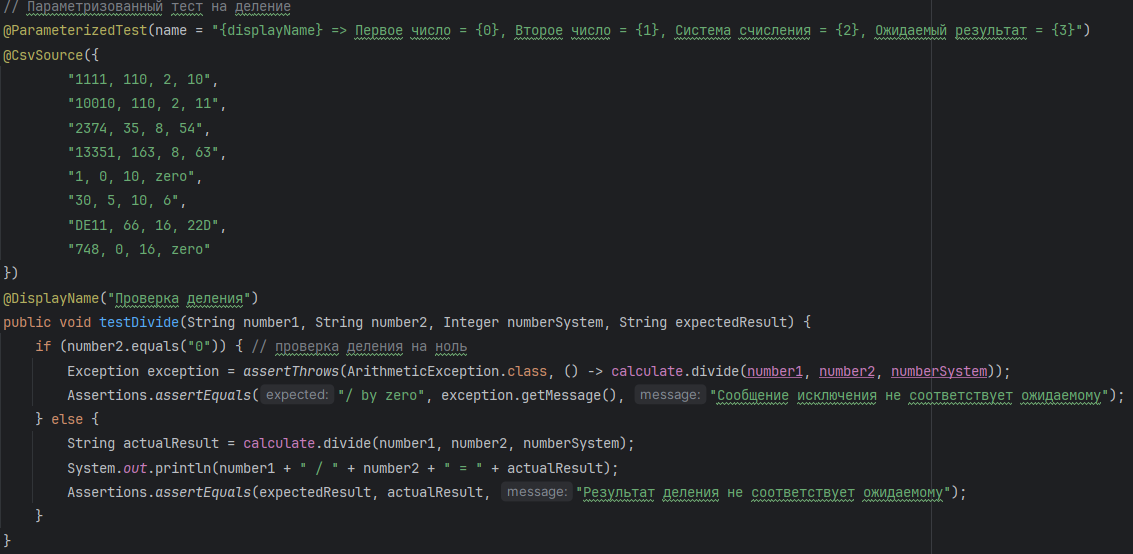
Параметризированные тесты. Источник данных – CSV data source, встроенный в JUnit5 и возвращающий перечисление списка параметров.

Источник данных для параметризированных тестов:

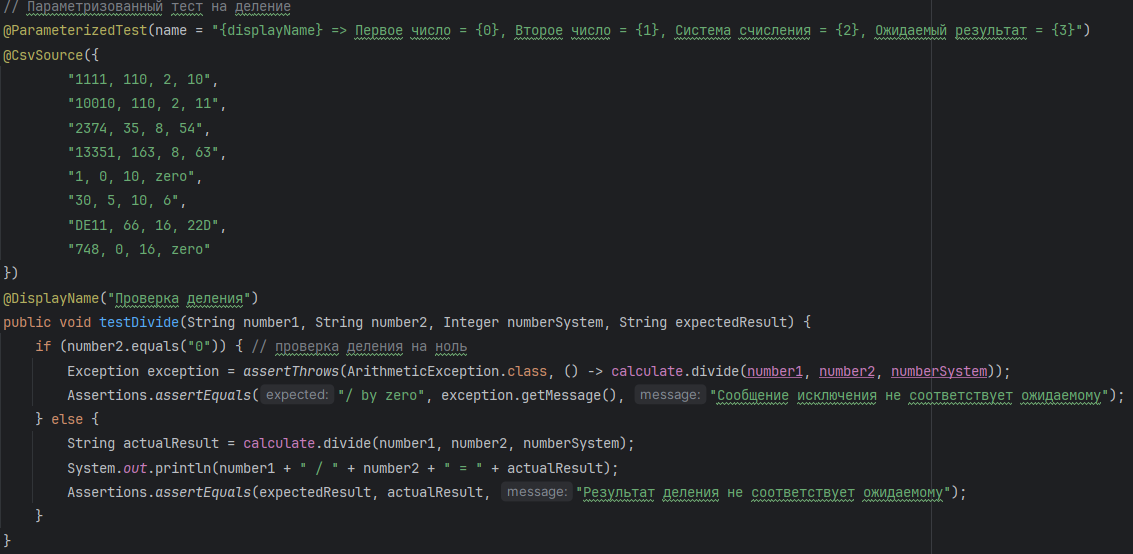






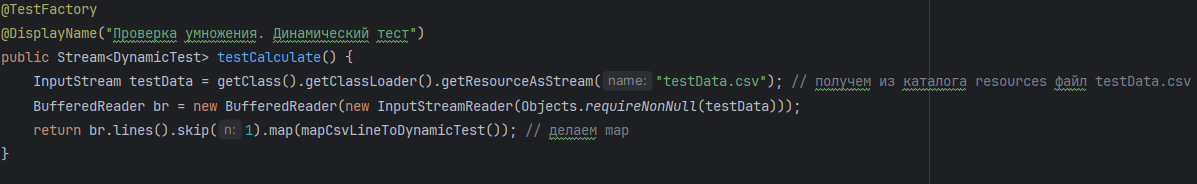


Источник данных для проверки на 0:

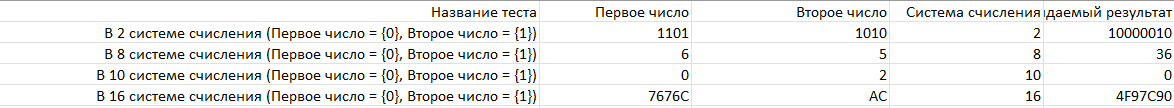


* 1. Результаты задания 3

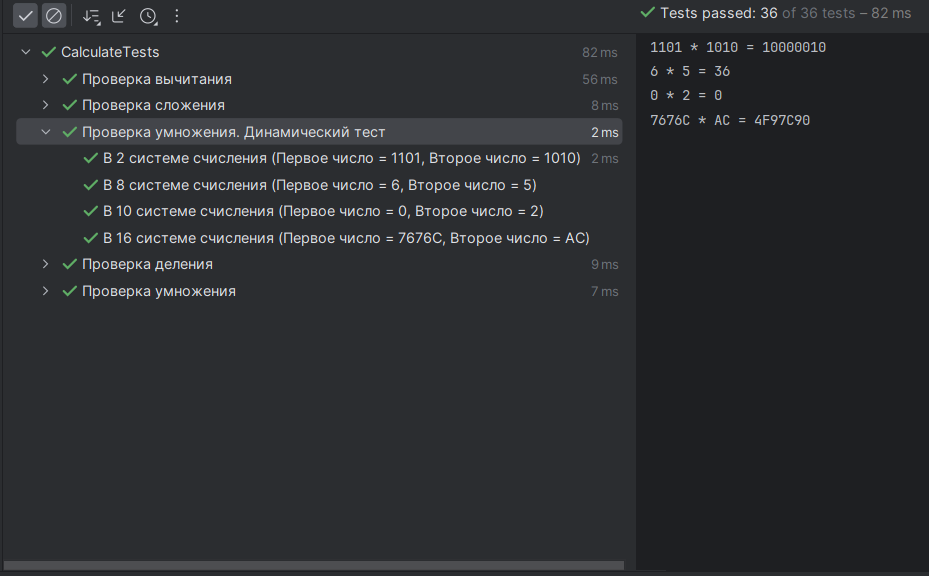
Для выполнения задания был реализован CSV data source. В качестве параметров допускается использовать как файл, так и строку с заданным разделителем. В данном случае использовался файл.



Структура CSV файла.



Результаты тестов:



1. Вывод

В результате выполнения лабораторной работы было реализовано простое консольное приложение калькулятор, позволяющее производить следующие арифметические операции в 4-х системах счисления: сложение, вычитание, умножение, деление.

Написаны динамические и параметризованные unit тесты. Написан и протестирован источник данных для CSV данных для тестов.

Написан отчёт.

[Исходный код доступен по ссылке](https://github.com/Allexeey/autotests-lab1)

ПРИЛОЖЕНИЕ А  
Листинг программы

import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args){  
 Calculate calculate = new Calculate();  
 Scanner in = new Scanner(System.*in*);  
  
 System.*out*.println("Введите систему счисления (Доступны значения - 2, 8, 10, 16): ");  
 int base = in.nextInt();  
 System.*out*.println("Введите первое число: ");  
 String num1 = in.next();  
 System.*out*.println("Введите второе число: ");  
 String num2 = in.next();  
 System.*out*.println("Введите арифметическую операцию (Доступны значения +, -, \*, /): ");  
 String operation = in.next();  
  
 switch (operation){  
 case "+":  
 System.*out*.println(calculate.plus(num1, num2, base));  
 break;  
 case "-":  
 System.*out*.println(calculate.minus(num1, num2, base));  
 break;  
 case "\*":  
 System.*out*.println(calculate.multiply(num1, num2, base));  
 break;  
 case "/":  
 System.*out*.println(calculate.divide(num1, num2, base));  
 default:  
 throw new IllegalStateException("Данная операция не найдена - " + operation);  
 }  
 }  
}

public class Calculate {  
// Класс калькулятор, вызывает 4 метода +, - , \*, \  
 public String plus(String number1, String number2, int base) { // на входе в этот метод поступает два числа со стринг значением, так как в 16-й системе существуют буквы  
 switch (base){ // на основе системы считсления переходим в нужную систему счисления  
 case 2:  
 Integer decimal1 = Integer.*parseInt*(number1, 2);  
 Integer decimal2 = Integer.*parseInt*(number2, 2);  
 int sum1 = decimal1 + decimal2;  
 return Integer.*toBinaryString*(sum1); //перевод в двоичную систему  
 case 8:  
 Integer decimal3 = Integer.*parseInt*(number1, 8);  
 Integer decimal4 = Integer.*parseInt*(number2, 8);  
 int sum2 = decimal3 + decimal4;  
 return Integer.*toOctalString*(sum2); //перевод в 8ю систему  
 case 10:  
 Integer decimal5 = Integer.*parseInt*(number1, 10);  
 Integer decimal6 = Integer.*parseInt*(number2, 10);  
 int sum3 = decimal5 + decimal6;  
 return String.*valueOf*(sum3); //  
 case 16:  
 Integer decimal7 = Integer.*parseInt*(number1, 16);  
 Integer decimal8 = Integer.*parseInt*(number2, 16);  
 int sum4 = decimal7 + decimal8;  
 return Integer.*toHexString*(sum4).toUpperCase();// перевод букв в заглавные  
 default:  
 throw new IllegalStateException("Система счисления не найдена - " + base);  
 }  
 }  
  
 public String minus(String number1, String number2, int base) {  
 switch (base){  
 case 2:  
 Integer decimal1 = Integer.*parseInt*(number1, 2);  
 Integer decimal2 = Integer.*parseInt*(number2, 2);  
 int sum1 = decimal1 - decimal2;  
 return Integer.*toBinaryString*(sum1);  
 case 8:  
 Integer decimal3 = Integer.*parseInt*(number1, 8);  
 Integer decimal4 = Integer.*parseInt*(number2, 8);  
 int sum2 = decimal3 - decimal4;  
 return Integer.*toOctalString*(sum2);  
 case 10:  
 Integer decimal5 = Integer.*parseInt*(number1, 10);  
 Integer decimal6 = Integer.*parseInt*(number2, 10);  
 int sum3 = decimal5 - decimal6;  
 return String.*valueOf*(sum3);  
 case 16:  
 Integer decimal7 = Integer.*parseInt*(number1, 16);  
 Integer decimal8 = Integer.*parseInt*(number2, 16);  
 int sum4 = decimal7 - decimal8;  
 return Integer.*toHexString*(sum4).toUpperCase();  
 default:  
 throw new IllegalStateException("Система счисления не найдена - " + base);  
 }  
 }  
  
 public String multiply(String number1, String number2, int base) {  
 switch (base){  
 case 2:  
 Integer decimal1 = Integer.*parseInt*(number1, 2);  
 Integer decimal2 = Integer.*parseInt*(number2, 2);  
 int sum1 = decimal1 \* decimal2;  
 return Integer.*toBinaryString*(sum1);  
 case 8:  
 Integer decimal3 = Integer.*parseInt*(number1, 8);  
 Integer decimal4 = Integer.*parseInt*(number2, 8);  
 int sum2 = decimal3 \* decimal4;  
 return Integer.*toOctalString*(sum2);  
 case 10:  
 Integer decimal5 = Integer.*parseInt*(number1, 10);  
 Integer decimal6 = Integer.*parseInt*(number2, 10);  
 int sum3 = decimal5 \* decimal6;  
 return String.*valueOf*(sum3);  
 case 16:  
 Integer decimal7 = Integer.*parseInt*(number1, 16);  
 Integer decimal8 = Integer.*parseInt*(number2, 16);  
 int sum4 = decimal7 \* decimal8;  
 return Integer.*toHexString*(sum4).toUpperCase();  
 default:  
 throw new IllegalStateException("Система счисления не найдена - " + base);  
 }  
 }  
  
 public String divide(String number1, String number2, int base) {  
 switch (base){  
 case 2:  
 Integer decimal1 = Integer.*parseInt*(number1, 2);  
 Integer decimal2 = Integer.*parseInt*(number2, 2);  
 int sum1 = decimal1 / decimal2;  
 return Integer.*toBinaryString*(sum1);  
 case 8:  
 Integer decimal3 = Integer.*parseInt*(number1, 8);  
 Integer decimal4 = Integer.*parseInt*(number2, 8);  
 int sum2 = decimal3 / decimal4;  
 return Integer.*toOctalString*(sum2);  
 case 10:  
 Integer decimal5 = Integer.*parseInt*(number1, 10);  
 Integer decimal6 = Integer.*parseInt*(number2, 10);  
 int sum3 = decimal5 / decimal6;  
 return String.*valueOf*(sum3);  
 case 16:  
 Integer decimal7 = Integer.*parseInt*(number1, 16);  
 Integer decimal8 = Integer.*parseInt*(number2, 16);  
 int sum4 = decimal7 / decimal8;  
 return Integer.*toHexString*(sum4).toUpperCase();  
 default:  
 throw new IllegalStateException("Система счисления не найдена - " + base);  
 }  
 }  
}

import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.DisplayName;  
import org.junit.jupiter.api.DynamicTest;  
import org.junit.jupiter.api.TestFactory;  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.CsvSource;  
  
import java.io.BufferedReader;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.util.Objects;  
import java.util.function.Function;  
import java.util.stream.Stream;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.junit.jupiter.api.Assertions.*assertThrows*;  
import static org.junit.jupiter.api.DynamicTest.*dynamicTest*;  
  
public class CalculateTests { // Класс для тестов  
  
 private final Calculate calculate = new Calculate();  
  
 // Параметризованный тест на сложение, на вход передаем первое число, второе число, систему счисления и ожидаем результат  
 @ParameterizedTest(name = "{displayName} => Первое число = {0}, Второе число = {1}, Система счисления = {2}, Ожидаемый результат = {3}")  
 @CsvSource({  
 "0, 1, 2, 1",  
 "101, 1111, 2, 10100",  
 "6, 5, 8, 13",  
 "456, 123, 8, 601",  
 "100, 100, 10, 200",  
 "0, 0, 10, 0",  
 "A15, BC, 16, AD1",  
 "3A5, 1A45F0D, 16, 1A462B2"  
 })  
 @DisplayName("Проверка сложения")  
 public void testPlus(String number1, String number2, Integer numberSystem, String expectedResult) {  
 String actualResult = calculate.plus(number1, number2, numberSystem); // передаем объекты в метод калькулятора  
 System.*out*.println(number1 + " + " + number2 + " = " + actualResult); // debug  
 Assertions.*assertEquals*(expectedResult, actualResult, "Результат сложения не соответствует ожидаемому"); // метод сравнения двух значений, на вход ему подается объект ожидаемый, актуальный и сообщение, если например результаты будут не равны  
 }  
  
 // Параметризованный тест на вычитание  
 @ParameterizedTest(name = "{displayName} => Первое число = {0}, Второе число = {1}, Система счисления = {2}, Ожидаемый результат = {3}")  
 @CsvSource({  
 "01, 1, 2, 0",  
 "1111, 101, 2, 1010",  
 "456, 173, 8, 263",  
 "4, 2, 8, 2",  
 "10, 100, 10, -90",  
 "19, 9, 10, 10",  
 "1BC4, AF, 16, 1B15",  
 "25EB, 8, 16, 25E3"  
 })  
 @DisplayName("Проверка вычитания")  
 public void testMinus(String number1, String number2, Integer numberSystem, String expectedResult) {  
 String actualResult = calculate.minus(number1, number2, numberSystem);  
 System.*out*.println(number1 + " - " + number2 + " = " + actualResult);  
 Assertions.*assertEquals*(expectedResult, actualResult, "Результат вычитания не соответствует ожидаемому");  
 }  
  
 // Параметризованный тест на умножение  
 @ParameterizedTest(name = "{displayName} => Первое число = {0}, Второе число = {1}, Система счисления = {2}, Ожидаемый результат = {3}")  
 @CsvSource({  
 "10001, 1110, 2, 11101110",  
 "11, 0, 2, 0",  
 "7, 7, 8, 61",  
 "5, 21, 8, 125",  
 "6, 8, 10, 48",  
 "200, 21, 10, 4200",  
 "AF, BC4, 16, 80AFC",  
 "14, 5EB, 16, 765C"  
 })  
 @DisplayName("Проверка умножения")  
 public void testMultiply(String number1, String number2, Integer numberSystem, String expectedResult) {  
 String actualResult = calculate.multiply(number1, number2, numberSystem);  
 System.*out*.println(number1 + " \* " + number2 + " = " + actualResult);  
 Assertions.*assertEquals*(expectedResult, actualResult, "Результат умножения не соответствует ожидаемому");  
 }  
  
 // Параметризованный тест на деление  
 @ParameterizedTest(name = "{displayName} => Первое число = {0}, Второе число = {1}, Система счисления = {2}, Ожидаемый результат = {3}")  
 @CsvSource({  
 "1111, 110, 2, 10",  
 "10010, 110, 2, 11",  
 "2374, 35, 8, 54",  
 "13351, 163, 8, 63",  
 "1, 0, 10, zero",  
 "30, 5, 10, 6",  
 "DE11, 66, 16, 22D",  
 "748, 0, 16, zero"  
 })  
 @DisplayName("Проверка деления")  
 public void testDivide(String number1, String number2, Integer numberSystem, String expectedResult) {  
 if (number2.equals("0")) { // проверка деления на ноль  
 Exception exception = *assertThrows*(ArithmeticException.class, () -> calculate.divide(number1, number2, numberSystem));  
 Assertions.*assertEquals*("/ by zero", exception.getMessage(), "Сообщение исключения не соответствует ожидаемому");  
 } else {  
 String actualResult = calculate.divide(number1, number2, numberSystem);  
 System.*out*.println(number1 + " / " + number2 + " = " + actualResult);  
 Assertions.*assertEquals*(expectedResult, actualResult, "Результат деления не соответствует ожидаемому");  
 }  
 }  
 // Динамический тест на умножение  
 @TestFactory  
 @DisplayName("Проверка умножения. Динамический тест")  
 public Stream<DynamicTest> testCalculate() {  
 InputStream testData = getClass().getClassLoader().getResourceAsStream("testData.csv"); // получем из каталога resources файл testData.csv  
 BufferedReader br = new BufferedReader(new InputStreamReader(Objects.*requireNonNull*(testData)));  
 return br.lines().skip(1).map(mapCsvLineToDynamicTest()); // делаем map  
 }  
  
 private Function<String, DynamicTest> mapCsvLineToDynamicTest() { // метод map  
 return s -> { // построчно передаем значения, которые получаются  
 final String[] values = s.split(";"); // используем регулярку со сплитом  
 return *dynamicTest*(values[0].replace("{0}", values[1]).replace("{1}", values[2]), () -> assertCalculate(values)); // метод dynamicTest  
 };  
 }  
  
 private void assertCalculate(String[] values) { // метод asserCalculate для динамических тестов  
 String result = calculate.multiply(values[1], values[2], Integer.*parseInt*(values[3]));  
 System.*out*.println(values[1] + " \* " + values[2] + " = " + result);  
 *assertEquals*(values[4], result, "Результат умножения не соответствует ожидаемому");  
 }  
}