Аналіз покращень в Java 19 у порівнянні з Java 17

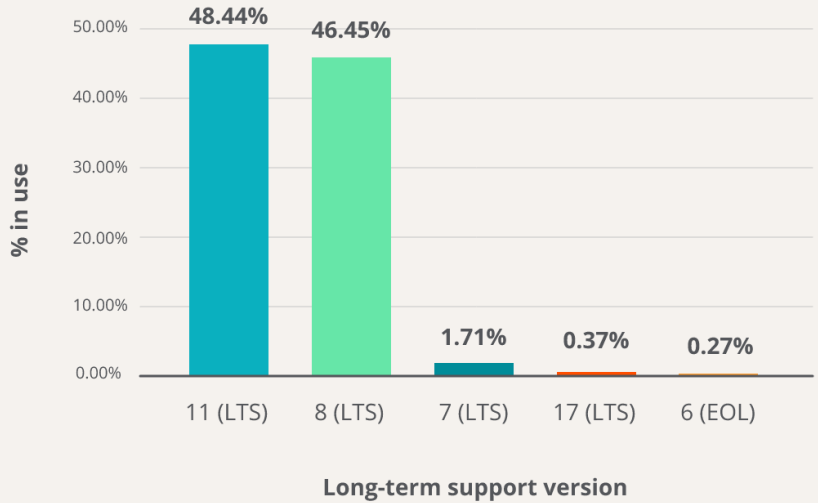
Курсова робота студента КН-3 Новака В. І.
Керівник курсової роботи Яремко С. А.

Вступ

Об’єкт дослідження: версії Java 17 та Java 19.
Мета дослідження: порівняння оновлень в цих версіях, та аналіз позитивних змін в версії Java 19 порівняно з саме Java 17.
Актуальність роботи полягає в дослідженні версій мови програмування Java, які вийшли рік-два тому, та аналізі оновлень в Java 19, які відповідають сучасним тенденціям програмування.

Java 17

Версія Java 17 була випущена 14 вересня 2021 року як LTS-версія.



Використання LTS-версій мови Java на початку 2022 року

Java 19

Версія Java 19 була випущена 20 вересня 2022.

- Віртуальні потоки
- Оновлення паттерн матчингу для switch
- Паттерни записів
- Структурована паралельність
- Linux/RISC-V Port
- Оновлення Vector API
- Foreign Function & Memory API

Порівняння покращень в Java 19 з Java 17

Для реалізації порівнянь було створено невеликий консольний додаток, який імітує роботу онлайн-магазину.

Клас OnlineShop	
	Списки catalog, customers і orders
	Методи для здійснення покупки товару
	Метод main для реалізації оновлень Java 19

Використання оновлень в Java 19

Щоби мати доступ до preview JEP-ів (та в нашому випадку до інкубованого JEP-а про структуровану паралельність, потрібно запустити файл .java в консолі таким чином:

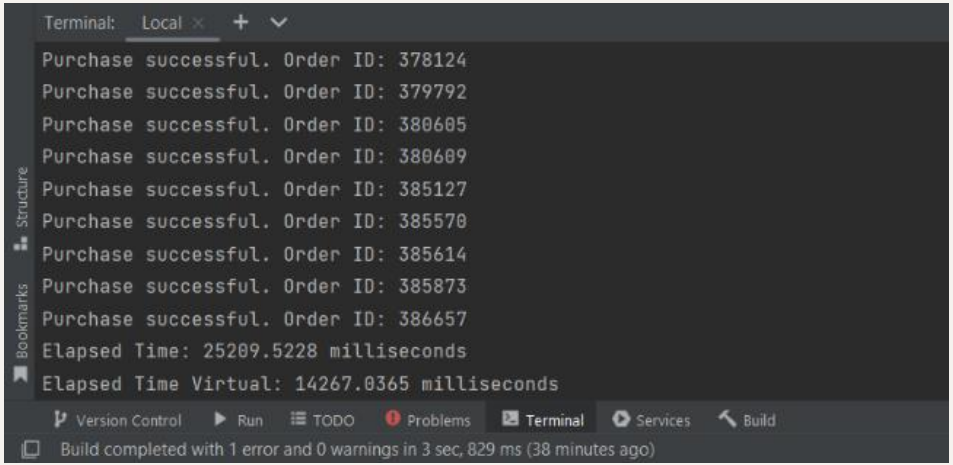
```
$ javac --release 19 --enable-preview --add-modules jdk.incubator.concurrent OnlineShop.java
$ java --enable-preview --add-modules jdk.incubator.concurrent OnlineShop
```

Порівняння віртуальних потоків з платформними

```
Runnable task = () -> {
        performPurchases(shop, 2);
 };
executor = Executors.newVirtualThreadPerTaskExecutor();
for (int i = 0; i < 100000; i++) {
    executor.submit(task);
}
executor.close();
```

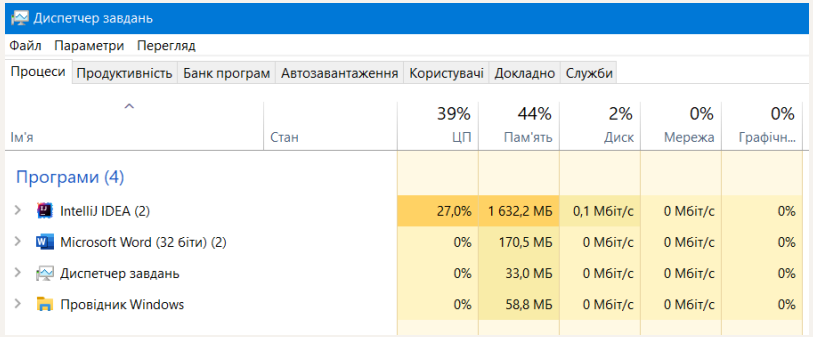
```
Runnable task = () -> {
        performPurchases(shop, 2);
 };
ExecutorService executor = Executors. newCachedThreadPool();
for (int i = 0; i < 100000; i++) {
    executor.submit(task);
}
executor.close();
```

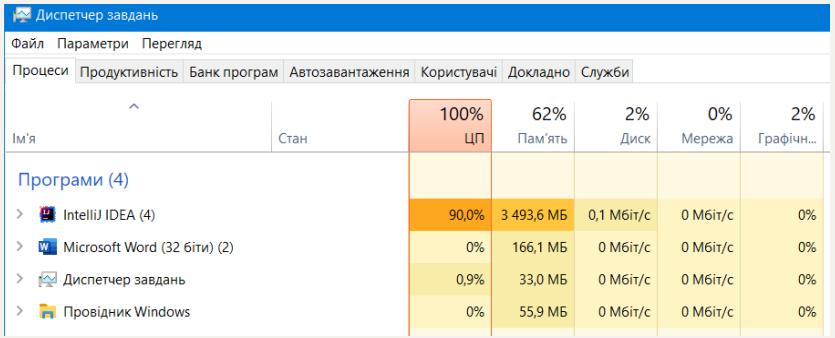
Порівняння віртуальних потоків з платформними



*Скріншот консолі **Intellij Idea** при виконанні програми*

Порівняння віртуальних потоків з платформними





Скріншоти диспетчера завдань при запуску віртуальних та платформних потоків відповідно

Реалізація структурованого паралелізму в Java 19

```
try (var scope = new StructuredTaskScope.ShutdownOnFailure()){
        Future<String> productName = scope.fork(() -> shop.getProductName(id));
        Future<Double> productPrice = scope.fork(() -> shop.getProductPrice(id));
        scope.join();
        scope.throwIfFailed();
        System.out.println("ID: " + id + " | Name: " + productName.resultNow() + " | Price: $" +
productPrice.resultNow()); }
```

Реалізація структурованого паралелізму в Java 17

Оскільки в версії Java 17 немає реалізації API для роботи зі структурованим паралелізмом, для цього можна використовувати вищезгаданий **java.util.concurrent.ExecutorService** API.

Основною проблемою такого підходу є велика ймовірність **витоку потоків** при помилці роботи.

Реалізація патернів для записів (records) в Java 19

```
switch (o) {
        case Product(String name, double price) when price > 100 ->
            System.out.println("Premium Product: " + name + ", " + price + "hrn.");
        case Product(String name, double price) ->
            System.out.println("Product: " + name + ", " + price + "hrn.");
        case Customer(String username, String password, String email) ->
            System.out.println("Our customer: " + username);
        case Order(int orderId, Customer customer, List<Product> products, double totalAmount,
String status) ->
            System.out.println("Order " + orderId + " made by our customer " + customer.username()
+ ", amount: " + totalAmount);
        default -> System.out.println("The object isn't an instance of OnlineShop.");
    }
```

Заміна патернів для записів (records) в Java 17

У версії Java 17 потрібно би було явно переводити тип змінної, та вже потім працювати з нею як з патерном. Це виглядало би так:

```
if (o instanceof String) {
    String s = (String)o;
    // код
}
```

Інші покращення в Java 19

- **Foreign Function & Memory API** (для оптимізованої роботи з ресурсами з інших мов програмування, як-от C, C++, тощо);
- **Vector API** (Fourth Incubator) (додано нові корисні функції для роботи з великими об’єктами Vector).

Висновки

01 Ми провели порівняння оновлень в Java 19 з Java 17

02 Ці покращення полегшили роботу в Java та оптимізували її

03 Важливими покращеннями можна виділити реалізацію віртуальних потоків та структурованої паралельності

04 Перспективами розвитку у даній темі є покращення віртуальних потоків, pattern matching, та створення нового функціоналу

Дякую за увагу!