Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра штучного інтелекту та програмного забезпечення

ЗВІТ
З ЛАБОРАТОРНОЇ РОБОТИ №1

дисципліна: «Крос-платформне програмування»

Виконав: студент групи КС23
Травченко Сергій Миколайович

Перевірив: доцент кафедри ШІтаПЗ
Споров Олександр Євгенович

Харків
2024

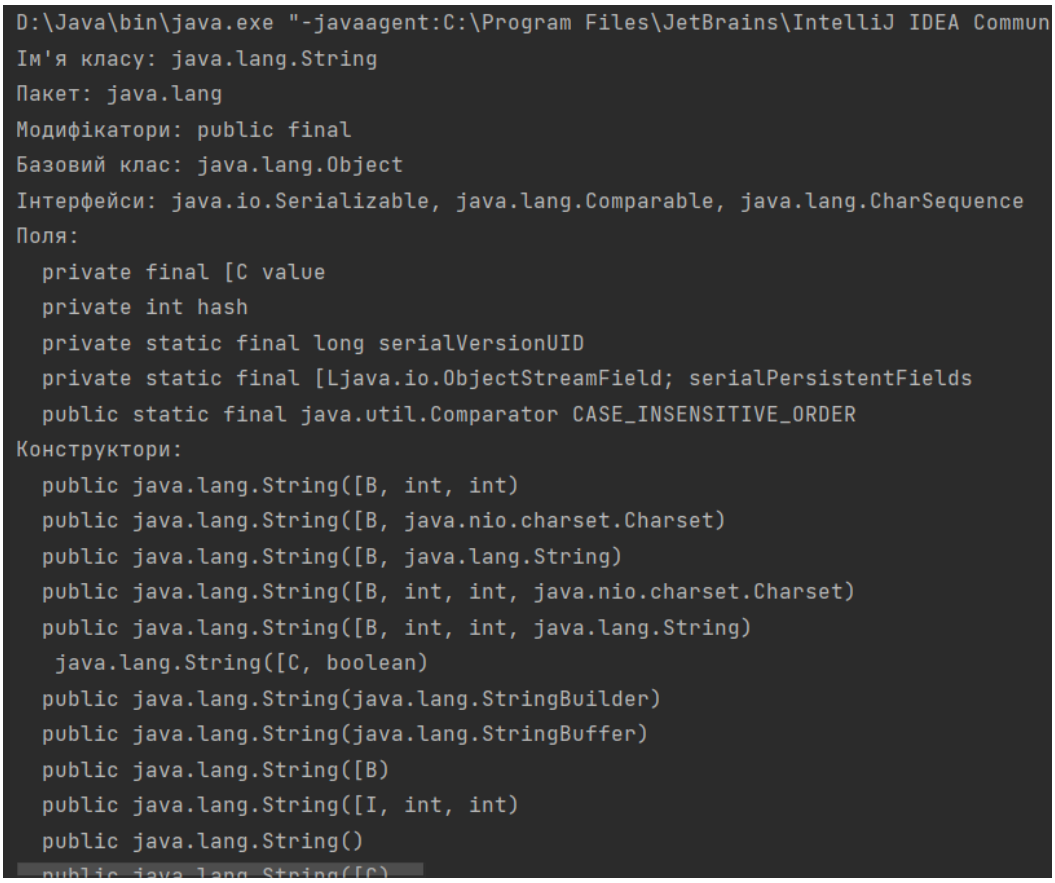
Завдання №1 Напишіть метод, що по повному імені типу, заданому у вигляді рядка, або по об'єкту типу Class, що створений попередньо, повертає рядок з його повним описом: ім'я пакета, в якому клас визначено, модифікатори та ім'я аналізованого класу, його базовий клас, список реалізованих інтерфейсів, а також список усіх полів, конструкторів та методів, оголошених у класі, та їх характеристики. При розв’язку задачі потрібно передбачити, що в програму для аналізу можуть бути передані як примітивні типи, так і типи-посилання (reference types): масиви, класи та інтерфейси.Для перевірки роботи напишіть консольну програму та програму з графічним інтерфейсом користувача.

Результати виконання завдання №1 наведено:
1. У лістингу 1 – вихідний код програми;
2. На малюнку 1.1 – результат виконання програми.

Лістинг 1. Вихідний код програми

```java
import java.lang.reflect.*;


public class Main {


    public static void main(String[] args) {
        // Приклад використання: аналізуємо клас String
        String className = "java.lang.String";
        analyzeClass(className);
    }


    public static void analyzeClass(String className) {
        try {
            Class<?> clazz = Class.forName(className);
            printClassInfo(clazz);
        } catch (ClassNotFoundException e) {
            System.out.println("Клас не знайдено: " + className);
        }
    }


    public static void analyzeClass(Class<?> clazz) { printClassInfo(clazz); }


    private static void printClassInfo(Class<?> clazz) {
        // Ім'я класу
        System.out.println("Ім'я класу: " + clazz.getName());

        // Пакет
        Package pkg = clazz.getPackage();
```

```java
    }

    1 usage
    private static String getFieldInfo(Field field) {
        return Modifier.toString(field.getModifiers()) + " " + field.getType().getNa
    }

    1 usage
    private static String getConstructorInfo(Constructor<?> constructor) {
        return Modifier.toString(constructor.getModifiers()) + " " + constructor.get
    }

    1 usage
    private static String getMethodInfo(Method method) {
        return Modifier.toString(method.getModifiers()) + " " + method.getReturnType
    }

    2 usages
    private static String getParametersInfo(Class<?>[] parameterTypes) {
        if (parameterTypes.length == 0) {
            return "()";
        }
        String[] parameterNames = getNames(parameterTypes);
        return "(" + String.join( delimiter: ", ", parameterNames) + ")";
    }

    2 usages
    private static String[] getNames(Class<?>[] classes) {
        String[] names = new String[classes.length];
        for (int i = 0; i < classes.length; i++) {
            names[i] = classes[i].getName();
```



Малюнок 1.1 – результат виконання програми

Завдання №1 Напишіть метод, що по повному імені типу, заданому у вигляді рядка, або по об'єкту типу Class, що створений попередньо, повертає рядок з його повним описом: ім'я пакета, в якому клас визначено, модифікатори та ім'я аналізованого класу, його базовий клас, список реалізованих інтерфейсів, а також список усіх полів, конструкторів та методів, оголошених у класі, та їх характеристики. При розв’язку задачі потрібно передбачити, що в програму для аналізу можуть бути передані як примітивні типи, так і типи-посилання (reference types): масиви, класи та інтерфейси.Для перевірки роботи напишіть консольну програму та програму з графічним інтерфейсом користувача.

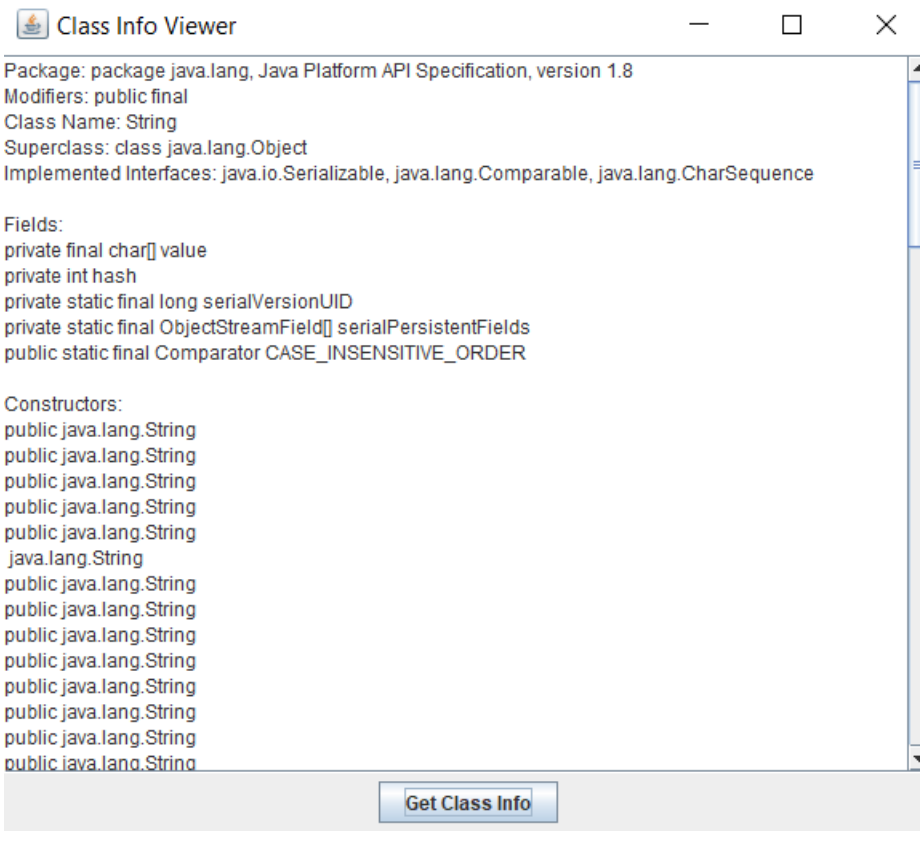
Результати виконання завдання №1 наведено:

1. У лістингу 1 – вихідний код програми;
2. На малюнку 1.2 – результат виконання програми.

Лістинг 1. Вихідний код програми

```java
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.lang.reflect.*;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;


public class ClassInfoViewer extends JFrame {

    private JTextArea textArea;


    public ClassInfoViewer() {
        setTitle("Class Info Viewer");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        textArea = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(textArea);
        add(scrollPane, BorderLayout.CENTER);

        JButton getInfoButton = new JButton("Get Class Info");

        getInfoButton.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) { showClassInfo(); }
```

```java
1 usage
private String getClassInfo(Class<?> clazz) {
    StringBuilder info = new StringBuilder();
    info.append("Package: ").append(clazz.getPackage()).append("\n");
    info.append("Modifiers: ").append(Modifier.toString(clazz.getModifiers())).appe
    info.append("Class Name: ").append(clazz.getSimpleName()).append("\n");
    info.append("Superclass: ").append(clazz.getSuperclass()).append("\n");

    Class<?>[] interfaces = clazz.getInterfaces();
    if (interfaces.length > 0) {
        info.append("Implemented Interfaces: ");
        for (Class<?> iface : interfaces) {
            info.append(iface.getName()).append(", ");
        }
        info.delete(info.length() - 2, info.length()); // Remove the trailing comma
        info.append("\n");
    }

    info.append("\nFields:\n");
    for (Field field : clazz.getDeclaredFields()) {
        info.append(Modifier.toString(field.getModifiers())).append(" ")
                .append(field.getType().getSimpleName()).append(" ")
                .append(field.getName()).append("\n");
    }

    info.append("\nConstructors:\n");
    for (Constructor<?> constructor : clazz.getDeclaredConstructors()) {
        info.append(Modifier.toString(constructor.getModifiers())).append(" ")
                .append(constructor.getName()).append("\n");
    }
```



Малюнок 1.2 – результат виконання програми

Завдання №2 Напишіть метод, що по отриманому об'єкту виводить його реальний тип та стан — список всіх полів, оголошених у класі, разом з їх значеннями, а також список оголошених у класі відкритих методів. Користувач може переглянути цей список, вибрати для виклику лише відкриті методи без параметрів, викликати їх на цьому об'єкті та переглянути результат виклику.

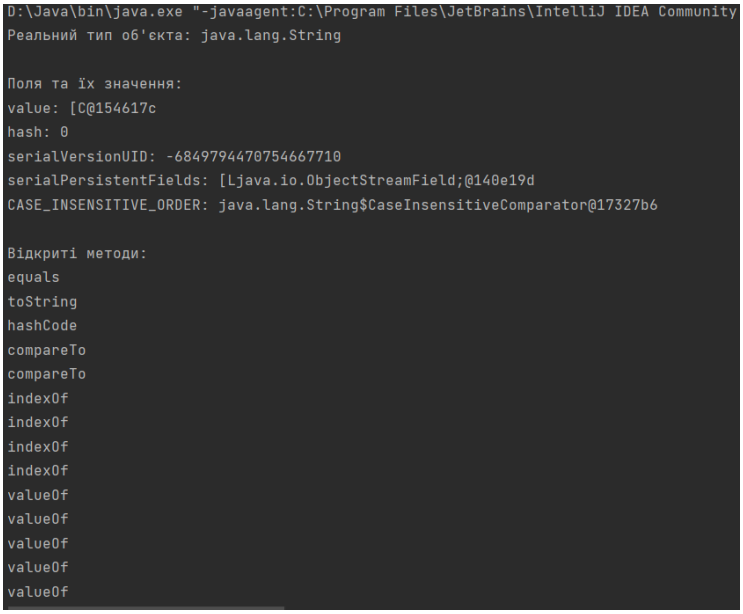
Результати виконання завдання №2 наведено:

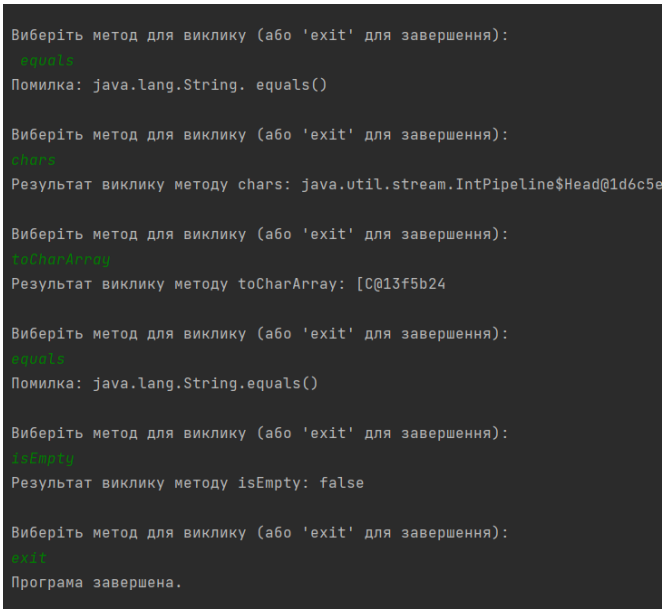
1. У лістингу 2 – вихідний код програми;
2. На малюнку 2.1 – результат виконання програми.

Лістинг 2. Вихідний код програми

```
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.Scanner;

no usages
public class Main {

    1 usage
    public static void inspectObject(Object obj) {
        // Отримати тип об'єкта
        Class<?> objClass = obj.getClass();
        System.out.println("Реальний тип об'єкта: " + objClass.getName());

        // Вивести список полів та їх значень
        System.out.println("\nПоля та їх значення:");
        Field[] fields = objClass.getDeclaredFields();
        for (Field field : fields) {
            field.setAccessible(true);
            try {
                System.out.println(field.getName() + ": " + field.get(obj));
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            }
        }

        // Вивести список відкритих методів
        System.out.println("\nВідкриті методи:");
        Method[] methods = objClass.getMethods();
        for (Method method : methods) {
            System.out.println(method.getName());
        }
    }
```

```
no usages
public static void main(String[] args) {
    // Створити об'єкт для інспекції
    // Приклад: можна створити власний клас і передати його об'єкт для інспекції
    // Object obj = new ВашКлас();

    // Наприклад, створимо об'єкт класу String для прикладу
    Object obj = new String( original: "Hello, World!");

    // Викликати метод для інспекції об'єкта
    inspectObject(obj);

    // Обробка виклику методу користувачем
    Scanner scanner = new Scanner(System.in);
    System.out.println("\nВиберіть метод для виклику (або 'exit' для завершення):");
    String methodName = scanner.nextLine();

    while (!methodName.equals("exit")) {
        try {
            Method selectedMethod = obj.getClass().getMethod(methodName);
            Object result = selectedMethod.invoke(obj);

            System.out.println("Результат виклику методу " + methodName + ": " + result);
        } catch (Exception e) {
            System.out.println("Помилка: " + e.getMessage());
        }

        System.out.println("\nВиберіть метод для виклику (або 'exit' для завершення):");
        methodName = scanner.nextLine();
    }

    System.out.println("Програма завершена.");
```





Малюнок 2.1 – результат виконання програми

Завдання №3 Напишіть метод, що отримує об'єкт, ім'я методу у вигляді рядка та список необхідних для виклику методу параметрів. Якщо цей метод може бути викликаний на заданому об'єкті, то вивести результат, інакше викинути виключення FunctionNotFoundException.

Результати виконання завдання №3 наведено:
1. У лістингу 3 – вихідний код програми;
2. На малюнку 3.1 – результат виконання програми.

Лістинг 3. Вихідний код програми

```java
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Arrays;
import java.util.List;

no usages
public class Main {

    no usages
    public static void main(String[] args) {
        // Приклад виклику методу
        SampleObject sampleObject = new SampleObject();
        String methodName = "sum";
        List<Object> parameters = Arrays.asList(2, 3);

        try {
            invokeMethod(sampleObject, methodName, parameters);
        } catch (FunctionNotFoundException e) {
            System.out.println(e.getMessage());
        }
    }

    1 usage
    public static void invokeMethod(Object object, String methodName, List<Object> paramet
        try {
            // Отримуємо клас об'єкта
            Class<?> objectClass = object.getClass();

            // Отримуємо метод за іменем і параметрами
            Method method = findMethod(objectClass, methodName, parameters);
```

```
D:\Java\bin\java.exe "-javaagent:C:\Program F
Результат виклику методу 'sum': 5

Process finished with exit code 0
```

Малюнок 3.1 – результат виконання програми

Завдання №4 Напишіть програму, що дозволяє створювати одновимірні масиви та матриці як примітивних, так і типів посилань (reference types), що будуть вказані під час роботи програми. Програма повинна вміти змінювати розміри масиву та матриці зі збереженням значень та перетворювати масиви та матриці на рядок.

Результати виконання завдання №4 наведено:
1. У лістингу 4 – вихідний код програми;
2. На малюнку 4.1 – результат виконання програми.

Лістинг 4. Вихідний код програми

```java
import java.util.Arrays;

no usages
public class Main {
    no usages
    public static void main(String[] args) {
        // Створення та робота з одновимірним масивом
        int[] primitiveArray = new int[]{1, 2, 3, 4, 5};
        printArray(primitiveArray);

        // Зміна розміру масиву
        primitiveArray = resizeArray(primitiveArray,  newSize: 8);
        printArray(primitiveArray);

        // Створення та робота з матрицею
        int[][] matrix = new int[][]{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        printMatrix(matrix);

        // Зміна розміру матриці
        matrix = resizeMatrix(matrix,  newRows: 4,  newCols: 3);
        printMatrix(matrix);

        // Перетворення масиву та матриці на рядок
        String arrayString = arrayToString(primitiveArray);
        String matrixString = matrixToString(matrix);

        System.out.println("Масив у вигляді рядка: " + arrayString);
        System.out.println("Матриця у вигляді рядка: " + matrixString);
    }

    // Друк масиву
```

```java
        int[] newArray = Arrays.copyOf(array, newSize);
        return newArray;
    }

    // Друк матриці
    2 usages
    private static void printMatrix(int[][] matrix) {
        System.out.println("Матриця:");
        for (int[] row : matrix) {
            System.out.println(Arrays.toString(row));
        }
    }

    // Зміна розміру матриці зі збереженням значень
    1 usage
    private static int[][] resizeMatrix(int[][] matrix, int newRows, int
        int[][] newMatrix = new int[newRows][newCols];
        for (int i = 0; i < Math.min(matrix.length, newRows); i++) {
            newMatrix[i] = Arrays.copyOf(matrix[i], newCols);
        }
        return newMatrix;
    }

    // Перетворення масиву на рядок
    1 usage
    private static String arrayToString(int[] array) {
        return Arrays.toString(array);
    }

    // Перетворення матриці на рядок
    1 usage
    private static String matrixToString(int[][] matrix) {
```

```
D:\Java\bin\java.exe "-javaagent:C:\Program Files\Jet
Масив: [1, 2, 3, 4, 5]
Масив: [1, 2, 3, 4, 5, 0, 0, 0]
Матриця:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Матриця:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
[0, 0, 0]
Масив у вигляді рядка: [1, 2, 3, 4, 5, 0, 0, 0]
Матриця у вигляді рядка: [1, 2, 3]
[4, 5, 6]
[7, 8, 9]
[0, 0, 0]


Process finished with exit code 0
```

Малюнок 4.1 – результат виконання програми

Завдання №5 Напишіть програму, що демонструє особливості застосування «універсальних» динамічних об'єктів проксі для профілювання методу (виводить на екран час обчислення методу) та для трасування методу (виводить на екран ім'я, параметри методу та обчислене значення).

Результати виконання завдання №5 наведено:
1. У лістингу 5 – вихідний код програми;
2. На малюнку 5.1 – результат виконання програми.

Лістинг 5. Вихідний код програми

```
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        long startTime = System.nanoTime();
        Object result = method.invoke(target, args);
        long endTime = System.nanoTime();

        System.out.println("Method " + method.getName() + " took " + (endTime - startTime

        return result;
    }
}

1 usage
class TracingHandler implements InvocationHandler {
    2 usages
    private final Object target;

    1 usage
    public TracingHandler(Object target) {
        this.target = target;
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        System.out.println("Method " + method.getName() + " called with parameters: " + a

        Object result = method.invoke(target, args);

        System.out.println("Method " + method.getName() + " returned: " + result);
```

```
        return result.toString();
    }
}

no usages
public class Main {
    no usages
    public static void main(String[] args) {
        // Profiling
        MyInterface profilingProxy = (MyInterface) Proxy.newProxyInstance(
                MyInterface.class.getClassLoader(),
                new Class[]{MyInterface.class},
                new ProfilingHandler(new MyImplementation())
        );

        System.out.println("Profiling Proxy Result: " + profilingProxy.calculate( a: 3,  b: 7));

        // Tracing
        MyInterface tracingProxy = (MyInterface) Proxy.newProxyInstance(
                MyInterface.class.getClassLoader(),
                new Class[]{MyInterface.class},
                new TracingHandler(new MyImplementation())
        );

        System.out.println("Tracing Proxy Result: " + tracingProxy.calculate( a: 5,  b: 8));
    }
}
```

```
D:\Java\bin\java.exe "-javaagent:C:\Program Files\Jet
Method calculate took 37900 nanoseconds to execute.
Profiling Proxy Result: 10
Method calculate called with parameters: [5, 8]
Method calculate returned: 13
Tracing Proxy Result: 13

Process finished with exit code 0
```

Малюнок 5.1 – результат виконання програми

Завдання №5 Напишіть програму, що дозволяє переглянути список конструкторів заданого під час роботи програми класу, вибрати потрібний конструктор та створити об'єкт цього класу, потім переглянути список усіх методів класу та вибрати потрібний метод. На кожному етапі програма повинна аналізувати та виводити на екран стан об'єкта (імена та значення його полів)

Результати виконання завдання №5 наведено:

1. На малюнку 6.1 – результат виконання програми.

```
D:\Java\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Co
Список конструкторів:
public java.lang.String(byte[],int,int)
public java.lang.String(byte[],java.nio.charset.Charset)
public java.lang.String(byte[],java.lang.String) throws java.io.UnsupportedE
public java.lang.String(byte[],int,int,java.nio.charset.Charset)
public java.lang.String(byte[],int,int,java.lang.String) throws java.io.Unsu
public java.lang.String(java.lang.StringBuilder)
public java.lang.String(java.lang.StringBuffer)
public java.lang.String(byte[])
public java.lang.String(int[],int,int)
public java.lang.String()
public java.lang.String(char[])
public java.lang.String(java.lang.String)
public java.lang.String(char[],int,int)
public java.lang.String(byte[],int)
public java.lang.String(byte[],int,int,int)

Стан об'єкта:
value: [C@154617c
hash: 0
serialVersionUID: -6849794470754667710
serialPersistentFields: [Ljava.io.ObjectStreamField;@140e19d
CASE_INSENSITIVE_ORDER: java.lang.String$CaseInsensitiveComparator@17327b6

Список методів:
public boolean java.lang.String.equals(java.lang.Object)
public java.lang.String java.lang.String.toString()
```

```
public java.lang.String java.lang.String.replace(java.lang.CharSequenc
public java.lang.String java.lang.String.replaceAll(java.lang.String,j
public java.lang.String java.lang.String.replaceFirst(java.lang.String
public java.lang.String[] java.lang.String.split(java.lang.String)
public java.lang.String[] java.lang.String.split(java.lang.String,int)
public boolean java.lang.String.startsWith(java.lang.String,int)
public boolean java.lang.String.startsWith(java.lang.String)
public java.lang.CharSequence java.lang.String.subSequence(int,int)
public java.lang.String java.lang.String.substring(int)
public java.lang.String java.lang.String.substring(int,int)
public char[] java.lang.String.toCharArray()
public java.lang.String java.lang.String.toLowerCase(java.util.Locale)
public java.lang.String java.lang.String.toLowerCase()
public java.lang.String java.lang.String.toUpperCase()
public java.lang.String java.lang.String.toUpperCase(java.util.Locale)
public java.lang.String java.lang.String.trim()
public final void java.lang.Object.wait() throws java.lang.Interrupted
public final void java.lang.Object.wait(long,int) throws java.lang.Int
public final native void java.lang.Object.wait(long) throws java.lang.
public final native java.lang.Class java.lang.Object.getClass()
public final native void java.lang.Object.notify()
public final native void java.lang.Object.notifyAll()
public default java.util.stream.IntStream java.lang.CharSequence.chars
public default java.util.stream.IntStream java.lang.CharSequence.codeP

Результат виклику методу length: 5

Process finished with exit code 0
```

Малюнок 6.1 – результат виконання програми