Discussion-5: Recursion

<u>**Discussion Topic:**</u>
What are the basic components required to create a recursive method? How can you avoid creating an infinitely recursive method? In your post, also provide an example of how recursion can be used in a specific example. Please be sure to provide appropriate source code to illustrate the scenario. In response to your peers, provide an additional example to those posted. Include additional code segments if applicable.

<u>**My Post:**</u>

Hello class,

A method that uses recursion to solve a problem by breaking that problem into smaller subproblems can be defined as a recursive algorithm or a recursive method.

The basic components to create a recursive method are a base case and a recursive case.

- A base case is a condition that when met stops the recursion, usually in an if statement.

A recursive case is a set of code lines or functionalities that are computed if the base case condition is not met, always followed by the recursive method calling itself usually with a modified input. Typically, the code lines and the recursive call are found in an else statement following the if statement checking if the base condition is met. However, If the if statement contains a return statement, the code lines and the recursive call are found right after the if statement.
Note that a recursive method that calls itself with an unmodified input, or a recursive method that does not take an input, will not create an infinitely recursive loop if and only if the base case condition is based on external factors that change independently of the method's input.

To avoid creating an infinitely recursive method, the method needs to contain at least one base case that will eventually be reached. Note that a recursive method can have more than one base case. For example, the recursive method can contain a base case that checks a specific condition, and others can act as safeguards. If the first base case condition is never reached, a safeguard such as a counter can limit the number of recursions based on the available computing memory, preventing a stack overflow error.
On a side note: the Python programming language has a built-in mechanism that limits the number of recursions a program can perform. If needed, this limit can be modified, either decreased or increased, by using the Python system (sys) library.

Here is an example of a recursion method:

```java
import java.util.Random;

public class AreWeThereYet {
    private static final Random randomGenerateMiles = new Random();

    public static void askAreWeThereYet(int totalMilesDriven, int tripTotalMiles) {
        // ---- Base case ---- We've arrived!
        if (totalMilesDriven >= tripTotalMiles) {
            System.out.println("We're here! Finally!");
            return;
        }
        // ---- Recursive case ----
        // Miles driven
        int milesDriven = randomGenerateMiles.nextInt(50) + 1; // Drive 1-50 miles

        // Keep asking and driving
        System.out.println("Are we there yet?");
        System.out.println("Not yet, we've traveled " + totalMilesDriven + " miles.");
        if (milesDriven + totalMilesDriven >= tripTotalMiles) {
            milesDriven = tripTotalMiles - totalMilesDriven;
        }
        System.out.println("--- Drives " + milesDriven + " miles ---");
        totalMilesDriven += milesDriven;
        // ---- Recursive call ----
        askAreWeThereYet(totalMilesDriven, tripTotalMiles);
    }

    public static void main(String[] args) {
        int tripTotalMiles = 100; // Total trip distance
        System.out.println("Trip total miles: " + tripTotalMiles);
        askAreWeThereYet(0, tripTotalMiles);
    }
}
```

Output:

```
Trip total miles: 100
Are we there yet?
Not yet, we've traveled 0 miles.
--- Drives 10 miles ---
Are we there yet?
Not yet, we've traveled 10 miles.
--- Drives 26 miles ---
Are we there yet?
Not yet, we've traveled 36 miles.
--- Drives 17 miles ---
Are we there yet?
Not yet, we've traveled 53 miles.
--- Drives 12 miles ---
Are we there yet?
Not yet, we've traveled 65 miles.
--- Drives 23 miles ---
Are we there yet?
Not yet, we've traveled 88 miles.
--- Drives 12 miles ---
We're here! Finally!
```