**I.Stack và Heap**

**1.Stack**

**1. Stack Memory (Bộ nhớ Ngăn xếp)**

* **Stack** là nơi lưu trữ các biến cục bộ (local variables) và các lời gọi hàm (method calls) của một chương trình.
* Mỗi khi một phương thức được gọi, một khối bộ nhớ mới sẽ được tạo ra trên Stack để lưu các biến cục bộ của phương thức đó.
* Sau khi phương thức kết thúc, bộ nhớ được cấp phát cho phương thức trên Stack sẽ được giải phóng ngay lập tức. Điều này có nghĩa là Stack tuân theo cơ chế **LIFO (Last In First Out)**.
* **Quản lý tự động**: Không cần lập trình viên can thiệp vì Java tự động quản lý bộ nhớ trong Stack.
* **Kích thước nhỏ**: Stack có kích thước hạn chế và thường nhỏ hơn Heap.
* **Tốc độ nhanh**: Việc cấp phát và giải phóng bộ nhớ trên Stack nhanh hơn so với Heap do tính tuần tự và tự động của Stack.

**2.Heap**

**2. Heap Memory (Bộ nhớ Đống)**

* **Heap** là vùng bộ nhớ lớn hơn và được sử dụng để lưu trữ các đối tượng trong Java. Mọi đối tượng được tạo bằng từ khóa new đều được lưu trữ trong Heap.
* Đối với một biến đối tượng, tham chiếu của nó được lưu trữ trên Stack, nhưng bản thân đối tượng được lưu trong Heap.
* **Quản lý bằng Garbage Collector**: Java sử dụng cơ chế **Garbage Collector** để tự động quản lý bộ nhớ Heap, thu hồi các đối tượng không còn được tham chiếu và giải phóng bộ nhớ.
* **Không tự động giải phóng**: Khi một đối tượng không còn được tham chiếu, nó vẫn có thể chiếm bộ nhớ trên Heap cho đến khi Garbage Collector dọn dẹp nó.
* **Kích thước lớn**: Heap có kích thước lớn hơn Stack, cho phép lưu trữ nhiều đối tượng phức tạp.

**Khi nào sử dụng Stack và Heap?**

* **Stack**: Sử dụng để lưu trữ các biến nguyên thủy (primitive) và tham chiếu đến các đối tượng. Khi phương thức kết thúc, bộ nhớ sẽ được giải phóng ngay lập tức.
* **Heap**: Dùng để lưu trữ các đối tượng, với vòng đời dài hơn và không bị phụ thuộc vào vòng đời của phương thức đang thực thi.

**So sánh**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Stack** | **Heap** |
| **Nơi lưu trữ** | |  | | --- | | Biến cục bộ, phương thức | | |  | | --- | | Các đối tượng | |
| **Quản lý bộ nhớ** | Tự động giải phóng khi phương thức kết thúc | Garbage Collector tự động quản lý |
| Kích thước | LIFO | Không theo thứ tự cố định |
| **Cơ chế** | Nhỏ hơn, giới hạn kích thước | Lớn hơn |
| **Tốc độ** | |  | | --- | |  |   Nhanh hơn | Chậm hơn |
| **Đối tượng** | Biến cục bộ, biến nguyên thủy | Đối tượng, biến tham chiếu |
| **Sự đồng bộ** | |  | | --- | |  |   Đồng bộ | Không đồng bộ |

**\*\*Cấp phát tĩnh và cấp phát động**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Cấp phát tĩnh** | **Cấp phát động** |
| **Thời điểm cấp phát** | Tại thời điểm biên dịch | Tại thời điểm chạy |
| **Kích thước** | Cố định không thay đổi | Linh hoạt,có thể thay đổi |
| **Vị trí bộ nhớ** | Stack hoặc phần tĩnh của chương trình | Heap |
| **Giải phóng bộ nhớ** | Tự động khi kết thúc chương trình | Garbage Collector quản lý hoặc lập trình viên tự giải phóng |
| **Ưu điểm** | Nhanh và dễ quản lý | Linh hoạt,hiệu quả hơn |
| **Nhược điểm** | Thiếu linh hoạt,gây lãng phí bộ nhớ | Quản lý phức tạp,hiệu suất có thể giảm |