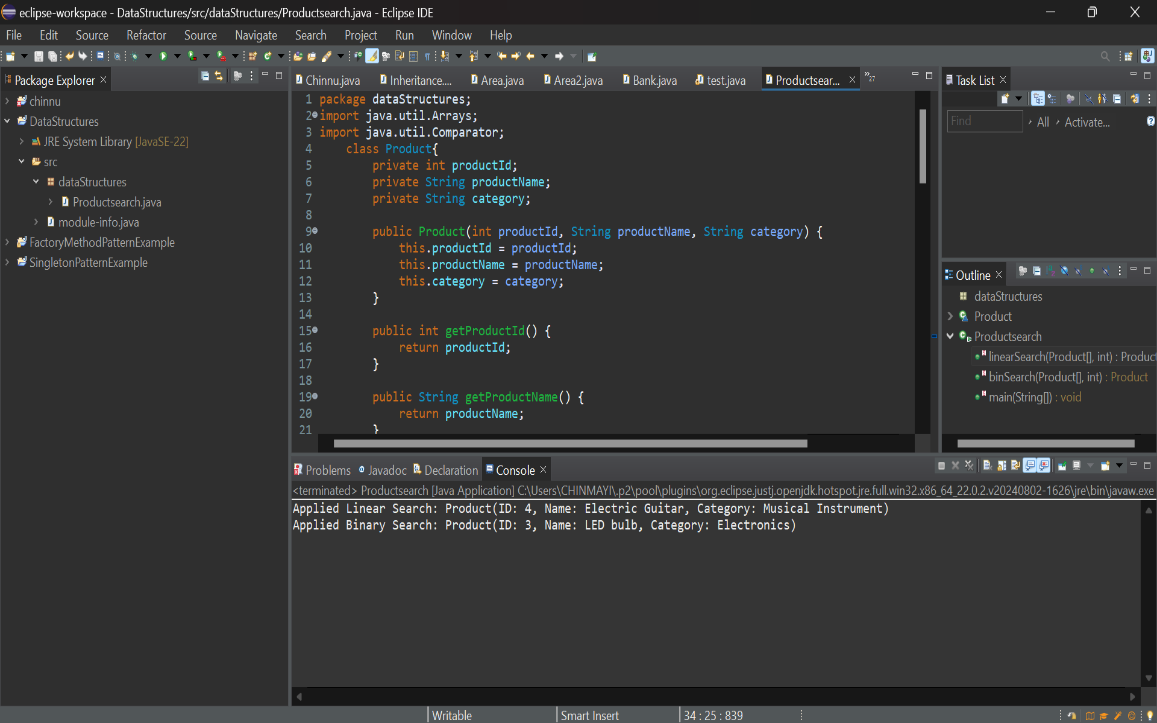
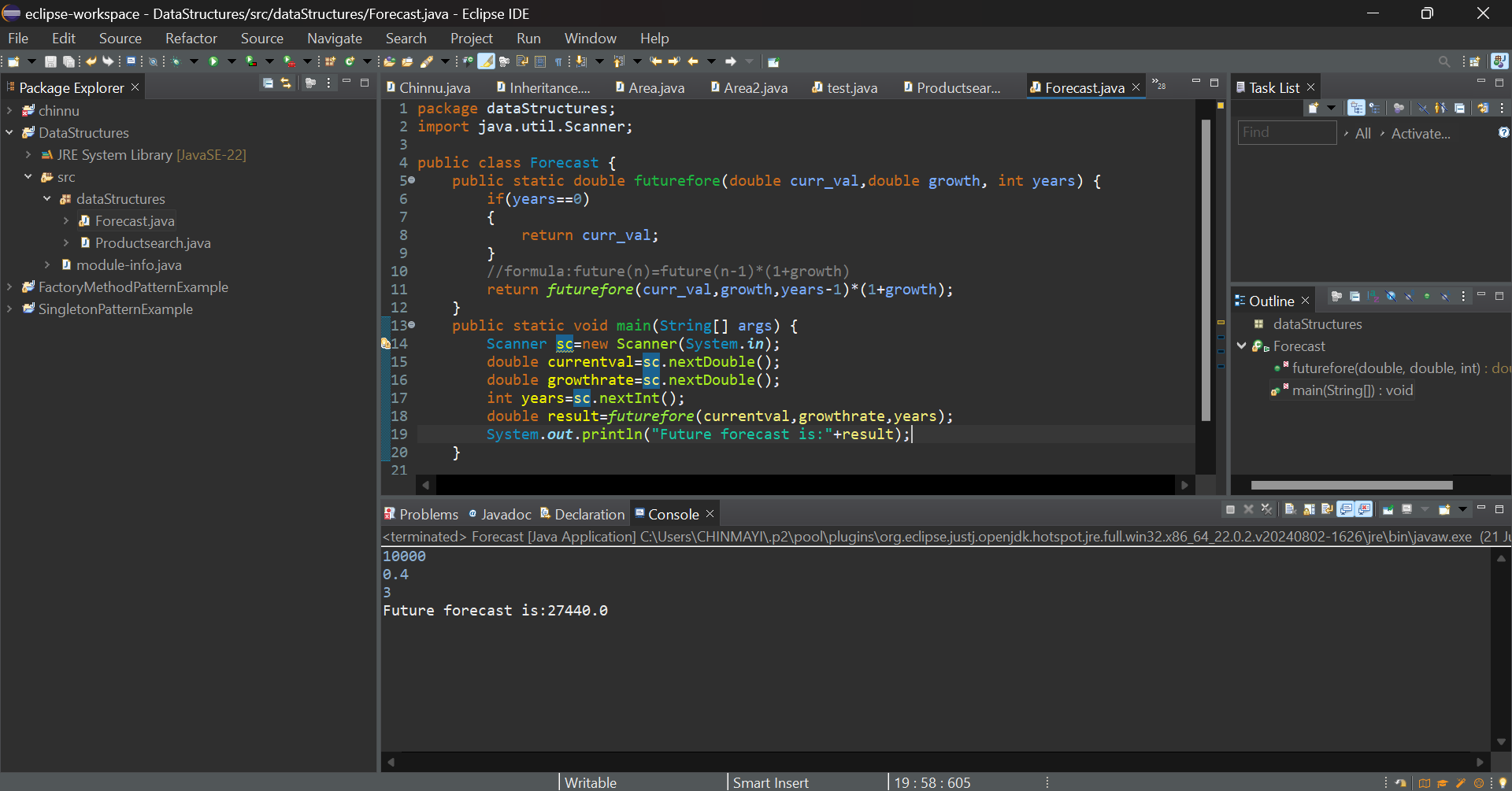
**Data Structures Solutions:**

1. **Productsearch:**

****

1. **Future Forecasting:**

****

**Search cases:**

**Big O Notation:**

* Big O notation is a metric to describe the worst-case time complexity and space complexity of an algorithm.
* This is to pick the most efficient algorithm for a particular problem.

For the search problem,

* Best case(omega notation) is the finding the search element at the first index. Best case time complexity for linear and binary search is **O(1).**
* Average case(theta notation) is the finding element in expected time or in the half or average elements in the array. Average time complexity for linear search is **O(n)** and for binary search is **O(log n).**
* Worst case(Big oh notation) is the finding element in the last index or not finding the element in the array. In this the search goes to extreme end of an array. Worst case time complexity for linear search is O(n) and O(log n) for binary search.

**Where to use search:**

Linear search can be used when the array is small in size and if the search element can be attained at the first indices often.

Binary search can be used when the data is sorted even within the largest data.

In real world scenario, binary search is used mainly in databases which are sorted already.

**Recursion**

Recursion is a technique where a function calls itself to solve smaller modules of a same problem. It contains a best case which is a stopping condition and recursive case where a function calls itself there.

Let us consider a factorial problem. It can be solved by using recursion.

Factorial of a number is, n!=n\*(n-1)\*(n-2)\*….\*1

In this, the base condition is that 0!=1.

Recursive condition is n\*factorial(n-1).

For forecasting future values based on current growth rates,

Introducing a formula for prediction as:

if (years == 0)

return curr\_val;

return futurefore(curr\_val, growth, years - 1) \* (1 + growth);

where years==0 is a base condition and “futurefore” is recursive call for forecasting.

For this time complexity is the O(n). where n is the years.

As the growth is based on no.of years where that is the no.of times the recursion occurs.

Space complexity is the O(n) where the call stack is used.

For this method, it is best for least number of iterations. But when the number of years are getting increased, it leads to **stack overflow**.

Hence, it can be solved by using iterative method rather than recursion when the input size is of medium size.

For best performance, we can use “Math.pow()” function when the input years are larger in size.