**What is Recursion?**

**Recursion** is when a function calls itself to solve a problem. It's like breaking a big problem into smaller, similar problems.

**Real-Life Example:**

Imagine you're looking for your keys in a messy house:

1. Check the living room
2. If not found, check the bedroom (same process)
3. If not found, check the kitchen (same process)
4. Keep repeating until you find them or run out of rooms

**Two Important Parts:**

1. **Base Case**: When to stop (like finding your keys)
2. **Recursive Call**: Calling the same function with different input

**Why Use Recursion?**

* Makes complex problems simpler to understand
* Code looks cleaner and shorter
* Some problems are naturally recursive (like our money growth)

**Financial Forecasting Tool**

This program calculates how much your money will grow over time using recursion. For example, if you invest $1000 at 10% growth per year, it tells you how much you'll have after 5 years.

**How Recursion Helps Here:**

Think of it like climbing stairs:

* To reach step 5, you first need to reach step 4
* To reach step 4, you first need to reach step 3
* And so on...

In our program:

* To calculate money after 5 years, first calculate after 4 years
* Then multiply by growth rate

**How It Works**

1. **Base Case**: If years = 0, return the original amount
2. **Check Memory**: See if we already calculated this year before
3. **Calculate**: Get previous year's value and multiply by (1 + growth rate)
4. **Save Result**: Store the answer so we don't calculate it again

**Time Complexity (How Fast It Runs)**

* **Speed**: Takes n steps where n = number of years
* **Memory Trick**: Saves previous calculations, so if you ask for the same thing again, it's instant
* **Example**: For 10 years, it does 10 calculations first time, then 0 calculations if you ask again

**Making It Faster**

**Current Optimization: Memory Storage**

*// Saves calculated results*

if (memo.containsKey(years)) {

return memo.get(years); *// Use saved result instead of calculating again*

}

**Alternative: Simple Loop (Faster for one-time use)**

public static double simpleCalculation(double initialValue, double rate, int years) {

double result = initialValue;

for (int i = 0; i < years; i++) {

result = result \* (1 + rate);

}

return result;