This means that our knapsack has capacity $(1+2\epsilon)W$. We throw out all items of weight exceeding W, since these can't be used in the solution we're comparing against.

We now round all remaining weights down to the nearest multiple of $\epsilon W/n$, and then multiply them all by $n/(\epsilon W)$. This means that all weights are now integers between 0 and n/ϵ. (Note that the items of weight less than $\epsilon W/n$ do get rounded down to 0, and yes, this means we will probably take them all, but as we'll see this is not a problem.)

So in time polynomial in n and $1/\epsilon$, by the dynamic programming algorithm for the knapsack problem with small weights, we can find the subset of (rounded) weight at most W which achieves the greatest value. The solution of (true) weight W and value V that was promised to exist must be available as an option, since its weight only went down, and since we find the best subset, we find one of value at least V. When we put all these items in our knapsack, each has a weight that may be up to $\epsilon W/n$ more than we thought (due to rounding down), so we use a weight of at most $W + n(\epsilon W/n) = W(1+\epsilon)$.

[1] ex662.412.328