person.h

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 33
 3: // Tjalling Otter & Emiel Krol
 4: // Person class: interface header
 5:
 6: #ifndef INCLUDED_PERSON_
 7: #define INCLUDED_PERSON_
 8:
 9: #include <string>
10: #include <iostream>
11: #include <sstream>
12:
13: class Person
14: {
15:   std::string d_name;     // name of person
16:   std::string d_address;  // address field
17:   std::string d_phone;    // telephone number
18:   size_t      d_mass;     // the mass in kg.
19:
20:   public:
21:     void setName(std::string const &name);
22:     void setAddress(std::string const &address);
23:     void setPhone(std::string const &phone);
24:     void setMass(size_t mass);
25:     // Setters
26:
27:     std::string const &name()     const;
28:     std::string const &address()  const;
29:     std::string const &phone()    const;
30:     size_t mass()                 const;
31:     // Getters
32:
33:     void insert (std::ostream &outputStream); // Storing data
34:     void extract(std::istream &inputStream);  // Extracting data
35: };
36:
37: #endif
```

person.ih

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 33
 3: // Tjalling Otter & Emiel Krol
 4: // Person class: implementation internal header
 5:
 6: #include "person.h"
 7:
 8: #define CERR std::cerr << __FILE__": "
 9:
10: using namespace std;
```

basicFunctions.cc

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 33
 3: // Tjalling Otter & Emiel Krol
 4: // Person member functions: basic setters and getters
 5:
 6: #include "person.ih"
 7:
 8: void Person::setName(string const &name)
 9: {
10:   d_name = name;
11: }
12:
13: string const &Person::name() const
14: {
15:   return d_name;
16: }
17:
18: void Person::setAddress(string const &address)
19: {
20:   d_address = address;
21: }
22:
23: string const &Person::address() const
24: {
25:   return d_address;
26: }
27:
28: // Phone number setter defined seperately
29:
30: string const &Person::phone() const
31: {
32:   return d_phone;
33: }
34:
35: void Person::setMass(size_t mass)
36: {
37:   d_mass = mass;
38: }
39:
40: size_t Person::mass() const
41: {
42:   return d_mass;
43: }
```

extract.cc

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 33
 3: // Tjalling Otter & Emiel Krol
 4: // Person member function: extract person data from istream
 5:
 6: #include "person.ih"
 7:
 8: void Person::extract(istream &inputStream)
 9: {
10:   string inputString;
11:   getline(inputStream, inputString);    // Get full line
12:   istringstream ss(inputString);   // Transfer line to istringstream
13:   size_t index = 0;                     // Initialise counter for switch
14:
15:   while (getline(ss, inputString, ',')) // While an element can still be extracted
16:   {
17:     switch (index)                      // Assign the object variables in order
18:     {
19:       case 0:
20:         setName(inputString);
21:         break;
22:       case 1:
23:         setAddress(inputString);
24:         break;
25:       case 2:
26:         setPhone(inputString);
27:         break;
28:       case 3:
29:         setMass(stoi(inputString));
30:         break;
31:     }
32:   ++index;
33:   }
34: }
```

insert.cc

```
    1: // Programming in C/C++
    2: // Week 4: Assignment 33
    3: // Tjalling Otter & Emiel Krol
    4: // Person member function: insert data into ostream
    5:
    6: #include "person.ih"
    7:
    8: void Person::insert(ostream &outputStream)
    9: {
   10:   outputStream << "NAME:    " << name()     << '\n';
   11:   outputStream << "ADDRESS: " << address()  << '\n';
   12:   outputStream << "PHONE:   " << phone()    << '\n';
   13:   outputStream << "MASS:    " << mass()     << '\n';
   14: }
   15: // Inserts all object characteristics into ostream. It was assumed that the
   16: // variable identifiers were also desirable.
```

setPhone.cc

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 33
 3: // Tjalling Otter & Emiel Krol
 4: // Person member function: set phone number after verification
 5:
 6: #include "person.ih"
 7:
 8: void Person::setPhone(string const &phone)
 9: {
10:   if (phone.empty())
11:     d_phone = " - not available -";
12:   else if (phone.find_first_not_of("0123456789") == string::npos)
13:     d_phone = phone;
14:   // Switched the two options above around from the example, as an empty string
15:   // will also not contain any non-numerical characters.
16:   else
17:     cout << "A phone number may only contain digits\n";
18: }
```

main.ih

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 34
 3: // Tjalling Otter & Emiel Krol
 4: // Main file: internal header
 5:
 6: #include <iostream>
 7: #include <string>
 8:
 9: using namespace std;
10:
11: void populateArray(Person array[], size_t sizeArray);
12: void printArray(Person array[], size_t sizeArray);
```

main.cc

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 34
 3: // Tjalling Otter & Emiel Krol
 4: // Main file
 5:
 6: #include "main.ih"
 7:
 8: int main()
 9: {
10:   Person personArray[5];  // Define an array of five Person objects
11:
12:   size_t arraySize = sizeof(personArray) / sizeof(personArray[0]);  // Determine length array
13:
14:   populateArray(personArray, arraySize);  // Populate array using input
15:   printArray(personArray, arraySize);     // Print array using object info
16: }
```

populateArray.cc

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 34
 3: // Tjalling Otter & Emiel Krol
 4: // Function: populate array using user input
 5:
 6: #include "person/person.h"
 7: using namespace std;
 8:
 9: void populateArray(Person array[], size_t sizeArray)
10: {
11:   for (size_t index = 0; index != sizeArray; ++index) // Loop through array
12:   {
13:     cout << "? ";               // Ask for user input
14:     array[index].extract(cin);  // Input that data into the extract function
15:   }
16: }
```

printArray.cc

```
 1: // Programming in C/C++
 2: // Week 4: Assignment 34
 3: // Tjalling Otter & Emiel Krol
 4: // Function: print object array
 5:
 6: #include "person/person.h"
 7: using namespace std;
 8:
 9: void printArray(Person array[], size_t sizeArray)
10: {
11:   for (size_t index = 0; index != sizeArray; ++index) // For each array element
12:     array[index].insert(cout);  // Output the object's info using insert
13: }
```