Plan

2023-10-10

1. String
2. Character
3. Practice

---

1. String
2. Character
3. Практика

Theory

▶ **English**

▼ **На русском**

Класс `String` в Java

Строки в Java представлены классом `String`. Этот класс предоставляет множество методов для различных операций со строками.

Создание строк

Строки можно создавать разными способами:

```
String str1="Привет, мир!";
String str2=new String("Привет, мир!");
```

Основные методы

- **`length()`**: Возвращает длину строки.

```
int len = str1.length();  // 12
```

- **`charAt(int index)`**: Возвращает символ строки по указанному индексу.

```
char ch = str1.charAt(0);  // 'П'
```

- **`substring(int beginIndex, int endIndex)`**: Возвращает подстроку, начиная с `beginIndex` и заканчивая `endIndex - 1`.

```
String sub = str1.substring(0, 6);  // "Привет"
```

- **`indexOf(String str)`** и **`lastIndexOf(String str)`**: Возвращает индекс первого и последнего вхождения подстроки в строке соответственно.

```
int first = str1.indexOf('м');  // 8
int last = str1.lastIndexOf('т');  // 5
```

- **`replace(char oldChar, char newChar)`**: Заменяет все вхождения символа `oldChar` на `newChar`.

```
String replaced = str1.replace(' ', '_');  // "Привет,_мир!"
```

- **`toLowerCase()`** и **`toUpperCase()`**: Возвращает новую строку, где все символы преобразованы к нижнему или верхнему регистру.

```
String lower = str1.toLowerCase();  // "привет, мир!"
String upper = str1.toUpperCase();  // "ПРИВЕТ, МИР!"
```

- **`trim()`**: Удаляет пробелы в начале и в конце строки.

```
String trimmed = "  Привет, мир!  ".trim();  // "Привет, мир!"
```

- **`split(String regex)`**: Разбивает строку на массив подстрок, используя регулярное выражение.

```
String[] words = str1.split(" ");  // ["Привет,", "мир!"]
```

Это лишь некоторые из многочисленных методов, предоставляемых классом `String`. С их помощью можно эффективно манипулировать строками и производить различные операции.

Работа с типом данных `char` и Unicode в Java

Описание типа данных `char`

В Java, `char` — это примитивный тип данных, используемый для хранения одного символа Unicode. Он занимает 16 бит и может представлять символы в диапазоне от `\u0000` (0) до `\uffff` (65,535).

Пример:

```
char myChar = 'A'; // или char myChar = '\u0041';
```

Упаковка и распаковка типа `char`

Примитивный `char` можно упаковать в объект типа `Character`.

Пример упаковки и распаковки:

```
char primitiveChar = 'B';
Character wrappedChar = Character.valueOf(primitiveChar); // Упаковка
char anotherPrimitiveChar = wrappedChar.charValue(); // Распаковка
```

Приведение к строке

Char можно легко привести к строке различными способами:

1. Конкатенация с пустой строкой: `myChar + ""`
2. Использование `String.valueOf(myChar)`
3. Конструктор `String`: `new String(new char[]{myChar})`

Логические операторы

Символы можно сравнивать с помощью стандартных логических операторов (`<`, `>`, `<=`, `>=`, `==`, `!=`), поскольку их численные Unicode-значения используются в сравнениях.

Unicode и кодовые точки

Unicode — это стандарт кодирования, представляющий текст в компьютерах. Каждому символу соответствует уникальная кодовая точка, представленная в шестнадцатеричной системе счисления. Например, символ "A" имеет кодовую точку U+0041.

Зачем использовать `\u`?

Символ `\u` используется для экранирования Unicode-символов. Это позволяет работать с символами, которые недоступны на клавиатуре или не отображаются в текстовом редакторе.

Техническое задание для программы проверки пароля

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

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.

3. Использование методов **Character** класса:

- Для проверки, является ли символ буквой нижнего регистра, используйте метод `Character.isLowerCase(char c)`.
- Для проверки, является ли символ буквой верхнего регистра, используйте метод `Character.isUpperCase(char c)`.
- Для проверки, является ли символ цифрой, используйте метод `Character.isDigit(char c)`.

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
     */
    private 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
     */
    private 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
     */
    private 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
     */
    private 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,
     */
    private boolean isLengthValid(String password) {
        return -1;
    }
}
```

Homework

▶ **English**

▼ **На русском**

Задача 1. Реализуйте метод, который подсчитает количество цифр в строке.

Пример 1: Дана строка "I am agent 007", Результат: В строке 3 цифр(ы)

Пример 2: Дана строка "In 2022, I went to the sea twice", Результат: В строке 4 цифр(ы)

Пример 3: Дана строка "I was in Berlin 3 times in 2023, and in 2022 I was there twice", Результат: В строке 9 цифр(ы)

- попробуйте разные подходы, с разбиением стоки на массив символов и с использованием charAt

Задача 2. Реализуйте метод, который подсчитает количество только верхнего регистра в строке от A до Z.

Пример 1: Дана строка "Hello World", Результат: 2 букв(ы) верхнего регистра

Пример 2: Дана строка "In 2022, I went to the sea twice", Результат: 2 букв(ы) верхнего регистра

Пример 3: Дана строка "I was in Berlin 3 times in 2023, and in 2022 I was there twice", Результат: 3 букв(ы) верхнего регистра

Задача 3*. Реализуйте метод, который принимает строку и возвращает новую строку, в которой все слова перевернуты, но порядок слов остается прежним.

Пример 1: Дана строка "Hello World", Результат: "olleH dlroW"

Пример 2: Дана строка "Java Programming", Результат: "avaJ gnimmargorP"

Пример 3: Дана строка "Easy come easy go", Результат: "ysaE emoc ysae og"

Задача 4: Разобраться с решением этой задачи:

Смена регистра символов

Напишите программу, которая принимает символ в верхнем регистре и преобразует его в нижний регистр, и наоборот.

Обратите внимание на эти строки:

```
if (inputChar >= 'A' && inputChar <= 'Z')
else if (inputChar >= 'a' && inputChar <= 'z')
```

```
public class CharUtils {

    public static char toUpperCase(char inputChar) {
        char outputChar = inputChar;

        if (inputChar >= 'A' && inputChar <= 'Z') {             // Обратите
            outputChar = (char) (inputChar + 'a' - 'A');
        } else if (inputChar >= 'a' && inputChar <= 'z') {      // Обратите
            outputChar = (char) (inputChar - ('a' - 'A'));
        } else {
            outputChar = inputChar;
        }

        return outputChar;
```

```
        }

    }
```

▼ Объяснение 1

1. Каждая буква в Unicode имеет числовой код. Для примера, символ `'A'` имеет числовой код 65, а `'a'` — 97 в десятичной системе.
2. Проверяем, находится ли введенный символ в диапазоне от 65 до 90 (верхний регистр). Если это так, то к числовому коду символа добавляем разницу между числовыми кодами `'a'` (97) и `'A'` (65), которая равна 32. Это преобразует символ в нижний регистр.

   Например, для `'A'` (числовой код 65): `65 + (97 - 65) = 65 + 32 = 97`, что является числовым кодом для `'a'`.
3. Аналогично, если символ в нижнем регистре (числовой код между 97 и 122), мы вычитаем 32, чтобы преобразовать его в верхний регистр.

   Например, для `'a'` (числовой код 97): `97 - (97 - 65) = 97 - 32 = 65`, что является числовым кодом для `'A'`.
4. Если числовой код символа не соответствует ни одному из этих диапазонов, символ остается без изменений.

▼ Объяснение 2

1. Для этого сначала проверяется, является ли символ буквой в верхнем регистре (`inputChar >= 'A' && inputChar <= 'Z'`).
2. Если это так, символ преобразуется в соответствующий символ в нижнем регистре. Это делается путем добавления разницы между `'a'` и `'A'` к символу. Например, если `inputChar` равен `'A'`, то `'A' + ('a' - 'A')` будет равно `'a'`.
3. Аналогично, если символ в нижнем регистре (`inputChar >= 'a' && inputChar <= 'z'`), он преобразуется в верхний регистр, вычитая разницу между `'a'` и `'A'`.
4. Если символ не является буквой алфавита, он остается без изменений.

Этот подход использует арифметические операции с символами, что возможно, потому что в Java символы представлены их числовыми Unicode-значениями.

Code

code/PasswordValidator_25/src/PasswordValidator.java

```
404: Not Found
```

code/Lesson_25/src/string/StringTasks.java

```java
package string;

import java.util.Arrays;

/**
 * @author Andrej Reutow
 * created on 10.10.2023
 */

// левый ctrl + q - обращение к документации
public class StringTasks {

    public static void main(String[] args) {
        String helloWorldString = "Hello World";
        String helloWorldString2 = "Hello World";
        String helloWorldString3 = new String("Hello World");

        System.out.println(helloWorldString == helloWorldString2); // true
        System.out.println(helloWorldString2 == helloWorldString3); // fals
        System.out.println(helloWorldString2.equals(helloWorldString3)); //
        System.out.println(helloWorldString.equals(helloWorldString2)); //

        helloWorldString2.length(); // 11
        helloWorldString3.length(); // 11

        // charAt(int index): Возвращает символ строки по указанному индекс
        System.out.println("\ncharAt0(int index): Возвращает символ строки
        char charAt0 = helloWorldString.charAt(0); // char 'H'
        System.out.println("Символ по индексу 0: " + (int) charAt0);

        char charLastChar = helloWorldString.charAt(helloWorldString.length
        System.out.println("Последний символ в строке: " + charLastChar);
        iterateStringWithCharAt(helloWorldString);
        iterateStringCharArray(helloWorldString3);

        // helloWorldString.charAt(1000); // throw StringIndexOutOfBoundsEx

        // substring(int beginIndex, int endIndex): Возвращает подстроку, на
        System.out.println("\nsubstring(int beginIndex, int endIndex): Возв
```

```
        // "Hello World"
        String subsctrResultAtIndex6 = helloWorldString.substring(6); // во
        System.out.println("Substring from index 6 " + subsctrResultAtIndex

        String subsctrResultFromIndex = helloWorldString.substring(2, 9); /
        System.out.println("Substring from index 6 " + subsctrResultFromInd

        System.out.println("\nНайти подстроку от символа до символа");
        System.out.println("результат 1: " + substringFromCharAToCharB(hell
        System.out.println("результат 2: " + substringFromCharAToCharB(hell

        System.out.println("\nindexOf(), lastIndexOf()");
        int fromInt = helloWorldString.indexOf('o');
        int toInt = helloWorldString.indexOf('r');
        int toIntLastIndex = helloWorldString.lastIndexOf('r');
        System.out.println("результат 3: " + helloWorldString.substring(fro

        fromInt = helloWorldString.indexOf('l');
        toInt = helloWorldString.lastIndexOf('l');
        System.out.println("результат 4: " + helloWorldString.substring(fro

        System.out.println("\nreplace(char oldChar, char newChar)");
        // Hello World
        // He##o Wor#d
        System.out.println(helloWorldString.replace('l', '#'));

        System.out.println("\ntoLowerCase() и toUpperCase(): Возвращает нов
        System.out.println("To upper case:" + "lowerCaseString".toUpperCase
        System.out.println("To upper case:" + helloWorldString.toUpperCase(
        System.out.println("To upper case:" + new String("some str").toUppe
        System.out.println("To upper case:" + "Hello я Андрей агент 007".to
        System.out.println("To lower case: " + "UPPER_CASE_STRING".toLowerC

        System.out.println("\ntrim(): Удаляет пробелы в начале и в конце ст
        String email = " user@email.com";
        String emailToLogin = "user@email.com";
        System.out.println("Find user by email: " + emailToLogin.equals(ema
        String trimEmail = email.trim();
        System.out.println("Find user by email: " + emailToLogin.equals(tri

        System.out.println("###");
        String someStr = "\n\t Hello\n";
        System.out.println(someStr); // перенос строки, с новой строки с пр
```

```
        System.out.println(someStr.trim());
        System.out.println("###");

        System.out.println("Hi my name is \t Andrej");
        System.out.println("Hi my name is \t Andrej".trim());

        System.out.println("Hi my name is \n Andrej");
        System.out.println("Hi my name is \n Andrej".trim());
        System.out.println("Hi my name is \n Andrej".replace(" \n ", ""));
        System.out.println("Hi my name is \n Andrej".replace("Andrej", "Vas
        System.out.println("Andrej Hi Andrej my name is \n Andrej Andrej".r
        System.out.println("Hi my name is \n Andrej".replace("\n Andrej", "

        Integer age = 40;
        Integer newAge = 18;
        System.out.println("i am 40 years old".replace(age.toString(), newA


        System.out.println("\nsplit(String regex): Разбивает строку на масс
        String name = "Dr. Vasja Pupkin";
        String name2 = "Vasja Pupkin";
        String title; // "Dr."
        String firstName; // "Vasja"
        String lastName; // "Pupkin"
        String[] strings = name2.split(" ");
        // name ->  {"Dr.", "Vasja", "Pupkin"}
        // name2 ->  {"Vasja", "Pupkin"}
        if (strings.length == 3) {
            title = strings[0];
            firstName = strings[1];
            lastName = strings[2];
        } else if (strings.length == 2) {
            title = null;
            firstName = strings[0];
            lastName = strings[1];
        } else if (strings.length == 1) {
            title = null;
            firstName = strings[0];
            lastName = null;
        } else {
            title = null;
            firstName = null;
            lastName = null;
```

```
        }
        System.out.println(Arrays.toString(strings));
        System.out.println("title " + title);
        System.out.println("firstName " + firstName);
        System.out.println("lastName " + lastName);

        System.out.println("split using ,");
        String csvFile = "Max;Pupkin;email@main.com";
        String[] csvLine = csvFile.split(";");
        System.out.println(Arrays.toString(csvLine));
    }

    // обрезать строку от символа до символа и вернуть результат
    static String substringFromCharAToCharB(String source, char from, char
        String result = "";
        int fromInt = findIndexOfChar(source, from);
//        int toInt = findIndexOfChar(source, to);
        int toInt = findLastIndexOfChar(source, to);

        if (fromInt >= 0 && toInt >= 0) {
            return source.substring(fromInt, toInt + 1);
        }

        return result;

        // "Hello World"
        // char from = 'o'
        // char to = 'r'
        // результат: "o Wor"
    }

    private static int findIndexOfChar(String source, char target) {
        for (int i = 0; i < source.length(); i++) {
            if (source.charAt(i) == target) {
                return i;
            }
        }
        return -1;
    }

    private static int findLastIndexOfChar(String source, char target) {
        for (int i = source.length() - 1; i >= 0; i--) {
            if (source.charAt(i) == target) {
```

```
                return i;
            }
        }
        return -1;
    }


    /**
     * перебор строки и печать каждого символа в консоль с указанием самого
     *
     * @param text строка для перебора
     */
    public static void iterateStringWithCharAt(String text) {
//        System.out.println("индекс: " + 0 + " символ: " + text.charAt(0))
//        System.out.println("индекс: " + 1 + " символ: " + text.charAt(1))
//        System.out.println("индекс: " + 2 + " символ: " + text.charAt(2))

        for (int i = 0; i < text.length(); i++) {
            System.out.println("индекс: " + i + " символ: " + text.charAt(i
        }
    }

    // комметарий
    static void iterateStringCharArray(String text) {
        char[] charArray = text.toCharArray();

        for (int i = 0; i < charArray.length; i++) {
            System.out.println("индекс: " + i + " символ: " + charArray[i])
        }
    }
}
```

code/Lesson_25/src/wrapper_char/CharTasks.java

```
package wrapper_char;

import string.StringTasks;

/**
 * @author Andrej Reutow
 * created on 10.10.2023
 */
```

```
public class CharTasks {

//    public static boolean hasUpperCaseLetter(String value) {
//        for (int i = 0; i < value.length(); i++) {
//            char currentChar = value.charAt(i);
//            if (currentChar >= 'A' && currentChar <= 'Z') {
//                return true;
//            }
//        }
//        return false;
//    }

    public static boolean hasUpperCaseLetter(String value) {
        for (int i = 0; i < value.length(); i++) {
            char currentChar = value.charAt(i);
            if (currentChar >= 65 && currentChar <= 93) {
                return true;
            }
        }
        return false;
    }
}
```

code/Lesson_25/src/wrapper_char/CharTasksApp.java

```
package wrapper_char;

/**
 * @author Andrej Reutow
 * created on 10.10.2023
 */
public class CharTasksApp {

    public static void main(String[] args) {

        String testStr1 = "abcd123";
        System.out.println("String " + testStr1 + " has upper case letter(s

        String testStr2 = "abcd123A";
        System.out.println("String " + testStr2 + " has upper case letter(s

        String testStr3 = "A";
        System.out.println("String " + testStr3 + " has upper case letter(s
```

```
        String testStr4 = "a";
        System.out.println("String " + testStr4 + " has upper case letter(s

        printAllUpperCaseLetters();
    }


    public static void printAllUpperCaseLetters() {
//        for (char i = 'A'; i <= 'Z'; i++) {
//            System.out.println("code: "+ (int)i + ", symbol: " + i);
//        }

        for (int i = 65; i <= 'Z'; i++) { // i <= 'Z' -> i <= 90
            System.out.println("code: "+ i + ", symbol: " + (char)i);
        }
    }
}
```

code/Lesson_25/src/wrapper_char/WrapperCharacter.java

```
package wrapper_char;

/**
 * @author Andrej Reutow
 * created on 10.10.2023
 */
public class WrapperCharacter {

    public static void main(String[] args) {
        char iconArrow = '\u27B3'; // ➳
        char iconA = '\u0041';
        char letterA = 65;
        char letterAAsChar = 'A';
        char letterALowerCase = 'a';

        System.out.println(iconArrow);
        System.out.println(iconA);
        System.out.println(letterA);

        System.out.println((int) iconArrow);

        char arrow = 10163;
        System.out.println(arrow);
```

```
        System.out.println('\uBDE7'); // 뷧

        char someChar = '➳';

        String testString = "➳뷧A";
        System.out.println();

        char charA = 65; // A
        char charB = 66; // B

        System.out.println(charB);
        System.out.println(charA);
        System.out.println((charB > charA));
    }
}
```