Java Stack memory

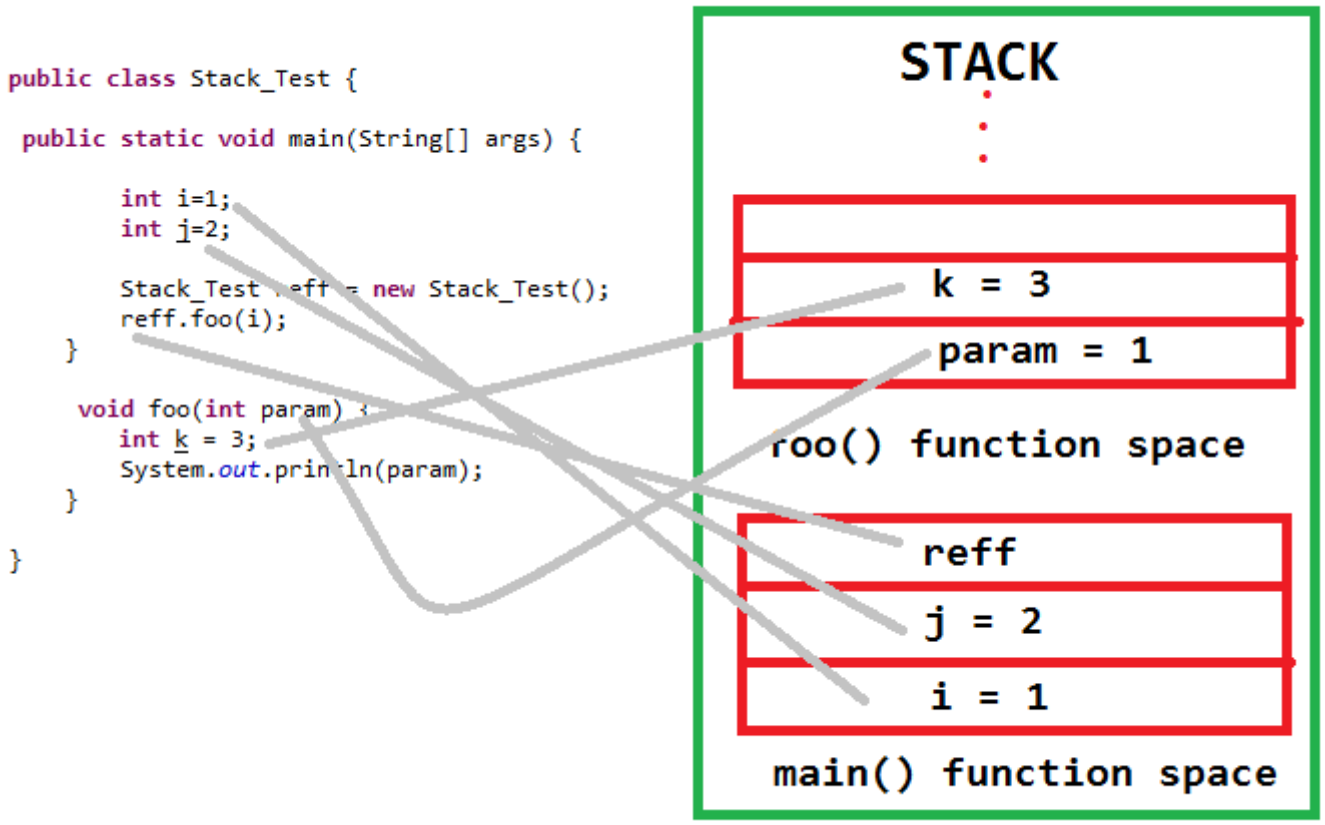
1. Java Stack là gì?

`Stack` là một vùng nhớ được sử dụng để lưu trữ các tham số và các biến local của phương thức mỗi khi một phương thức được gọi ra. Các tham số và các biến local của một phương thức tạo thành một bản ghi kích hoạt, còn được gọi là một stack frame. Các bản ghi kích hoạt được đẩy vào một stack khi phương thức được gọi và đẩy ra khỏi stack khi phương thức trả về. Sự tồn tại tạm thời của các biến này quyết định thời gian sống của các biến.

2. Đặc điểm của của Java Stack memory

- Stack memory được sử dụng cho quá trình thực thi của mỗi thread.
- Bất cứ khi nào gọi 1 hàm, một khối bộ nhớ mới sẽ được tạo trong Stack cho hàm đó để lưu các biến local. Khi hàm thực hiện xong, khối bộ nhớ cho hàm sẽ bị xoá, và giải phóng bộ nhớ trong stack.
- Stack memory bao gồm các giá trị cụ thể của method: các biến local/nguyên thủy và các tham chiếu tới các đối tượng chứa ở trong heap memory được tham chiếu bởi method.
- Stack memory được tham chiếu theo thứ tự LIFO (Last In First Out – vào cuối cùng thì ra đầu tiên). Tức là lưu trữ kiểu ngăn xếp (stack). Khi có một method được thực thi, một block được tạo ra trong stack memory để chứa các biến nguyên thủy local và các tham chiếu tới các object. Khi method kết thúc, block đó sẽ không còn được sử dụng và được phục vụ cho method tiếp theo.
- Stack memory có kích thước rất nhỏ so với Heap memory.

3. Giải thích cách hoạt động của Java Stack



Step 1. Khi chạy chương trình, một thread sẽ khởi tạo và sẽ gọi hàm main ở dòng 1. Một khối bộ nhớ được tạo trong stack cho hàm main().

- `int i=1` : Một biến local được tạo, loại primitive được lưu trong cùng khối bộ nhớ của hàm main()

- `int j=2` : Một biến local được tạo, loại primitive được lưu trong vùng nhớ tiếp theo của stack main()

- `Stack_Test() reff = new Stack_Test()` : một đối tượng được tạo loại Object sẽ được lưu trong bộ nhớ Heap và biến tham chiếu `reff` được lưu trong Stack của hàm main()

Step 2. Hàm `foo()` thì được gọi, vì vậy nó sẽ tạo một khối mới bộ nhớ trong stack cho hàm foo()

- `int param` : Một biến local được tạo (Giá trị được truyền vào), loại primitive được lưu trong cùng khối bộ nhớ của hàm foo()

- `int k=3` : Một biến local được tạo, loại primitive được lưu trong vùng nhớ tiếp theo của stack foo()

- Hàm foo() sẽ kết thúc sau khi hàm `System.out.println()` được thực thi, vì vậy bộ nhớ trong stack cho hàm foo() sẽ được giải phóng.

Step 3. Theo quy luật LIFO, foo() vào sau chết trước, và sau đó hàm main() cũng kết thúc, bộ nhớ trong stack cho hàm main() cũng được giải phóng.

Step 4. Chương trình kết thúc.

4. Chương trình gây lỗi Stack Overflow Error

Chương trình sau là ví dụ gây ra lỗi Stack Overflow Error trong Java, do thực hiện hàm đệ quy dẫn tới không thể giải phóng được bộ nhớ Stack gây tràn bộ nhớ.

```
public class Show_StackOverFlowError {

   public static void main(String[] args) {

      methodOne();
   }

   public static void methodOne(){
     System.out.println("Method One");
     methodTwo();
   }

   public static void methodTwo(){

     System.out.println("Method Two");
     methodOne();
   }
}
```

Output:

```
Method One
Method Two
Method One
Method Two
Method One
Method Two
.
.
.
.
.
.

Exception in thread "main" java.lang.StackOverflowError
at java.base/sun.nio.cs.UTF_8$Encoder.encodeLoop(UTF_8.java:695)
```

```
at java.base/java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:578)
at java.base/sun.nio.cs.StreamEncoder.implWrite(StreamEncoder.java:292)
at java.base/sun.nio.cs.StreamEncoder.implWrite(StreamEncoder.java:281)
at java.base/sun.nio.cs.StreamEncoder.write(StreamEncoder.java:125)
at java.base/java.io.OutputStreamWriter.write(OutputStreamWriter.java:211)
at java.base/java.io.BufferedWriter.flushBuffer(BufferedWriter.java:120)
at java.base/java.io.PrintStream.write(PrintStream.java:526)
at java.base/java.io.PrintStream.print(PrintStream.java:666)
at java.base/java.io.PrintStream.println(PrintStream.java:803)
at Show_StackOverFlowError.methodOne(Show_StackOverFlowError.java:16)
at Show_StackOverFlowError.methodTwo(Show_StackOverFlowError.java:22)
at Show_StackOverFlowError.methodOne(Show_StackOverFlowError.java:17)
at Show_StackOverFlowError.methodTwo(Show_StackOverFlowError.java:22)
at Show_StackOverFlowError.methodOne(Show_StackOverFlowError.java:17)
```

Revision #1
Created 3 November 2019 15:25:07 by Laptrinh.vn
Updated 3 November 2019 15:51:40 by Laptrinh.vn