Лабораторна робота №3

Посилальні типи даних мови Java

Варіанти завдання

Завдання 3.1.

Варіант 1. В кожному слові тексту k-у літеру замінити вказаним символом. Якщо k більше довжини слова, коригування не виконувати.

Варіант 2. В тексті кожну літеру замінити її порядковим номером в алфавіті. При виведенні в одному рядку друкувати текст з двома пропусками між літерами, в наступному рядку внизу під кожною літерою друкувати її номер.

Варіант 3. В тексті після літери Р, якщо вона не остання в слові, помилково надрукована літера А замість О. Внести виправлення в текст.

Варіант 4. В тексті слова заданої довжини замінити вказаним рядком, довжина якого може не співпадати з довжиною слова.

Варіант 5. В тексті після k-го символу вставити заданий підрядок.

Варіант 6. Після кожного слова тексту, яке закінчується заданим підрядком, вставити вказане слово.

Варіант 7. В залежності від ознаки (0 або 1) в кожному рядку тексту вилучити вказаний символ скрізь, де він зустрічається, або вставити його після k-го символу.

Варіант 8. З невеликого тексту вилучити всі символи, крім пропусків, які не є літерами. Між послідовностями літер, що йдуть одна за одною, залишити хоча би один пропуск.

Варіант 9. З тексту вилучити всі слова вказаної довжини, які починаються на приголосну літеру.

Варіант 10. Вилучити з тексту його частину, що обмежена двома символами, які вводяться (наприклад, між дужками(’ та ‘)’ або між зірочками ‘*’ і т.д.).

Завдання 3.2.

Створити класи, специфікації яких наведені нижче. Визначити конструктори та методи setТип(), getТип(), toString(). Визначити додатково методи в класі, що створює масив об'єктів. Задати критерій вибору даних та вивести ці дані на консоль.

Варіант 1. **Student**: id, Прізвище, Ім'я, По батькові, Дата народження, Адреса, Телефон, Факультет, Курс, Група.

Скласти масив об'єктів. Вивести:

a) список студентів заданого факультету;
b) списки студентів для кожного факультету та курсу;
c) список студентів, які народились після заданого року;
d) список навчальної групи.

Варіант 2. **Customer**: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Номер банківського рахунку.

Скласти масив об'єктів. Вивести:

a) список покупців в алфавітному порядку;
b) список покупців, у яких номер кредитної картки знаходиться в заданому інтервалі;
c) список покупців, імена яких починаються із вказаного рядку

Варіант 3. **Patient**: id, Прізвище, Ім'я, По батькові, Адреса, Телефон, Номер медичної карти, Діагноз.

Скласти масив об'єктів. Вивести:

a) список пацієнтів, які мають вказаний діагноз;
b) список пацієнтів, номер медичної карти у яких знаходиться в заданому інтервалі;
c) список пацієнтів, телефони яких закінчується на вказані цифри.

Варіант 4. **Abiturient**: id, Прізвище, Ім'я, По батькові, Адреса, Телефон, Оцінки.

Скласти масив об'єктів. Вивести:

a) список абітурієнтів, які мають незадовільні оцінки;
b) список абітурієнтів, середній бал у яких вище заданого;
c) вибрати задане число n абітурієнтів, що мають найвищий середній бал (вивести також повний список абітурієнтів, що мають напівпрохідний бал).

Варіант 5. **Book**: id, Назва, Автор(и), Видавництво, Рік видання, Кількість сторінок, Ціна, Обкладинка.

Скласти масив об'єктів. Вивести:

a) список книг заданого автора;
b) список книг, що видані заданим видавництвом;
c) список книг, що випущені після заданого року.

Варіант 6. **House**: id, Номер квартири, Площа, Поверх, Кількість кімнат, Вулиця, Тип будівлі, Термін експлуатації.

Скласти масив об'єктів. Вивести:

a) список квартир, які мають задане число кімнат;
b) список квартир, які мають задане число кімнат та розташовані на поверсі, який знаходиться в заданому проміжку;
c) список квартир, які мають площу, що перевищує задану.

Варіант 7. **Phone**: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Дебет, Кредит, Час міських та міжміських розмов.

Скласти масив об'єктів. Вивести:

a) відомості про абонентів, у яких час міських розмов перевищує заданий;
b) відомості про абонентів, які користувались міжміським зв'язком;
c) відомості про абонентів в алфавітному порядку.

Варіант 8. **Car**: id, Марка, Модель, Рік випуску, Колір, Ціна, Реєстраційний номер.

Скласти масив об'єктів. Вивести:

a) список автомобілів заданої марки;
b) список автомобілів заданої моделі, які експлуатуються більше n років;
c) список автомобилів заданого року випуску, ціна яких більше вказаної.

Варіант 9. **Product**: id, Найменування, Виробник, Ціна, Термін зберігання, Кількість.

Скласти масив об'єктів. Вивести:

a) список товарів для заданого найменування;
b) список товарів для заданого найменування, ціна яких не перевищує задану;
c) список товарів, термін зберігання яких більше заданого.

Варіант 10. **Train**: Пункт призначення, Номер поїзду, Час відправки, Число місць (загальних, купе, плацкарт, люкс).

Скласти масив об'єктів. Вивести:

a) список поїздів, які прямують до заданого пункту призначення;
b) список поїздів, які прямують до заданого пункту призначення та відправляються після заданої години;
c) список поїздів, які відправляються до заданого пункту призначення та мають загальні місця.

Короткі теоретичні відомості

Описання власних типів

У мові програмування Java, програміст може визначати власні типи через створення відповідних класів. Опис класу найчастіше розміщується в окремому файлі, ім'я якого повинно співпадати з іменем класу.

Приклад описання класу

```
package lab3;
import java.util.*;

public class Lab3 {
    private int x; // змінна екземпляра класу
    private int y = 71; // змінна екземпляра класу
    public final int CURRENT_YEAR = 2007; // константа
    protected static int bonus; // змінна класу
    static String version = "Java SE 7"; // змінна класу
    protected Calendar now;

    public int method(int z) { // параметр метода
        z++;
        int a; // локальна змінна метода
        //a++; // помилка компіляції, значення не задано
        a = 4; // ініціалізація
        a++;
        now = Calendar.getInstance(); // ініціалізація
        return a + x + y + z;
    }
}
```

У розглянутому прикладі в якості змінних екземпляра класу, змінних класу та локальних змінних методу використані дані примітивних типів, що не є посиланнями на об'єкти (крім String). Дані можуть бути посиланнями, призначити яким реальні об'єкти можна за допомогою оператора **new**.

Конструктори

Конструктор — це метод, який автоматично викликається при створенні об'єкта класу і виконує дії з ініціалізації об'єкта. Конструктор має те ж ім'я, що й клас; викликається не по імені, а тільки разом з ключовим словом **new** при створенні екземпляра класу. конструктор не повертає значення, але може мати параметри і бути перевантаженим.

Приклад описання класу, що містить конструктори

```
package lab2;

public class Quest {
    private int id;
    private String text;
    // конструктор без параметрів (за замовчуванням)
    public Quest() {
       super(); /* якщо клас буде оголошений без конструктора, то
                   компілятор надасть його саме в такому вигляді */
    }
    // конструктор з параметрами
    public Quest(int idc, String txt) {
        super();
/* виклик конструктора суперкласу явним чином
   не є обов'язковим, компілятор вставить його автоматично */
        id = idc;
        text = txt;
    }
}
```

Об'єкт класу Quest може бути створений двома способами, які викликають один з конструкторів:

```
Quest a = new Quest();
//ініціалізація полів значеннями за замовчуванням
Quest b = new Quest(71, "Сколько бит занимает boolean?");
```

Оператор new викликає конструктор, тому в круглих дужках можуть стояти аргументи, що передаються конструктору. Якщо конструктор в класі не визначений, Java надає конструктор за замовчуванням без параметрів, який ініціалізує поля класу значеннями за замовчуванням, наприклад: 0, false, null. Якщо ж конструктор з параметрами визначений, то конструктор за замовчуванням стає недоступним і для його виклику необхідно явне оголошення такого конструктора.

В наступному прикладі оголошено клас Point з двома полями (атрибутами), конструктором і методами для ініціалізації та отримання значень атрибутів.

Приклад програми, що описує та використовує клас Point

```
package lab2;
public class Point {

    double x;
    double y;

    public Point(double xx, double yy) {
        x = xx;
        y = yy;
    }
    public double getX() {
        return x;
    }
    public double getY() {
        return y;
    }
}
```

```
package lab2;
public class LocateLogic {
    public double calculateDistance(Point t1, Point t2) {
        /* обчислення відстані */
        double dx = t1.getX() - t2.getX();
        double dy = t1.getY() - t2.getY();
        return Math.hypot(dx, dy);
    }
}
```

```
package lab2;
public class Runner {
    public static void main(String[] args) {
        // локальні змінні не є членами класу
        Point t1 = new Point(5, 10);
        Point t2 = new Point(2, 6);
        System.out.print("расстояние равно : " +
               new LocateLogic().calculateDistance(t1, t2));
    }
}
```

Рядки

Рядок у мові Java - це основний носій текстової інформації. Це не масив символів типу char, а об'єкт відповідного класу. Системна бібліотека Java містить класи String, StringBuilder і StringBuffer, що підтримують роботу з рядками і визначені в пакеті java.lang, що підключається автоматично. Ці класи оголошені як final, що означає неможливість створення власних породжених класів з властивостями рядків.

Клас String

Кожен рядок, створюваний за допомогою оператора **new** або за допомогою літерала (укладений в подвійні апострофи), є об'єктом класу String. Особливістю об'єкта класу String є те, що його значення не може бути змінено після створення об'єкту за допомогою якого методу класу, так як будь-яка зміна рядка приводить до створення нового об'єкта. При цьому посилання на об'єкт класу String можна змінити так, щоб воно вказувало на інший об'єкт і тим самим на інше значення.

Клас String підтримує кілька конструкторів, наприклад:

String(), String(String str), String(byte asciichar[]), String(char[] unicodechar), String(StringBuffer sbuf), String(StringBuilder sbuild) та інші.

Коли Java зустрічає літерал, укладений в подвійні лапки, автоматично створюється об'єкт типу String, на який можна встановити посилання. Таким чином, об'єкт класу String можна створити, присвоюючи посиланню на клас значення існуючого літерала, або за допомогою оператора **new** і конструктора, наприклад:

```
String s1 = "berkut.homelinux.com";

String s2 = new String("berkut.homelinux.com");
```

Клас String містить наступні (тут наведені основні, але не всі) методи для роботи з рядками:

`String concat(String s)` або “+” – злиття рядків;

`boolean` `equals(Object ob)` та `equalsIgnoreCase(String s)` – порівняння рядків з урахуванням та без урахування регістра відповідно;

`int` `compareTo(String s)` та `compareToIgnoreCase(String s)` – лексикографічне порівняння рядків з урахуванням та без урахування регістра. Метод виконує віднімання кодів символів рядка, що викликає метод та рядка, що передається до методу та повертає ціле значення. Метод повертає значення нуль у випадку, якщо equals() повертає значення true;

boolean `contentEquals(StringBuffer ob)` – порівняння рядка та вмісту об'єкта типу StringBuffer;

`String substring(`**int** `n,` **int** `m)` – отримання з рядку підрядка довжини m-n, починаючи з позиції n. Нумерація символів в рядку починається з нуля;

`String substring(`**int** `n)` – отримання з рядку підрядка, починаючи з позиції n;

int `length()` – визначення довжини рядка;

int `indexOf(`**char** `ch)` – визначення позиції символу у рядку;

static `String valueOf(значення)` – перетворення змінної примітивного типу у рядок;

`String toUpperCase()/toLowerCase()` – перетворення всіх символів рядка, який викликає метод у верхній/нижній регістр;

`String replace(`**char** `c1,` **char** `c2)` – заміна у рядку всіх входжень першого символу другим символом;

`String intern()` – заносить рядок в "пул" літералів та повертає його об'єктне посилання;

`String trim()` – вилучення всіх пропусків на початку та в кінці рядка;

char `charAt(`**int** `position)` – отримання символу із вказаної позиції (нумерація з нуля);

boolean `isEmpty()` – повертає true, якщо довжина рядка дорівнює 0;

`static String format(String format, Object... args), format(Locale l, String format, Object... args)` – генерує форматований рядок, отриманий з використанням формату, інтернаціоналізації та ін.;

`String[] split(String regex), split(String regex,` **int** `limit)` – пошук входження в рядок заданого регулярного виразу (розділителя) та поділ початкового рядка у відповідності з цим на масив рядків.

У всіх випадках виклику методів, яків потрібно змінити рядок, створюється новий об'єкт типу String.

Класи StringBuilder та StringBuffer

Класи StringBuilder та StringBuffer є “близнюками” та за призначенням наближені до класу String, але, на відмінність від останнього, вміст та розміри об'єктів класів StringBuilder та StringBuffer можна змінювати.

За допомогою відповідних методів та конструкторів об'єкти класів StringBuffer, StringBuilder та String можна перетворювати один до одного. Конструктор класу StringBuffer (так саме як і StringBuilder) може приймати в якості параметра об'єкт String або невід'ємний розмір буфера. Об'єкти цього класу можна перетворювати в об'єкт класу String методом toString() або за допомогою конструктора класу String.

Слід звернути увагу на такі методи:

`void` `setLength(`**`int`** `n)` – установка розміру буфера;

`void` `ensureCapacity(`**`int`** `minimum)` – установка гарантованого мінімального розміру буфера;

`int` `capacity()` – отримання поточного розміру буфера;

`StringBuffer append(параметри)` – додавання до вмісту об'єкта рядкового представлення аргументу, який може бути символом, значенням примітивного типу, масивом та рядком;

`StringBuffer insert(параметри)` – вставка символу, об'єкта або рядка в указану позицію;

`StringBuffer deleteCharAt(`**`int`** `index)` – вилучення символу;

`StringBuffer delete(`**`int`** `start,` **`int`** `end)` – вилучення підрядку;

`StringBuffer reverse()` – перестановка вмісту об'єкта у зворотному порядку.

В класі присутні також методи, аналогічні методам класу String, такі як replace(), substring(), charAt(), length(), getChars(), indexOf() та ін