TUGAS PEMROGRAMAN BERORIENTASI OBJEK

Dosen : Alun Sujjada, S. Kom, M.T

Nama	: Riza Rumayanti Dewi
NIM	: 20200040161
Kelas	: TI20B
Jurusan	: Teknik Informatika

September 25, 2021

Pertanyaan : Buatlah resume(ulasan) informasi tentang versi terakhir JAVA saat ini ! Jelaskan fitur-fitur yang ditambahkan beserta kelebihan-kelebihannya

Jawaban:

Versi terakhir JAVA saat ini yaitu Java Platform (JDK) 17 yang telah dirilis pada 14 September 2021 dengan 10 fitur baru + 2 penghapusan fitur + 2 penghentian fitur.

Java 17 dirilis sebagai versi Java LTS berikutnya. Dukungan jangka panjang (LTS) adalah kebijakan manajemen siklus hidup produk di mana rilis stabil perangkat lunak komputer dipertahankan untuk jangka waktu yang lebih lama daripada edisi standar. Istilah ini biasanya dicadangkan untuk perangkat lunak sumber terbuka, yang menggambarkan edisi perangkat lunak yang didukung selama berbulan-bulan atau bertahun-tahun lebih lama dari edisi standar perangkat lunak. Ini menstandarisasi kelas yang disegel, mengembalikan semantik floating-point yang selalu ketat, dan menunjukkan pencocokan pola untuk pernyataan switch.

Fitur-fitur yang ditambahkan beserta kelebihannya:

1) Filter Deserialisasi Khusus Konteks

Izinkan aplikasi untuk mengonfigurasi filter deserialisasi khusus konteks dan yang dipilih secara dinamis menggunakan pabrik filter seluruh JVM yang digunakan untuk memilih filter untuk setiap operasi deserialisasi.

Motivasi:

Deserialisasi data yang tidak tepercaya adalah operasi yang secara intrinsik berisiko, karena konten aliran data yang masuk diperoleh dalam banyak kasus melalui klien yang tidak dikenal atau tidak diautentikasi.

Kunci untuk mencegah serangan serialisasi adalah dengan melarang instance kelas arbitrer dari deserialized, yang mencegah eksekusi metode mereka baik secara langsung maupun tidak langsung.

Penyerang dapat menjalankan kode dari kelas mana pun dengan niat buruk dengan membangun aliran secara hati-hati. Integritas objek aplikasi, objek perpustakaan, dan runtime Java dapat dikompromikan jika konstruksi objek melibatkan efek samping yang mengubah status atau memicu operasi lain.

2) API Vektor (Inkubator Kedua)

Memperkenalkan API vektor platform-agnostik yang pertama kali diintegrasikan ke dalam JDK 16 dan akan diinkubasi lagi di JDK 17 untuk mengekspresikan komputasi vektor yang dikompilasi secara andal ke instruksi vektor optimal pada arsitektur CPU yang didukung saat runtime, mengungguli komputasi skalar yang setara. API vektor di JDK 17 telah ditingkatkan untuk kinerja dan implementasi, termasuk peningkatan untuk menerjemahkan vektor byte ke dan dari array boolean.

Sasaran:

API yang jelas dan ringkas : API harus mampu mengekspresikan berbagai macam komputasi vektor dengan cara yang jelas dan ringkas.

Platform agnostic : API harus independen dari arsitektur CPU, memungkinkan implementasi pada berbagai arsitektur yang mendukung instruksi vektor.

Kompilasi dan kinerja runtime yang andal pada arsitektur x64 dan AArch64

Degradasi anggun : Jika komputasi vektor tidak dapat dikompilasi secara efisien ke instruksi vektor, peringatan dapat dikeluarkan.

3) Fungsi Asing & API Memori (Inkubator)

Memperkenalkan API yang memungkinkan program Java untuk memanggil pustaka asli dan memproses data asli tanpa risiko JNI, dengan menjalankan fungsi asing secara efisien (yaitu, kode di luar JVM) dan mengakses memori asing dengan aman (yaitu memori yang tidak ditangani oleh JVM). Dalam proposal JEP ini merupakan evolusi dari dua API inkubasi sebelumnya: API Akses Memori Asing dan API Penaut Asing. yang sebelumnya ditargetkan di Java 14, 15 dan Java 16

Sasaran:

Kemudahan penggunaan : Ganti model pengembangan Java murni yang unggul dengan Java Native Interface (JNI).

Performa : Performa yang mirip dengan API yang ada seperti JNI atau sun.misc.Unsafe, jika tidak lebih baik. Pertunjukan.

Umum : Menyediakan cara untuk bekerja pada berbagai jenis memori eksternal (misalnya, memori asli, memori persisten dan memori heap) dan untuk mengakomodasi platform lain dari waktu ke waktu (misalnya, x86 32-bit) dan fungsi asing yang ditulis dalam bahasa selain C (misalnya, C++, FORTAN).

Keamanan : Nonaktifkan operasi default yang tidak aman hanya jika pengembang aplikasi atau pengguna akhir telah secara tegas memilih ikut serta.

4) Menghentikan Penghapusan Manajer Keamanan

Sejak Java 1.0, telah ada Manajer Keamanan. Namun, telah jarang digunakan selama bertahun-tahun. Untuk memajukan Java, Manajer Keamanan telah tidak digunakan lagi di Java 17 dan akan dihapus di versi mendatang, bersama dengan Applet API lama (JEP 398).

Sasaran:

TUGAS PEMROGRAMAN BERORIENTASI OBJEK

Dosen : Alun Sujjada, S. Kom, M.T

Nama	: Riza Rumayanti Dewi
NIM	: 20200040161
Kelas	: TI20B
Jurusan	: Teknik Informatika

September 25, 2021

Siapkan pengembang untuk penghapusan Manajer Keamanan di versi Java yang akan datang. Jika program Java pengguna bergantung pada Manajer Keamanan, berikan peringatan.

Evaluasi apakah API atau mekanisme baru diperlukan untuk memperbaiki kasus penggunaan unik dan terbatas di mana Manajer Keamanan telah digunakan, seperti memblokir System::exit.

5) Hapus Kompilator AOT dan JIT Eksperimental

Hapus kompiler sebelumnya (AOT) dan just-in-time (JIT) berbasis Java eksperimental karena penggunaan yang terbatas dan upaya yang diperlukan untuk mempertahankannya sangat signifikan. Meskipun mempertahankan antarmuka kompiler JVM tingkat Java eksperimental (JVMCI) sehingga pengembang dapat terus menggunakan versi kompiler yang dibuat secara eksternal untuk kompilasi JIT dengan menggunakan kompiler Graal (GraalVM).

Motivasi:

Sebagai fitur eksperimental JDK 9 telah terintegrasi dengan kompilasi sebelumnya (alat jaotc). Untuk kompilasi AOT, jaotc menggunakan compiler Graal yang ditulis Java. Karena karakteristik eksperimental ini belum digunakan, dan ada upaya yang cukup besar untuk mempertahankan dan meningkatkannya. Build JDK 16 yang dirilis oleh Oracle tidak memiliki fungsi ini, juga tidak ada yang mengeluh.

6) Kelas Tertutup

Kelas Tertutup diusulkan dan disampaikan sebagai fitur pratinjau kedua di JDK 16. Tingkatkan bahasa pemrograman Java dengan kelas dan antarmuka tertutup. Kelas dan antarmuka yang disegel membatasi kelas atau antarmuka mana yang dapat diperluas atau diimplementasikan.

Sasaran:

Aktifkan pembuat kelas atau antarmuka untuk mengontrol kode mana yang akan mengimplementasikannya.

Berikan pendekatan yang lebih deklaratif daripada pengubah akses untuk membatasi penggunaan superclass.

Mendukung arah pencocokan pola masa depan dengan dasar untuk memeriksa tren secara menyeluruh.

Perbaikan:

Dalam JLS, jelaskan definisi kata kunci kontekstual, yang menggantikan definisi sebelumnya tentang pengidentifikasi terbatas dan kata kunci terbatas.

Perkenalkan urutan karakter yang disegel dan tidak disegel dan izinkan mereka sebagai kata kunci kontekstual.

Seperti kelas anonim dan ekspresi lambda, kelas lokal tidak boleh menjadi subkelas kelas tersegel saat memutuskan subkelas yang diizinkan secara implisit dari kelas tertutup atau antarmuka tertutup.

Sebuah kelas dapat disegel dengan menggunakan pengubah yang disegel - izin untuk deklarasinya. Setelah klausa perluasan dan implementasi apa pun, klausa izin menentukan kelas yang diizinkan/diizinkan untuk memperluas kelas tersegel.

Contoh: di bawah ini menunjukkan bagaimana kelas Hewan menentukan tiga subkelas yang diizinkan:

```
package com.techgeeknext.example.species;

public sealed class Animal
permits cat, cow, fish {...}
```

Catatan: Kelas-kelas yang ditentukan oleh izin harus ditempatkan dekat dengan superclass, baik dalam modul yang sama atau dalam paket yang sama.

```
package com.techgeeknext.example.species;

public sealed class Animal
com.techgeeknext.example.species.cat,
com.techgeeknext.example.species.cow,
com.techgeeknext.example.species.fish
{...}
```

7) Hapus Aktivasi RMI

Mekanisme Aktivasi Remote Method Invocation (RMI) akan dihapus, tetapi RMI lainnya harus dipertahankan. Mekanisme Aktivasi RMI telah menjadi redundan dan tidak lagi digunakan. JEP 385 di Java SE 15 tidak digunakan lagi dan merekomendasikannya untuk dihapus.

8) Pencocokan Pola untuk sakelar (Pratinjau)

Pencocokan pola untuk sakelar memperluas bahasa pola Java untuk memungkinkan ekspresi dan pernyataan sakelar diverifikasi terhadap berbagai pola, masing-masing dengan tindakan yang berbeda. Hal ini memungkinkan untuk mengekspresikan kueri berorientasi data yang kompleks dengan cara yang sederhana dan aman. Operator instanceof di JDK 16 telah diperluas untuk menerima pola tipe dan melakukan pencocokan pola. Ekstensi sederhana yang diusulkan menyederhanakan idiom instanceof-and-cast yang terkenal.

Sasaran:

Membiarkan pola muncul dalam label kasus meningkatkan ekspresi dan penerapan frasa dan pernyataan beralih.

Biarkan permusuhan nol dari titik balik sejarah dilonggarkan jika diperlukan.

TUGAS PEMROGRAMAN BERORIENTASI OBJEK

Dosen : Alun Sujjada, S. Kom, M.T

Nama	: Riza Rumayanti Dewi
NIM	: 20200040161
Kelas	: TI20B
Jurusan	: Teknik Informatika

September 25, 2021

Dua pola baru akan diperkenalkan:

Pola yang dijaga: untuk memperbaiki logika pencocokan pola menggunakan ekspresi boolean arbitrer.

Pola dalam kurung: untuk menjernihkan ambiguitas penguraian.

Pastikan bahwa semua ekspresi dan pernyataan sakelar yang ada dikompilasi dengan semantik yang identik dan menjalankannya tanpa modifikasi apa pun.

9) Enkapsulasi Internal JDK dengan Kuat

Enkapsulasi semua elemen internal JDK dengan kuat, kecuali untuk API internal penting seperti sun.misc.Unsafe. Tidak mungkin lagi melonggarkan enkapsulasi ketat bagian internal dengan satu opsi baris perintah, seperti dari JDK 9 hingga JDK 16.

10) Menghentikan Applet API untuk Penghapusan

Applet API secara efektif tidak berguna, karena semua vendor browser web telah menghapus atau mengungkapkan rencana untuk menghentikan dukungan untuk plug-in browser Java . Sementara API Applet tidak digunakan lagi di Java 9, itu tidak dihapus sebelumnya.

11) Port macOS/AArch64

Port JDK ke arsitektur baru macOS/AArch64 mengharapkan permintaan di masa mendatang

Motivasi:

Keputusan Apple untuk pindah dari x64 ke AArch64 di komputer Macintosh-nya. Untuk Linux, versi Java AArch64 sudah tersedia, dan pengembangan pada port Windows saat ini sedang berlangsung.

Karena perbedaan dalam konvensi tingkat rendah seperti antarmuka biner program dan kumpulan register prosesor yang dicadangkan, pengembang Java berencana untuk menggunakan kembali kode AArch64 yang ada dari port ini dengan menggunakan kompilasi bersyarat, seperti standar di port JDK.

Perubahan untuk MacOS/AArch64 berpotensi memecah port Linux/AArch64, Windows/AArch64, dan MacOS/x64 saat ini, meskipun kemungkinan ini dapat dikurangi dengan pengujian praintegrasi.

12) Pipa Rendering macOS baru

Perlunya jalur rendering Java 2D baru untuk macOS menggunakan kerangka kerja Apple Metal baru. Seperti hari ini, Java 2D sepenuhnya bergantung pada OpenGL. Meskipun Apple tidak lagi

menggunakan pustaka rendering OpenGL di macOS 10.14, tetapi kerangka kerja Metal menggantikan pustaka rendering OpenGL.

Sasaran:

Menyediakan pipeline rendering yang berfungsi sepenuhnya untuk Java 2D API berbasis kerangka kerja Logam macOS.

Dalam aplikasi dan tolok ukur dunia nyata tertentu, berikan kinerja yang sebanding atau lebih baik daripada pipeline OpenGL.

Berdampingan dengan pipeline OpenGL hingga dihentikan.

13) Pembangkit Angka Acak Pseudo yang Ditingkatkan

Memperkenalkan antarmuka dan implementasi baru untuk generator nomor pseudorandom (PRNG), yang mencakup PRNG yang dapat dilompati dan kelas baru algoritma PRNG yang dapat dipisah (LXM). Memotivasi rencana tersebut adalah fokus pada beberapa bidang untuk perbaikan di bidang pembuatan nomor pseudorandom di Jawa. RandomGenerator, antarmuka modern, akan memiliki API seragam untuk semua PRNG saat ini dan baru, serta empat antarmuka RandomGenerator khusus.

Sasaran:

Buatlah lebih mudah untuk menggunakan algoritma PRNG yang berbeda dalam aplikasi yang berbeda.

Dengan menyediakan aliran objek PRNG, Anda dapat mendukung pemrograman berbasis aliran dengan lebih baik.

Hapus kode yang berlebihan dari grup PRNG saat ini.

Pertahankan tindakan kelas java.util.Random sejauh mungkin.

14) Kembalikan Semantik Titik Mengambang Selalu Ketat

Daripada memiliki semantik floating-point yang parah (strictfp) dan semantik floating-point default yang sangat berbeda, buat operasi floating-point seragam ketat. Ini akan mengembalikan bahasa dan mesin virtual ke semantik titik-mengambang aslinya, seperti sebelum pengenalan mode titik-mengambang yang ketat dan default di Java SE 1.2.

Sasaran:

Terus tingkatkan keamanan dan pemeliharaan JDK, yang merupakan salah satu tujuan utama Project Jigsaw.

Dorong pengembang untuk beralih dari bagian internal dan menuju API standar sehingga mereka dan penggunanya dapat meningkatkan ke rilis Java mendatang dengan mudah.