Ministry of SMEs and Startups × KOSME Korea SMEs and Startups Agency × LIKELION

JDK, GC, Serialization

Nguyễn Anh Tuấn

KTECH COLLEGE

Nội dung bài giảng

1. Java Development Kit (JDK)
2. Garbage Collection (GC)
3. Serialization

Java Development Kit

Java Development Kit (JDK)

- JDK là bộ công cụ phát triển Java, cung cấp mọi thứ cần thiết để phát triển, biên dịch, và chạy các ứng dụng Java. JDK bao gồm các công cụ như:
  - **Javac**: Trình biên dịch Java, chuyển đổi mã nguồn Java (.java) thành mã bytecode (.class).
  - **JRE**: Môi trường thực thi Java, bao gồm các thư viện cốt lõi và JVM.
  - **JVM**: Máy ảo thực thi các mã bytecode do Javac tạo ra.
  - **Javadoc**: Công cụ tạo tài liệu API từ mã nguồn Java.
  - **JConsole**: Công cụ giám sát hiệu suất ứng dụng Java.
  - …

Javac

- Javac là trình biên dịch của Java, được sử dụng để chuyển đổi mã nguồn Java thành mã bytecode. Các bước thực hiện của Javac bao gồm:
  - **Đọc mã nguồn (Lexical Analysis)**: Đọc mã nguồn từ các tệp .java và chuyển đổi thành một loạt các **token**.
  - **Phân tích cú pháp (Parsing)**: Phân tích danh sách các token để xây dựng cây cú pháp Abstract Syntax Tree (AST), biểu diễn các syntax có cấu trúc hơn.
  - **Phân tích ngữ nghĩa (Semantic Analysis)**: Kiểm tra kiểu dữ liệu, xác thực các biến được khai báo trước khi sử dụng, kiểm tra các quy tắc phạm vi, xác minh các phương thức được gọi đúng cách.
  - **Tạo mã (Code Generation)**: Chuyển đổi cây cú pháp có ngữ nghĩa thành mã bytecode.
  - **Tối ưu hóa mã (Optimization)**: Bao gồm loại bỏ mã không cần thiết, cải thiện việc sử dụng bộ nhớ, sắp xếp lại các lệnh để tối ưu hóa hiệu suất
- Cú pháp của Javac:
  - javac [options] [sourcefiles] [@argfiles]
  - Ví dụ: javac helloWorld.java
  - Sử dụng **javac -help** để hiển thị trợ giúp về cú pháp và các tuỳ chọn của javac.

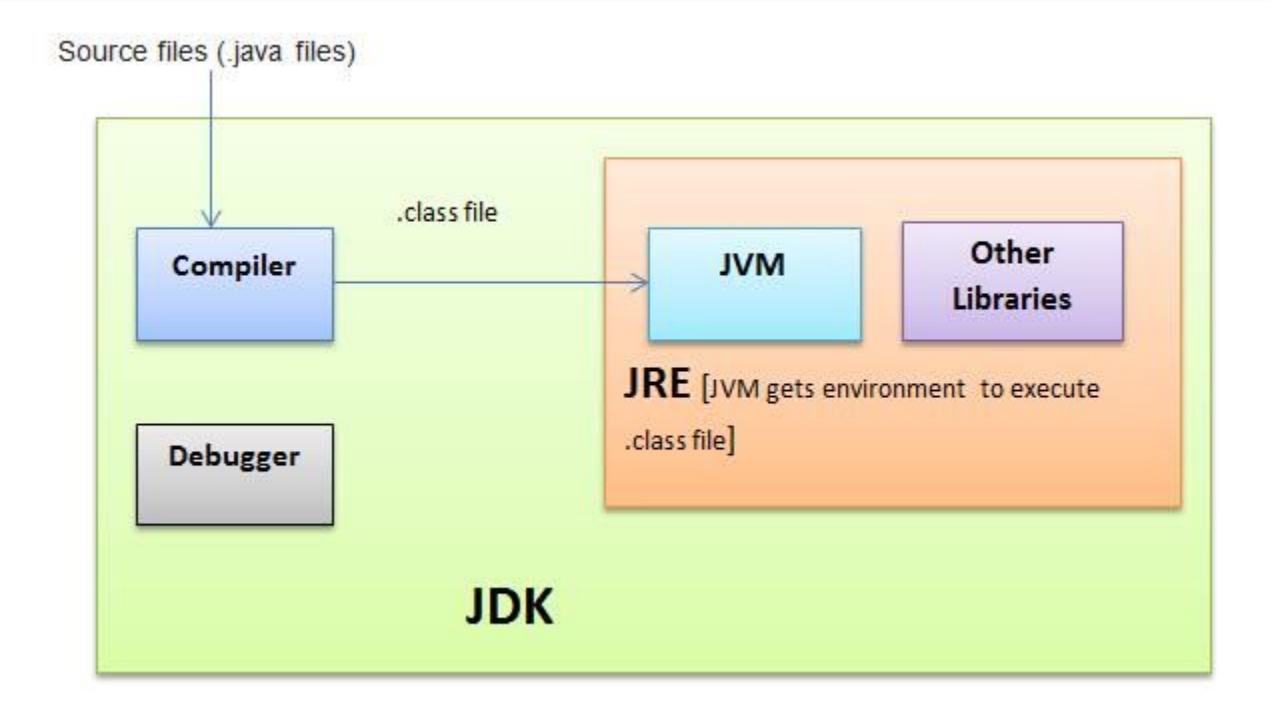
Java Runtime Environment (JRE)

- JRE là môi trường chạy của Java, cung cấp các thư viện và các tệp hỗ trợ cần thiết để chạy các ứng dụng Java. JRE bao gồm JVM, các thư viện Java lõi và các thành phần cần thiết khác để thực thi mã bytecode.

Java Virtual Machine (JVM)

- JVM là một phần mềm mô phỏng máy tính ảo, chịu trách nhiệm thực thi các mã bytecode Java. JVM đảm bảo tính tương thích giữa các nền tảng khác nhau, cho phép chương trình Java chạy trên bất kỳ hệ điều hành nào có cài đặt JVM. Các chức năng chính của JVM bao gồm:
  - **Tải lớp (Class Loader)**: Quản lý và nạp các lớp Java vào bộ nhớ.
  - **Bộ thu gom rác (Garbage Collector)**: Quản lý và giải phóng bộ nhớ không còn được sử dụng.
  - **Trình thông dịch và Just-In-Time (JIT) Compiler**: Chuyển đổi mã bytecode thành mã máy để thực thi nhanh hơn.

Hình minh hoạ



Hình minh hoạ

Garbage Collection

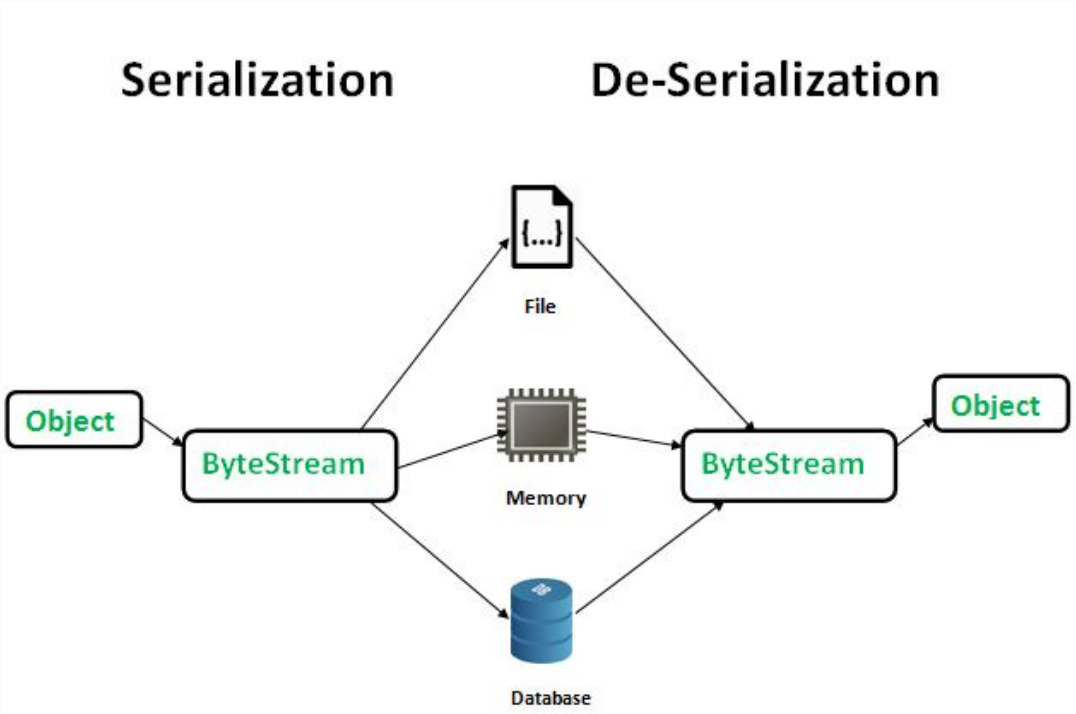
Garbage collection (GC)

- **Định nghĩa:** GC là quá trình tự động quản lý bộ nhớ trong JVM. GC tự động thu hồi bộ nhớ không còn được sử dụng, giúp tránh rò rỉ bộ nhớ.
- Khi một đối tượng không còn tham chiếu từ bất kỳ nơi nào trong chương trình, nó sẽ trở thành "garbage" (rác) và có thể được thu hồi. GC bao gồm các bước chính:
  - **Marking (Đánh dấu)**: GC quét qua các đối tượng trong Heap và đánh dấu các đối tượng đang được tham chiếu.
  - **Sweeping (Quét)**: GC thu hồi bộ nhớ của các đối tượng không được đánh dấu.
  - **Compacting (Nén)**: GC có thể sắp xếp lại các đối tượng còn lại để giảm phân mảnh bộ nhớ.

Serialization

Serialization

- **Serialization** là quá trình chuyển đổi một đối tượng thành một chuỗi byte để có thể lưu trữ đối tượng đó vào tệp tin, cơ sở dữ liệu hoặc truyền qua mạng.
- Khi cần sử dụng lại đối tượng, quá trình **deserialization** sẽ tái tạo đối tượng từ chuỗi byte đã lưu trữ.
- Serialization hữu ích trong các ứng dụng phân tán, lưu trữ dữ liệu tạm thời, và giao tiếp giữa các thành phần khác nhau của hệ thống.



Một số đặc điểm về Serialization

❖ **Serializable Interface**

➢ Interface này không có phương thức nào. Một đối tượng khi implement interface Serializable sẽ được đánh dấu (marker interface) cho biết đối tượng của lớp có thể được serializable.

```
import java.io.Serializable;

public class Person implements Serializable {
    // Định danh duy nhất cho lớp để kiểm tra tính tương thích trong quá trình deserialization
    private static final long serialVersionUID = 1L;
        private String name;
        private int age;

        // Constructor, getters, setters
    }
}
```

Một số đặc điểm về Serialization

- **SerialVersionUID** là một định danh duy nhất cho mỗi lớp serializable. Nó được sử dụng để xác minh rằng người gửi và người nhận của đối tượng serialized có các lớp tương thích. Nếu một lớp không khai báo serialVersionUID, Java sẽ tính toán giá trị này dựa trên các thành phần của lớp, điều này có thể dẫn đến lỗi khi có thay đổi trong lớp.
- Mỗi lớp java chỉ có thể có 1 **SerialVersionUID** duy nhất. Các lớp java là độc lập với nhau nên **SerialVersionUID** có thể cùng giá trị ở mỗi lớp.

```
public class Person implements Serializable {
        private static final long serialVersionUID = 1L;
}

public class Animal implements Serializable {
        private static final long serialVersionUID = 1L;
        private static final long serialVersionUID2 = 2L;    // Không phải là serialVersionUID
}
```

Một số đặc điểm về Serialization

- **Thuộc tính transient**: Nếu có thuộc tính nào không cần được serialized, bạn có thể sử dụng từ khóa transient để loại trừ thuộc tính đó

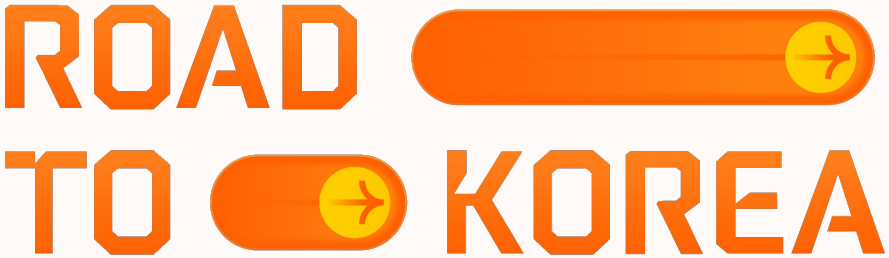
```
public class Person implements Serializable {
        private static final long serialVersionUID = 1L;

        private String name;
        private int age;
        private transient String password;  // Không cần serialize

        // Constructor, getters, setters
}
```

Một số trường hợp sử dụng Serialization

- Lưu trữ đối tượng vào tệp hoặc cơ sở dữ liệu:
  - Lưu trữ cấu hình của người dùng, thông tin phiên làm việc, …
- Truyền đối tượng qua mạng:
  - Gửi thông tin người dùng từ một ứng dụng client tới server, hoặc trao đổi dữ liệu giữa các microservice.
- Lưu trữ trong bộ nhớ tạm thời (Caching):
  - Lưu trữ đối tượng trong bộ nhớ đệm (cache) để tăng tốc độ truy xuất dữ liệu.
- Truyền thông tin giữa các thành phần khác nhau của ứng dụng:
  - Ví dụ: Truyền thông tin giữa các hoạt động (Activity) hoặc dịch vụ (Service) trong ứng dụng.



Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

mail@mail.com hoặc Zalo 0xxx xxx xxx