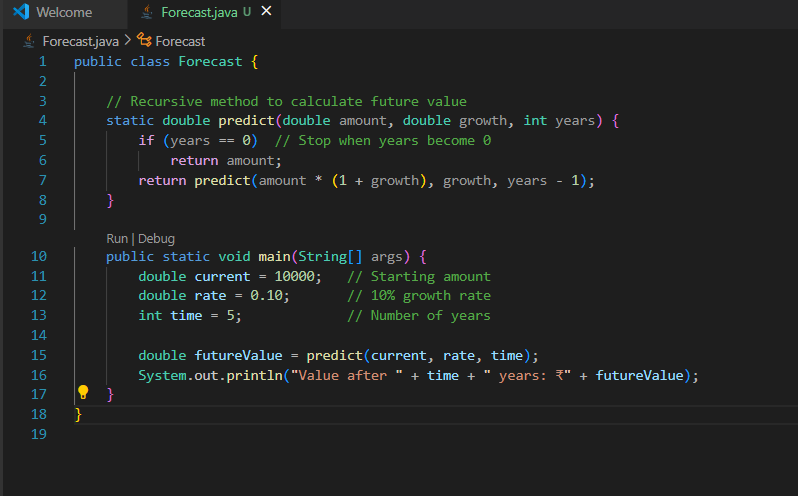
## **Exercise 7:Financial Forecasting**

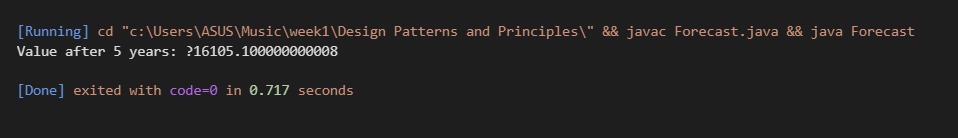
## **What is Recursion?**

**Recursion** is a programming technique where a function **calls itself** to solve a part of a bigger problem. It continues doing this until a specific condition, called the **base case**, is met — which stops the recursion.

## **How Recursion Simplifies Problems**

* It helps in solving problems that can be **broken into smaller, similar subproblems**.
* Makes the code **shorter, cleaner, and easier to understand** for tasks like calculations, searching, or repeating actions.
* Removes the need for writing complex loops in many cases.
* Especially useful for problems with a repetitive, nested, or step-by-step nature.





## **Time Complexity of the Recursive Algorithm**

In the financial forecasting recursion:

**Each recursive call handles 1 year at a time.**

If we need to predict the value for **n years**, the function calls itself **n times** before stopping.

### **Time Complexity:**

* **Time Complexity:** O(n)
  + Because it makes one recursive call for each year.
  + The number of calls increases linearly with the number of years.

## **How to Optimize a Recursive Solution**

Recursion can lead to **excessive or repeated calls**, especially for large values. To avoid this and make the program efficient, we can:

### **1. Use Iteration (Loop instead of Recursion)**

* Replace the recursive function with a simple loop.
* Loops don’t use extra stack memory and run faster for many cases.

### **2. Apply Memoization**

* Store already computed results in a temporary storage (like an array or map).
* Avoids recalculating the same value multiple times.

### **3. Use Tail Recursion (if supported)**

* A special form of recursion where the recursive call is the last action.
* Some languages can optimize tail recursion to work like a loop.