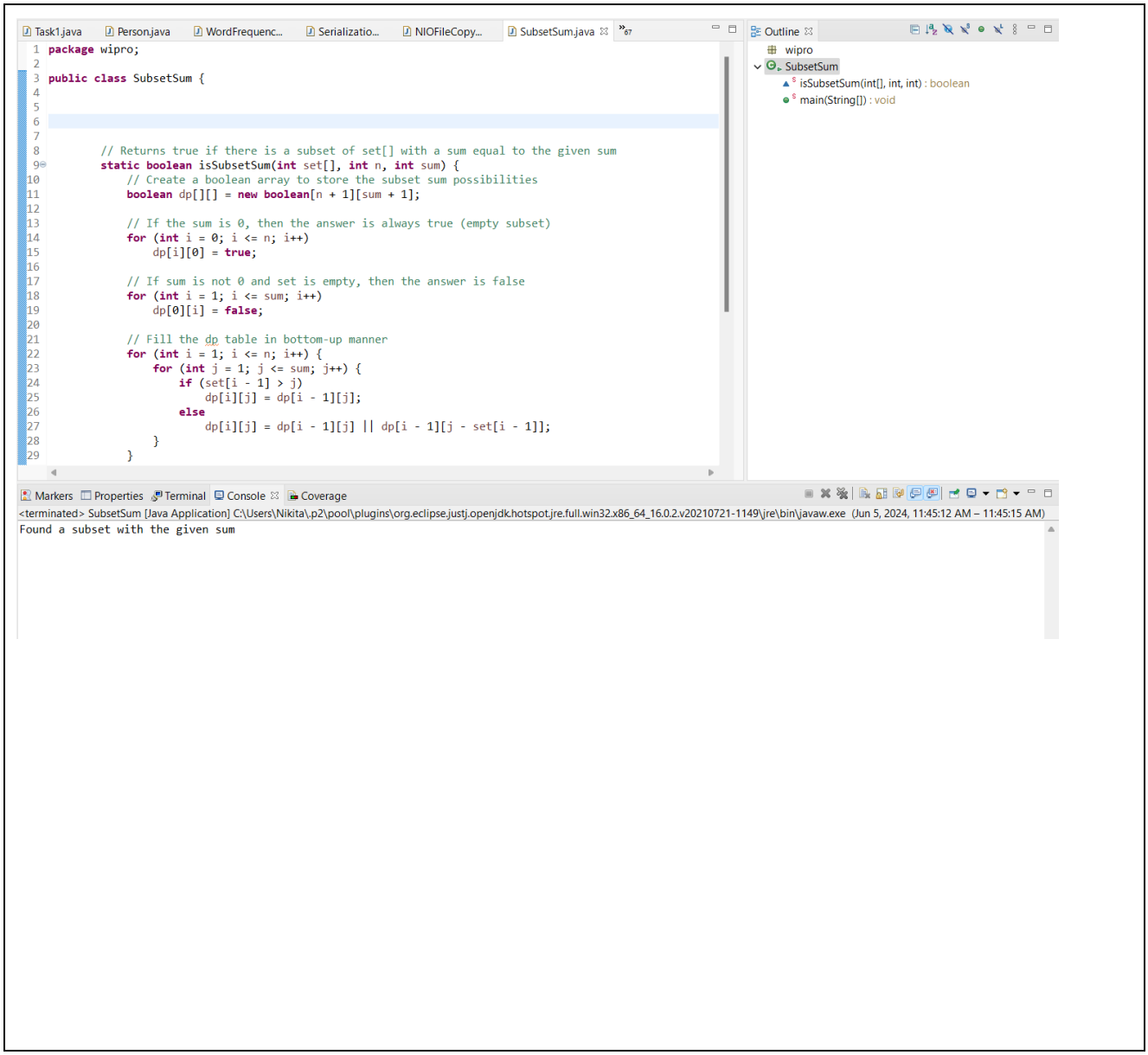
Task 3: New IO (NIO)

Use NIO Channels and Buffers to read content from a file and write to another file.

```java
package wipro;

public class SubsetSum {




       // Returns true if there is a subset of set[] with a sum equal to the given sum
          static boolean isSubsetSum(int set[], int n, int sum) {
              // Create a boolean array to store the subset sum possibilities
              boolean dp[][] = new boolean[n + 1][sum + 1];

              // If the sum is 0, then the answer is always true (empty subset)
              for (int i = 0; i <= n; i++)
                  dp[i][0] = true;

              // If sum is not 0 and set is empty, then the answer is false
              for (int i = 1; i <= sum; i++)
                  dp[0][i] = false;

              // Fill the dp table in bottom-up manner
              for (int i = 1; i <= n; i++) {
                  for (int j = 1; j <= sum; j++) {
                      if (set[i - 1] > j)
                          dp[i][j] = dp[i - 1][j];
                      else
                          dp[i][j] = dp[i - 1][j] || dp[i - 1][j - set[i - 1]];
                  }
              }

              return dp[n][sum];
          }

          public static void main(String args[]) {
              int set[] = {3, 34, 4, 12, 5, 2};
              int sum = 9;
              int n = set.length;

              if (isSubsetSum(set, n, sum))
                  System.out.println("Found a subset with the given sum");
              else
                  System.out.println("No subset with the given sum");
          }
      }
```




```java
package wipro;

public class SubsetSum {



        // Returns true if there is a subset of set[] with a sum equal to the given sum
        static boolean isSubsetSum(int set[], int n, int sum) {
            // Create a boolean array to store the subset sum possibilities
            boolean dp[][] = new boolean[n + 1][sum + 1];

            // If the sum is 0, then the answer is always true (empty subset)
            for (int i = 0; i <= n; i++)
                dp[i][0] = true;

            // If sum is not 0 and set is empty, then the answer is false
            for (int i = 1; i <= sum; i++)
                dp[0][i] = false;

            // Fill the dp table in bottom-up manner
            for (int i = 1; i <= n; i++) {
                for (int j = 1; j <= sum; j++) {
                    if (set[i - 1] > j)
                        dp[i][j] = dp[i - 1][j];
                    else
                        dp[i][j] = dp[i - 1][j] || dp[i - 1][j - set[i - 1]];
                }
            }
```