1. Thế nào là cấp phát tĩnh và cấp phát động ?

1.1. Cấp phát tĩnh

* Bộ nhớ được cấp phát tại thời điểm biên dịch.
* Kích thước và địa chỉ bộ nhớ được xác định trước khi chạy chương trình.
* Trong java **static variables** được cấp phát tĩnh khi chương trình bắt đầu chạy và chỉ được giải phóng khi chương trình kết thúc.

1.2. Cấp phát động

* Bộ nhớ được cấp phát tại thời điểm runtime.
* Trong java ,ta sử dụng từ khoá new để tạo ra 1 mảng hoặc 1 đối tượng với bộ nhớ được cấp phát động
* Tất cả các đối tượng được cấp phát động trong Java và được lưu trữ trong Heap memory.
* Trong java có cơ chế Garbage Collector giúp tự đôộng thu gom và giải phóng bộ nhớ .

1. Phân biệt bộ nhớ heap và bộ nhớ stack ?

2.1. Bộ nhớ Stack

* Bộ nhớ stack được sử dụng để lưu trữ các biến cục bộ (local variables), các tham số của hàm (method parameters), và các khung ngăn xếp (stack frames) cho mỗi lời gọi hàm.
* Stack thường có kích thước cố định, được xác định khi chương trình bắt đầu chạy.
* Stack hoạt động theo nguyên tắc **LIFO** (Last In, First Out). Mỗi lần một hàm được gọi, một khung ngăn xếp mới được thêm vào đầu stack để lưu thông tin của hàm đó. Khi hàm kết thúc, khung ngăn xếp được giải phóng
* Tốc đọ truy cập vào stack rất nhanh vì dữ liệu được sắp xếp liên tiếp trong bộ nhớ và không yêu cầu quản lý thủ công.
* Stack Scope : Các biến được cấp phát trên stack có **phạm vi cục bộ**. Điều này có nghĩa là chúng chỉ tồn tại trong phạm vi của hàm (method) hoặc khối mã (code block) mà chúng được khai báo. Khi hàm kết thúc hoặc khi luồng thực thi rời khỏi khối mã đó, các biến này sẽ bị xóa và vùng nhớ trên stack được giải phóng

2.2. Bộ nhớ Heap

* Bộ nhớ heap được cấp phát động, sử dụng để lưu trữ các đối tượng và biến toàn cục trong chương trình.
* Kích thước không bị giới hạn , tuy nhiên cần có cơ chế garbage collector để giúp giải phóng bộ nhớ .
* Các đối tượng lưu trữ trong heap 1 các rải rác do đó tốc dộ truy cập chậm hơn stack
* Heap scope : Các biến được cấp phát trên heap có phạm vi toàn cục hoặc không xác định rõ ràng. Các đối tượng trên heap tồn tại cho đến khi không còn tham chiếu nào trỏ đến chúng, bất kể chúng được tạo ra trong phương thức nào