Plan

2023-10-13

Repetition: Wrapper, String, Unit test

Повторение: Wrapper, String, Unit test

Theory

▶ **English**

▼ **На русском**

Классы Wrapper:

Классы-оболочки (Wrapper classes) используются для оборачивания примитивных типов данных в объекты. В Java существуют следующие классы-оболочки:

- `Integer` для `int`
- `Double` для `double`
- `Boolean` для `boolean`
- `Character` для `char` и так далее.

Методы класса String:

Класс `String` в Java предоставляет множество методов для работы со строками. Вот некоторые из них:

- `length()` - возвращает длину строки.
- `charAt(int index)` - возвращает символ по указанному индексу.
- `substring(int beginIndex, int endIndex)` - возвращает подстроку из строки.
- `toUpperCase()` и `toLowerCase()` - изменяют регистр символов строки.
- `indexOf(String str)` и `lastIndexOf(String str)` - Возвращает индекс первого и последнего вхождения подстроки в строке соответственно.
- `replace(char oldChar, char newChar)` - Заменяет все вхождения символа `oldChar` на `newChar`.
- `trim()` - Удаляет пробелы в начале и в конце строки
- `split(String regex)`: Разбивает строку на массив подстрок, используя регулярное выражение.

Unit Testing:

Unit Testing - это процесс проверки отдельных компонентов (или "юнитов") программы для убеждения в их правильной работы. В Java, для тестирования используется фреймворк JUnit.

Основные методы класса Assertions в JUnit:

Конечно, вот основные методы класса Assertions в виде таблицы:

Метод	Описание
`assertEquals(expected, actual)`	Проверяет, что `expected` равно `actual`.
`assertNotEquals(expected, actual)`	Проверяет, что `expected` не равно `actual`.
`assertTrue(condition)`	Проверяет, что `condition` истинно.
`assertFalse(condition)`	Проверяет, что `condition` ложно.
`assertNull(object)`	Проверяет, что `object` является `null`.
`assertNotNull(object)`	Проверяет, что `object` не является `null`.
`assertArrayEquals(expectedArray, actualArray)`	Проверяет, что два массива expected и actual равны по содержимому.

Задачи:

1. Реализовать 5 методов для калькулятора (сложение, вычитание, умножение, деление с остатком, целая часть от деления), который работает с целыми числами. Покрыть все методы тестами.
2. Напишите программу, которая удаляет все вхождения определенного символа из строки и напишите тесты.
3. Напишите программу, которая определит, содержит ли строка только уникальные символы (без повторений) и напишите тесты.

Техническое задание для программы проверки пароля

Описание: Создайте класс `PasswordValidator` на Java для проверки пароля на соответствие требованиям, которые будут устанавливаться через конструктор класса. Класс `PasswordValidator` должен быть протестирован.

Требования:

1. Пароль должен содержать минимум заданное количество букв нижнего регистра.
2. Пароль должен содержать минимум заданное количество букв верхнего регистра.
3. Пароль должен содержать минимум заданное количество цифр.
4. Пароль должен иметь заданную длину.
5. Пароль должен содержать хотя бы один из символов, указанных в списке символов.
6. Количество символов из списка должно быть не менее определенного значения.

Интерфейс:

1. Создайте класс `PasswordValidator` с полями, определенными в конструкторе:
   - `minLowerCase` (int): Минимальное количество букв нижнего регистра.
   - `minUpperCase` (int): Минимальное количество букв верхнего регистра.
   - `minDigits` (int): Минимальное количество цифр.
   - `minLength` (int): Минимальная длина пароля.
   - `symbolList` (String): Список символов, которые должны быть в пароле.
   - `minSymbolCount` (int): Минимальное количество символов из списка.
2. В классе `PasswordValidator` создайте метод `isValid`, который принимает строку (пароль) для проверки и возвращает `true`, если пароль соответствует всем требованиям, и `false` в противном случае.

Пример использования:

```
public class Main {
    public static void main(String[] args) {
        int minLowerCase = 2;
        int minUpperCase = 2;
        int minDigits = 1;
        int minLength = 12;
        String symbolList = "!@#$%^";
        int minSymbolCount = 2;

        PasswordValidator validator = new PasswordValidator(minLowerCase, m

        String password = "MyP@ssword123";
        boolean isValid = validator.isValid(password);

        if (isValid) {
            System.out.println("Пароль верный.");
        } else {
            System.out.println("Пароль не соответствует требованиям.");
        }
    }
}
```

Скелет класса `PasswordValidator`:

```
/**
 * @author Andrej Reutow
 * created on 09.10.2023
 * <p>
 * Класс для проверки пароля на соответствие заданным требованиям.
```

```
 */
public class PasswordValidator {

    private final int minLowerCase;
    private final int minUpperCase;
    private final int minDigits;
    private final int minLength;
    private final String symbolList;
    private final int minSymbolCount;

    /**
     * Конструктор класса PasswordValidator для инициализации параметров пр
     *
     * @param minLowerCase   Минимальное количество букв нижнего регистра.
     * @param minUpperCase   Минимальное количество букв верхнего регистра.
     * @param minDigits      Минимальное количество цифр.
     * @param minLength      Минимальная длина пароля.
     * @param symbolList     Список символов, которые должны быть в пароле.
     * @param minSymbolCount Минимальное количество символов из списка.
     */
    public PasswordValidator(int minLowerCase,
                             int minUpperCase,
                             int minDigits,
                             int minLength,
                             String symbolList,
                             int minSymbolCount) {

    }

    /**
     * Проверяет, соответствует ли заданный пароль требованиям.
     *
     * @param password Пароль для проверки.
     * @return true, если пароль соответствует требованиям, и false в проти
     */
    public boolean isValid(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданное количество цифр.
     *
```

```
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество цифр, и false
     */
    public boolean isDigitsContains(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданные символы из списка.
     *
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество символов из с
     */
    public boolean isSymbolsContains(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданное количество символов верхнего
     *
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество символов верх
     */
    public boolean isUpperCaseContains(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданное количество символов нижнего р
     *
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество символов нижн
     */
    public boolean isLowerCaseContains(String password) {

        return false;
    }

    /**
     * Проверяет, является ли длина пароля достаточной.
```

```
     *
     * @param password Пароль для проверки.
     * @return true, если длина пароля больше или равна минимальной длине,
     */
    public boolean isLengthValid(String password) {
        return -1;
    }
}
```

Homework

▶ **English**

▼ **На русском**

1. Напишите программу, которая принимает строку с клавиатуры и определяет, является ли она палиндромом (строкой, которая читается одинаково с начала и с конца, игнорируя пробелы и регистр символов) и напишите тесты.

2. Напишите программу, которая которая сравнивает две версии строковых чисел (например, "1.2.3" и "2.0.1") и возвращает результат сравнения и напишите тесты.

   Если 'version1' меньше чем 'version2', метод возвращает '-1'. Если 'version1' равен 'version2', метод возвращает '0'. Если 'version1' больше чем 'version2', метод '1'.

   скелет метода:

```
public static int compareVersion(String version1, String version2) {
    //   Если 'version1' меньше чем 'version2', метод возвращает '-1'.
    //   Если 'version1' равен 'version2',      метод возвращает '0'.
    //   Если 'version1' больше чем 'version2', метод '1'.
}
```

Пример: compareVersion("1.2.3", "2.0.1") // метод должен вернуть -1

compareVersion("1.2.3", "1.2.3") // метод должен вернуть 0

compareVersion("2.0.1", "1.2.3") // метод должен вернуть 1

Code

code/Lesson_28/src/Wrappers.java

```java
import java.util.Scanner;

/**
 * @author Andrej Reutow
 * created on 13.10.2023
 */
public class Wrappers {

    public static void main(String[] args) {
        int intNumber = 10; // нет методов у примитивных типов
        Integer integerNumber = intNumber;


        String stringNumber = String.valueOf(true); // "true"

        Integer parsedInteger = Integer.parseInt("2023");
        Double parsedDouble = Double.parseDouble("2023");


        char[] chars = {'H', 'e', 'l', 'l', 'o'};
        String stringValueOfChars = String.valueOf(chars); // "hello"
        System.out.println(stringValueOfChars);

        int value = Integer.valueOf(integerNumber);

        boolean b = true;
        Boolean parsedBoolean = Boolean.parseBoolean("true");


        int x = 1; // true
        int y = 0; // false


        String line = "158949 #65+8198Max150000.99$";

        String col1 = "158949658198";
        String col2 = "Max";
        String col3 = "150000.99";

        long id = Long.parseLong(col1);
        String name = col2;
        Double sum = Double.parseDouble(col3);
```

```
//        Scanner scanner = new Scanner(System.in);
//        String action = scanner.nextLine();
//        Integer selectedActionAsNumber = Integer.parseInt(action);
//
//        if (selectedActionAsNumber == 1) {
//            System.out.println("Вы выбрали пункт 1");
//        }

        System.out.println(getDigits(line));
    }

    public static String getDigits(String source) { // вернуть из входящей
        if (source == null){
            return null;
        }

        String res = "";
        for (int i = 0; i < source.length(); i++) {
            char currentChar = source.charAt(i);
            if (currentChar >= '0' && currentChar <= '9') { // 0 1 2 3 ... 
//                res += currentChar;
                res = res + currentChar;
            }
        }
        return res;
    }
}
```

code/Lesson_28/src/WrappersTest.java

```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

/**
 * @author Andrej Reutow
 * created on 13.10.2023
 */
public class WrappersTest {

    @Test
    public void test_getDigits_validString_digitsString() {
        // Given
        String testString = "a17b8";
```

```
        // When
        String result = Wrappers.getDigits(testString);
        // Then
        String expectedResult = "178";
        Assertions.assertNotNull(result);
        Assertions.assertEquals(expectedResult, result);
    }

    @Test
    @DisplayName("Тестирование метода getDigits на вход строка со значением
    public void test_getDigits_inputValueIsNull_resultIsNull() {
        // Given
        String testString = null;
        // When
        String result = Wrappers.getDigits(testString);
        // Then
        Assertions.assertNull(result);
    }

    @Test
    public void test_getDigits_valueIsEmptyString_emptyString() {
        // Given
        String testString = "";
        // When
        String result = Wrappers.getDigits(testString);
        // Then
        String expectedResult = "";
        Assertions.assertNotNull(result);
        Assertions.assertEquals(expectedResult, result);
    }
}
```

code/PasswordValidator/src/test/PasswordValidatorTest.java

```
package test;


/**
 * @author Andrej Reutow
 * created on 13.10.2023
 */
public class PasswordValidatorTest {
```

```
}
```

code/PasswordValidator/src/Application.java

```
/**
 * @author Andrej Reutow
 * created on 13.10.2023
 */
public class Application {

    public static void main(String[] args) {
        int minLowerCase = 2;
        int minUpperCase = 2;
        int minDigits = 1;
        int minLength = 12;
        String symbolList = "!@#$%^";
        int minSymbolCount = 2;

        PasswordValidator validator1 = new PasswordValidator(minLowerCase, 

        PasswordValidator validator2 = new PasswordValidator(5, 1, 2, 10, s


        String password = "MyP@ssword123!";
        boolean isValid1 = validator1.isValid(password);            // true
        boolean isValid2 = validator2.isValid(password);            // fals

        if (isValid1) {
            System.out.println("Пароль верный.");
        } else {
            System.out.println("Пароль не соответствует требованиям.");
        }
    }
}
```

code/PasswordValidator/src/PasswordValidator.java

```
/**
 * @author Andrej Reutow
 * created on 09.10.2023
 * <p>
```

```
 * Класс для проверки пароля на соответствие заданным требованиям.
 */
public class PasswordValidator {

//    private final int minLowerCase;
//    private final int minUpperCase;
//    private final int minDigits;
//    private final int minLength;
//    private final String symbolList;
//    private final int minSymbolCount;

    /**
     * Конструктор класса PasswordValidator для инициализации параметров пр
     *
     * @param minLowerCase   Минимальное количество букв нижнего регистра.
     * @param minUpperCase   Минимальное количество букв верхнего регистра.
     * @param minDigits      Минимальное количество цифр.
     * @param minLength      Минимальная длина пароля.
     * @param symbolList     Список символов, которые должны быть в пароле.
     * @param minSymbolCount Минимальное количество символов из списка.
     */
    public PasswordValidator(int minLowerCase,
                             int minUpperCase,
                             int minDigits,
                             int minLength,
                             String symbolList,
                             int minSymbolCount) {
    }

    /**
     * Проверяет, соответствует ли заданный пароль требованиям.
     *
     * @param password Пароль для проверки.
     * @return true, если пароль соответствует требованиям, и false в проти
     */
    public boolean isValid(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданное количество цифр.
     *
```

```
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество цифр, и false
     */
    public boolean isDigitsContains(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданные символы из списка.
     *
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество символов из с
     */
    public boolean isSymbolsContains(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданное количество символов верхнего
     *
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество символов верх
     */
    public boolean isUpperCaseContains(String password) {

        return false;
    }

    /**
     * Проверяет, содержит ли пароль заданное количество символов нижнего р
     *
     * @param password Пароль для проверки.
     * @return true, если пароль содержит заданное количество символов нижн
     */
    public boolean isLowerCaseContains(String password) {

        return false;
    }

    /**
     * Проверяет, является ли длина пароля достаточной.
```

```
     *
     * @param password Пароль для проверки.
     * @return true, если длина пароля больше или равна минимальной длине,
     */
    public boolean isLengthValid(String password) {
        return false;
    }
}
```