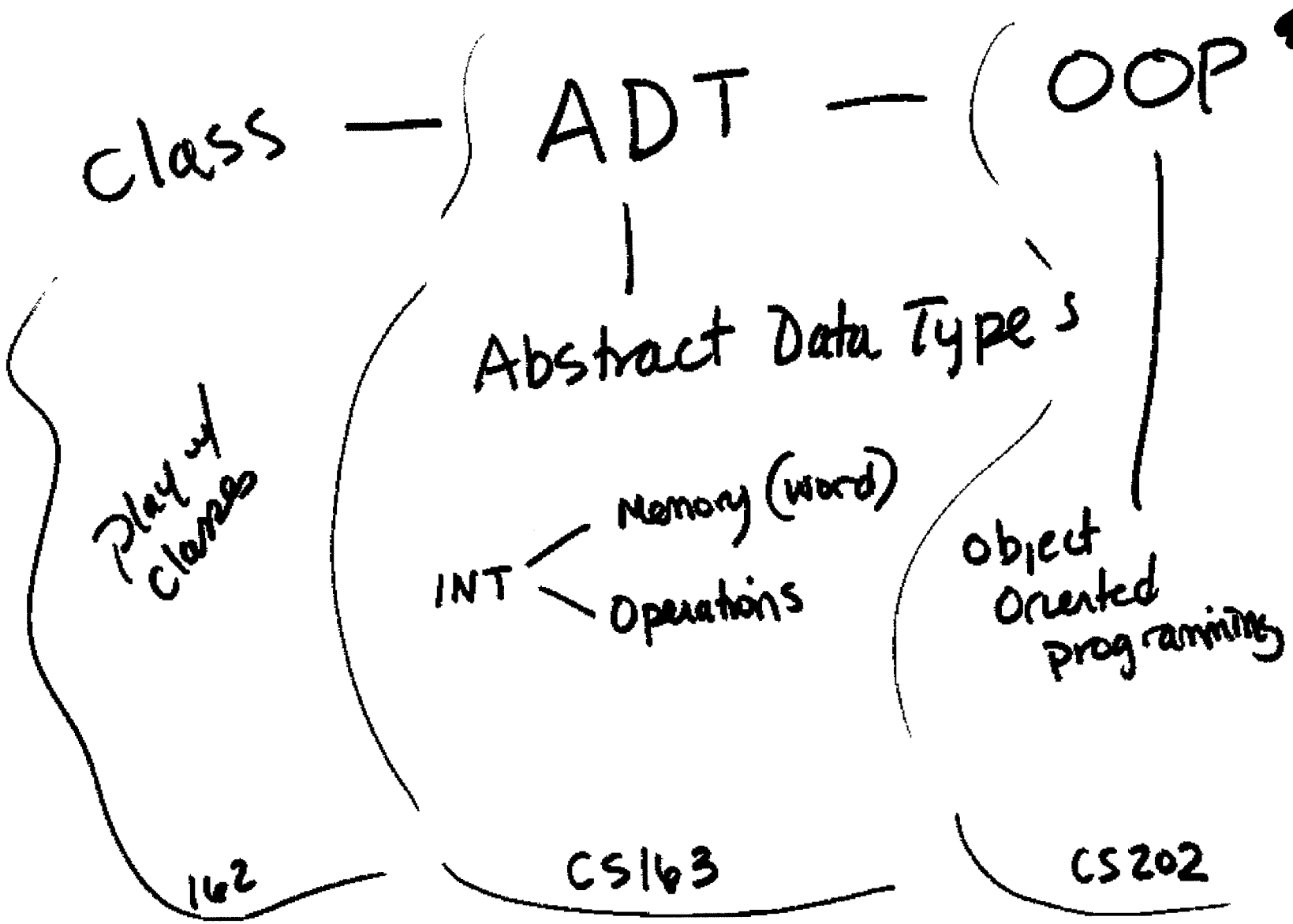
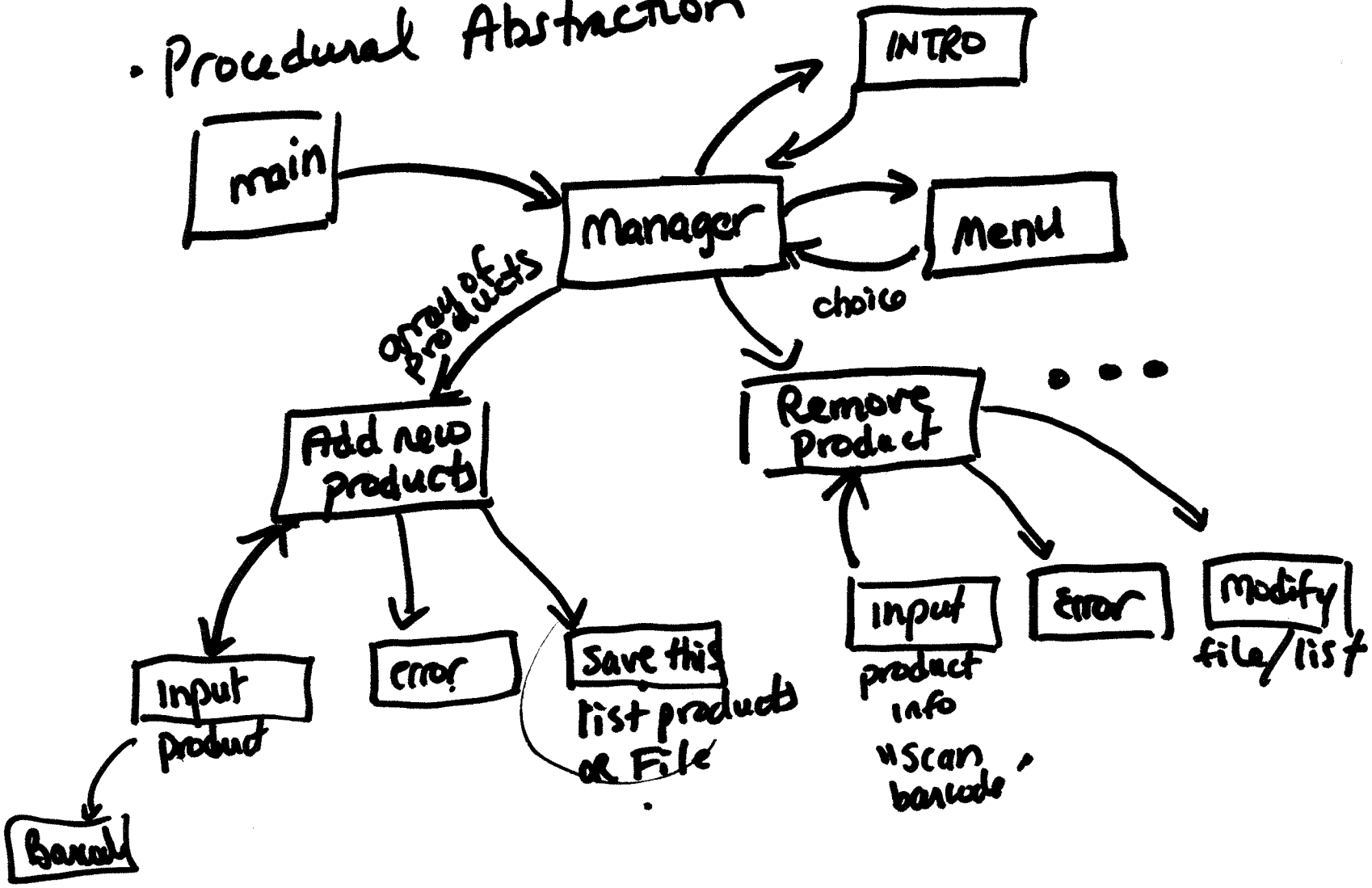

Class
ADT
OOP
Abstract Data Types
Play w/ classes
INT
Memory (word)
Operations
Object Oriented programming
162
CS163
CS202

- Manage inventory of store
- Procedural Abstraction



Structures

- Group different types of data
- C & C++
- Simplify
  - Grouping
  - Arrays of Arrays
  - Passing of data (pass by reference)

```
struct Inventory
{
    char name[21];
    char description[131];
    int quantity;
    char barcode[13];
    float price;
};
```

← A new data type name

should be constants

Members

Global

Procedural Abstraction

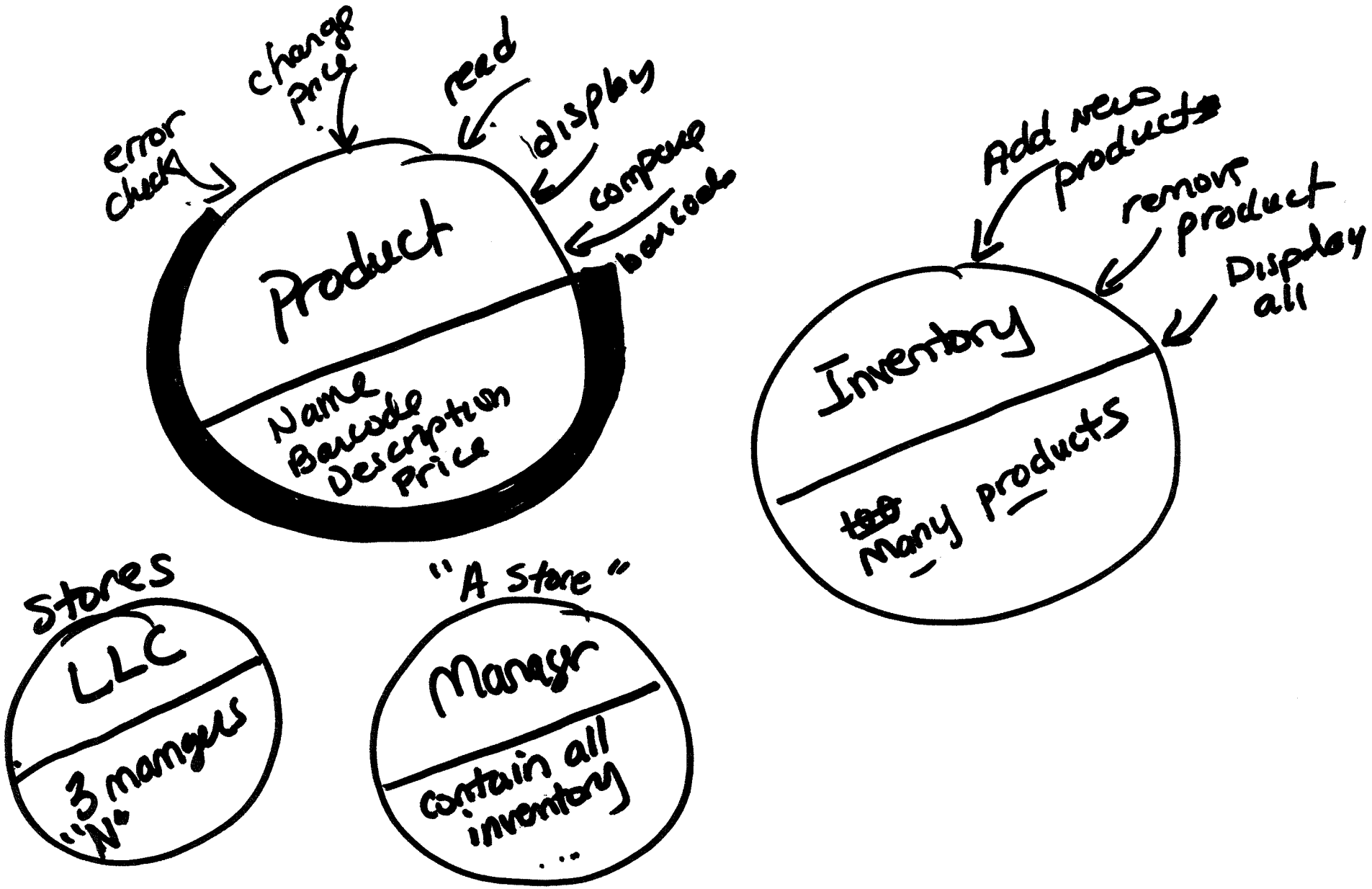
manager

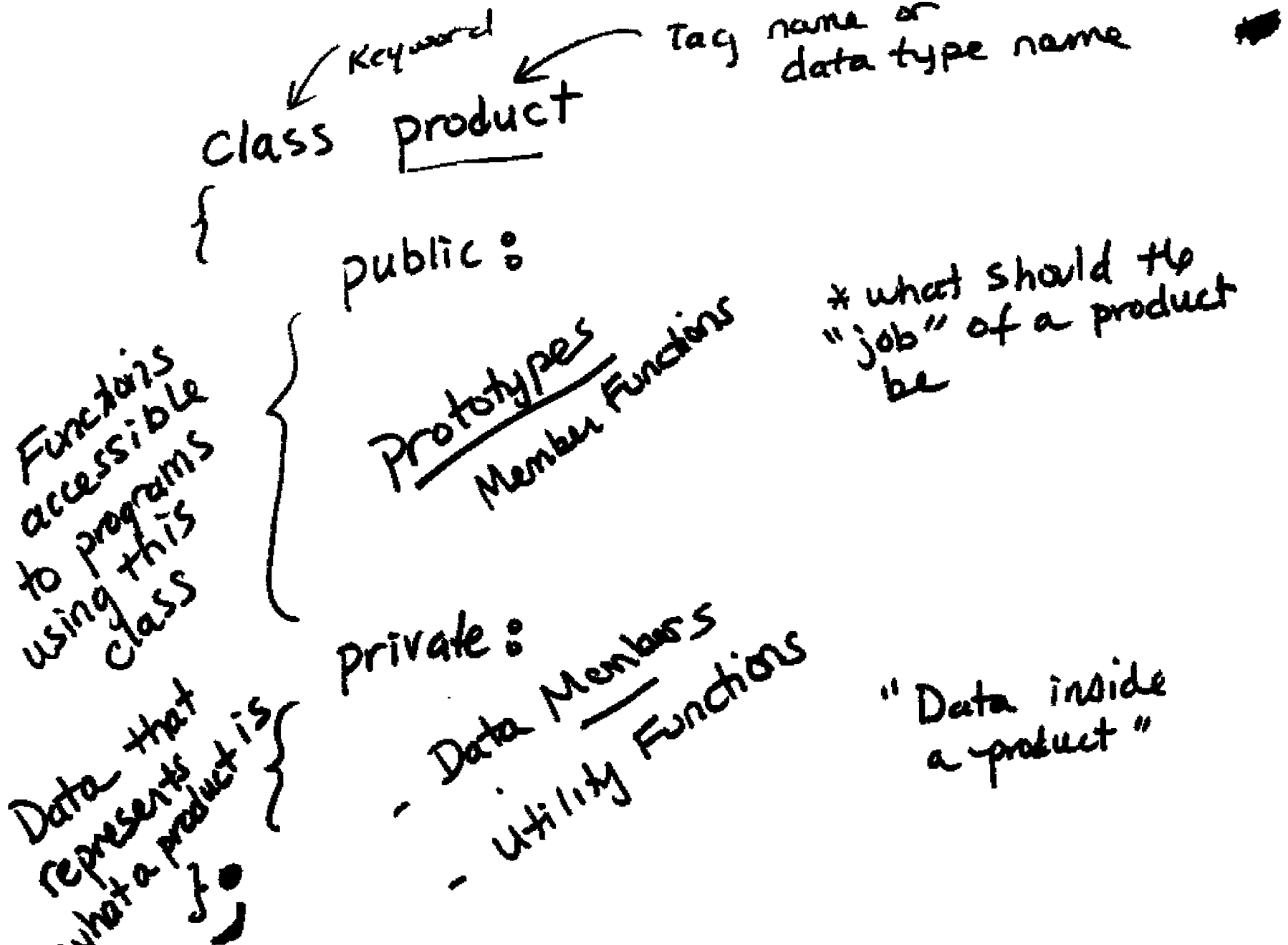
```
Inventory array_dry [100];
int num_dry = 0;
Inventory array_dairy [100];
int num_dairy = 0;
if(!add ( array_dry, num_dry, array_dairy, num_dairy ))
    cout << "Error";
```

Prototype

```
int add ( Inventory [], int &, Inventory [], int &);
```

Data Abstraction




Keyword
Tag name or data type name
Class Product
{
public:
Prototypes
Member Functions
Functions accessible to programs using this class
* what should the "job" of a product be
private:
- Data Members
- Utility Functions
Data that represents what a product is
};
"Data inside a product"

Class Interface

Product.h

```
#include
// const
// structs
// prototypes

class product
{   public:
        product();          // constructor
        void read();
        void display();
        int  contact_distributor();
        int  check_barcode(char barc[]);
    private:
        char name[SIZE_NAME];
        float price;
        char barcode[SIZE_BAR];
        char distributor[100];
};
```

initialize data members to some values

Prototypes

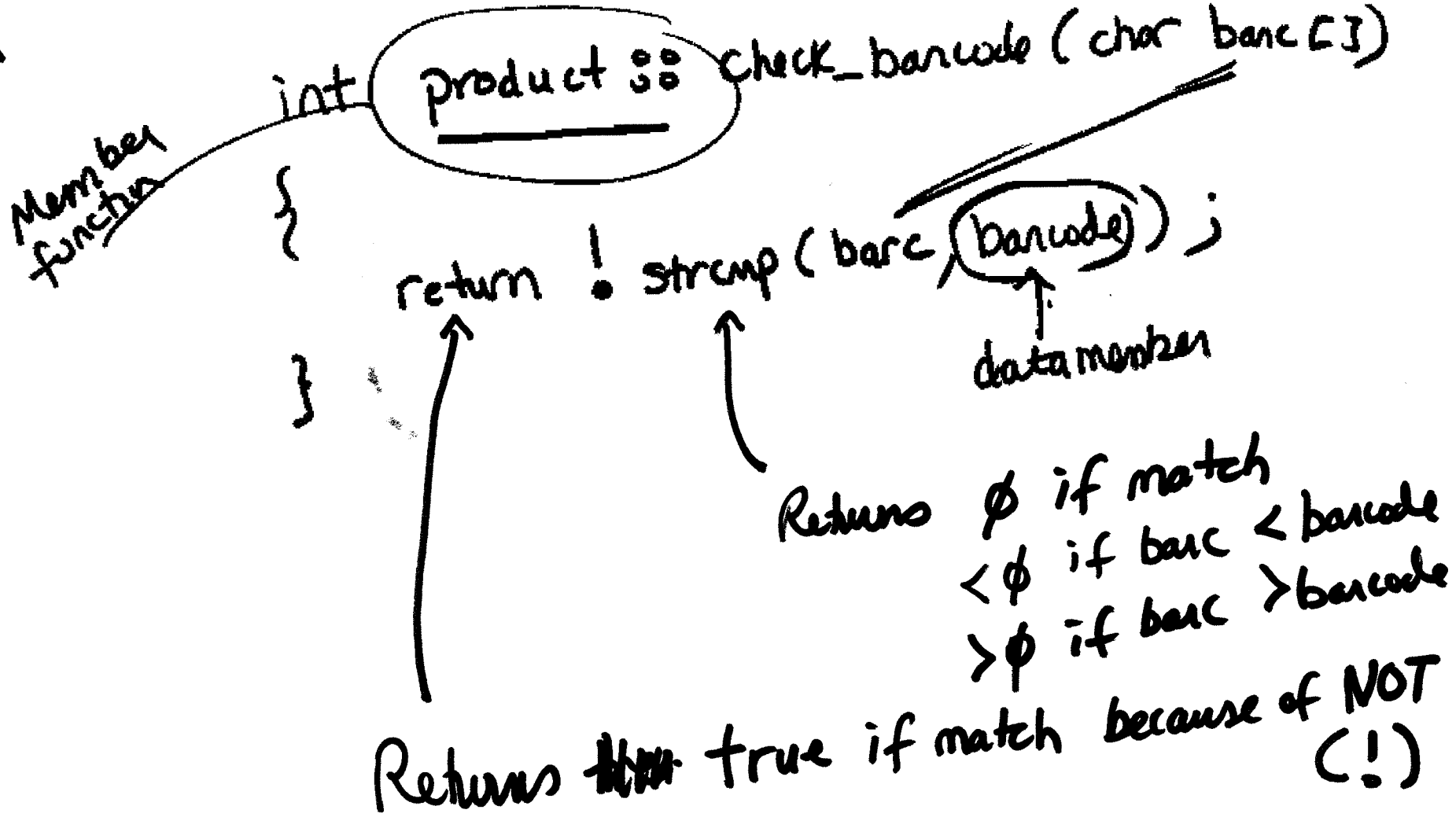
Data members

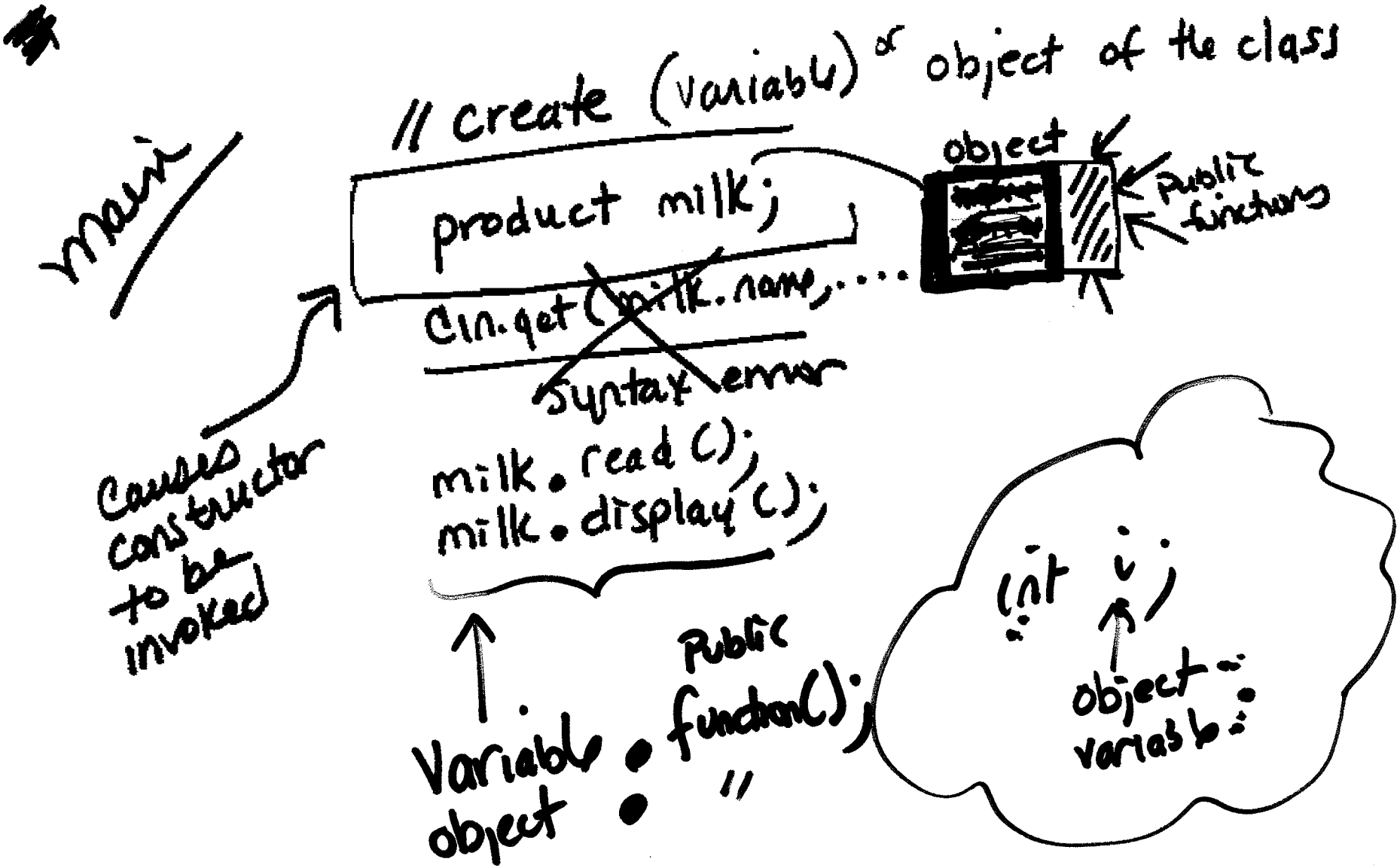
.h

Yikes!

Later: .cpp

```
product :: product ()    //initialize data members
          ↑ function name
classname ↑
          scope resolution operator
{
    name [0] = '\0';
    price = 0.0;
    barcode [0] = '\0';
    distributor [0] = '\0';
}
void product :: read ()
{
    cout << " please enter product name ";
    cin.get ( name, SIZE_NAME, '\n');
    cin.ignore (100, '\n');   cout << " Enter Price ";
    cin >> price ;            cin.ignore ();
        ;
}
```

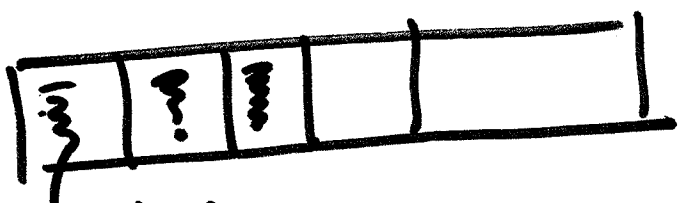

int product :: check_barcode (char barc[])
Member function
{
return ! strcmp (barc , barcode) ;
}
data member
Returns 0 if match
< 0 if barc < barcode
> 0 if barc > barcode
Returns true if match because of NOT (!)


// create (variable) of object of the class
main
product milk;
object
Public functions
cin.get(milk.name,....
Syntax error
milk.read();
milk.display();
Causes constructor to be invoked
Variable object
Public function();
//
int i;
object variable

Main

```
product all [SIZE];
int num_prod = 0;
    }
load_inventory (all, num_prod);
```

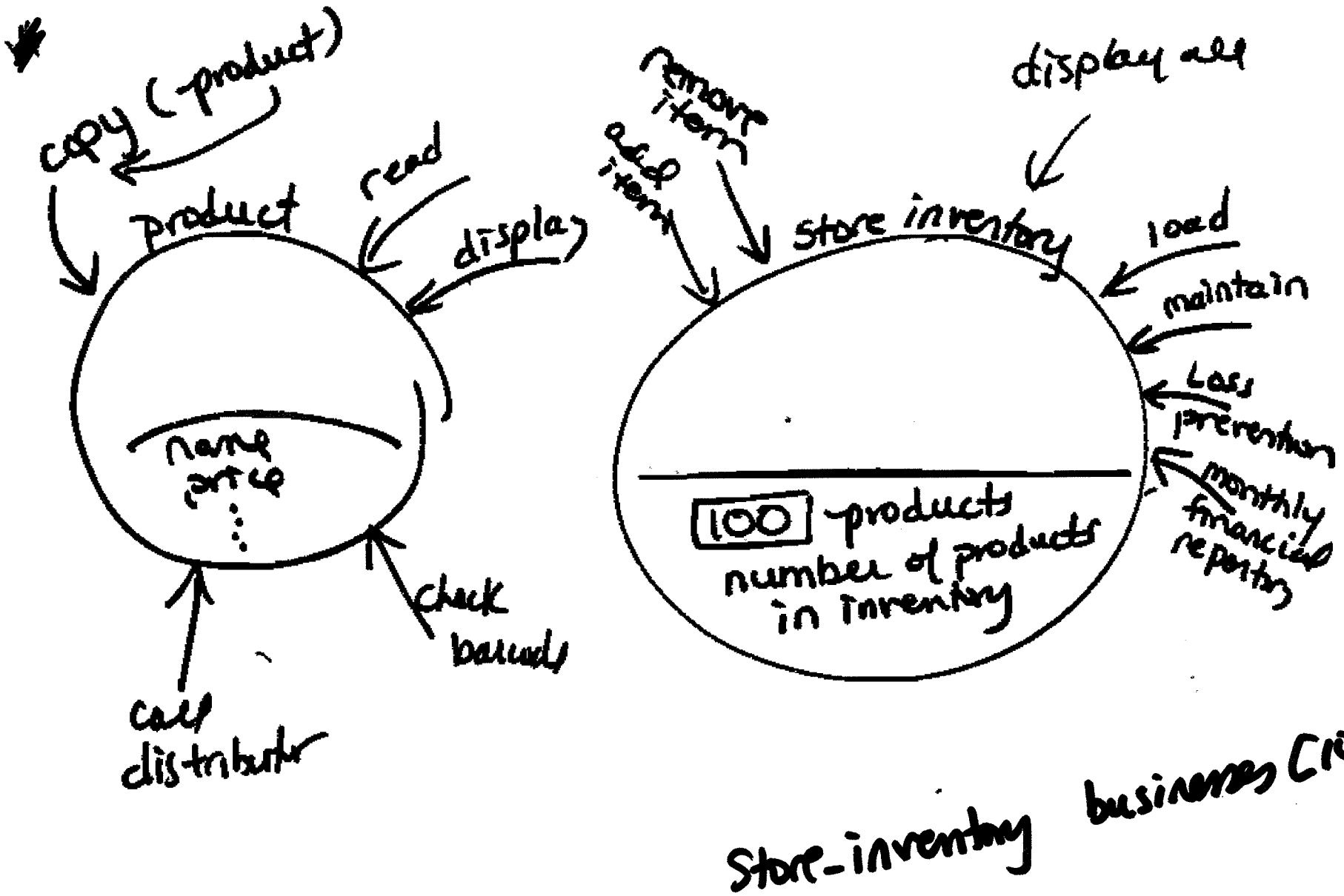
```
void load_inventory (product array [],
                     int & num_prod)
{
    :
    :
    array [num_prod] . read ();
    ++num_prod;
}
```



Using Procedural Abstactn Insted

NO classes

```
void load_inventory ( product array [],
                      int & num_prod)
{
   :
   cout << "Enter name ..." ;
   cin.get ( array [num_prod].name, SIZE_NAME, '\n');
   cin.ignore();
   cout << " Enter price ... " ;
   cin >> array [num_prod].price;
   cin.ignore();
   :
   ++ num_prod;
```



Store_inventory businesses[100];

Product.class ← already!

Class store_inventory

```
{ public:
      store_inventory();
      int add_item ( Product &);
      int remove_item ( char barcode [ ]);
      int display_all ();
      int load_all ();
      int input_product ( );

  private:
      Product all_products [SIZE];
      int num_prod;
};
```