- Base classes/functions are seemingly fine
  - set, get and pack functions for each header
  - have been debugged and work as intended
  - the only issue is variable input types for set functions
    - e.g. function expects DWORD, but user might input and int, short int, singed char, etc…
    - how well can these variable types be converted?
    - Other options (templates, unions) are notorious for errors

- Super classes!
  - “Write once, read many”
  - More useful for unpacking, rather than unpacking
  - Create functions to give back many combinations of data
  - Main issues:
    - User needs to point to empty buffer of right size, how does user know what the size is? (they don’t)
      - Could have separate function that returns size needed for the intended function (get_someinfo_size(); get_someinfo(empty_buffer[size]);)
    - Is the data returned in SNOLAB formatted 32-bit DWORDs, or in as individual pieces of information?
      - If latter, needs some standard for how that information is presented

```
//unpacking class
class unpack_databank{
private:
    DWORD *data_bank;
    DWORD x7_header[2];
    DWORD *x6_header;
    int dsize;
public:
    unpack_databank(DWORD *databank_array,int databank_size){ //point to entire data bank array, input data bank size
        dsize = databank_size;
        data_bank = new DWORD[dsize];
        for(int j = 0;j<databank_size;j++){ //fill private data_bank array with input
            data_bank[j] = databank_array[j];
        }
        //get number of prims in event and fill in 0x7 header array
        int np;
        for (int j = 0;j<dsize;j++){
            if(head_type(data_bank[j]) == 0x7){
                x7_header[0] = data_bank[j];
                x7_header[1] = data_bank[j+1];
            }
        }
        prim_header ph(x7_header);
        np = ph.get_nr_prims_event();
        //create 2d array for 0x6 header array
        x6_header = new DWORD[6][np];

        //fill in 0x6 header array
        int k = 0;
        for (int j = 0;j<dsize;j++){
            if(head_type(data_bank[j]) == 0x6){
                x6_header[0][k] = data_bank[j];
                x6_header[1][k] = data_bank[j+1];
                x6_header[2][k] = data_bank[j+2];
                x6_header[3][k] = data_bank[j+3];
                x6_header[4][k] = data_bank[j+4];
                x6_header[5][k] = data_bank[j+5];
                k++;
            }
        }
    }

    ~unpack_databank(){
        delete[] data_bank;
        delete[] x6_header;
    }

    int get_num_prims(){
        int np;
        prim_header ph(x7_header);
        np = ph.get_nr_prims_event();
        return np;
    }

    void get_all_prim_info(DWORD *empty_buffer){ //size of buffer needs to be 6*(num prims) + 2
        int np;
        prim_header ph(x7_header);
        np = ph.get_nr_prims_event();

        empty_buffer[0] = x7_header[0];
        empty_buffer[1] = x7_header[1];
        int x;
        for (int k = 0;k<np;k++){
            for(int j = 0;j<6;j++){
                x = 6*k + j + 2;
                empty_buffer[x] = x6_header[j][k];
            }
        }
    }
};
```