- User vs. Application
- Client vs. Client Program
- Data Structure
- Abstract Data Type - Is a new ADT
- Data Abstraction - Process of Reading

So, what is the difference between:

Data Structures vs. ADT?
Readability

What happens if these are more fields?

```
int add (char *name, char *id, float grade)
{
  class roster;
}
```

```java
public class Student
{
  public Student (char *id, char *name, float grade)
  {
    this;
  }
}
```
Copy
Malware

\[ \text{head} \rightarrow \text{student} = \text{student} \]

If

\[ \text{head} \rightarrow \text{new node} \]

\[ \text{head} = \text{new node} \]

Else

\[ \text{head} \rightarrow \text{fake function} (\text{student 85}) \]

ListClass

100
Smith

\text{student object}

\text{Client}

Please input

Pended
Assumption: head is NULL

head = new node;
head.font = student.copy(s);

Node next = NULL;

Student

Client

add function(student list)
So To copy a student
struct student
{
    char * name;
    char * id;
    float grade;
};

struct node
{
    student * student;
    node * next;
};

class list
{
    public:
    int add(student &);
    private:
    node * head;
}
#include "list.h"

structs
    clean interface

#include "list.h"

just class
    Member Functions

#include "list.h"

Application
    "test bed"

#include "list.h"

std::vector<NDNode> functions

get list.cpp

list.cpp

client.cpp
int student::set (char * nme,
                 char * newid,
                 newgrade)
{
    float

    name = new char [strlen(nme) + 1];
    strcpy(name, nme);

    psu-id = new char [strlen(newid) + 1];
    strcpy(psu-id, newid);

    grade = new grade;
}

```c
int student::copy ( student & old )
{
    name = new char [ strlen ( old.name ) + 1 ];
    strcpy ( name, old.name );
    PSU-Id = new char [ strlen ( old.PSU-Id ) + 1 ];
    strcpy ( PSU-Id, old.PSU-Id );
    grade = old.grade;
}
```
Given this, what do you think we should do about:

- Prompts
- Output of data
- Input of data
- Error messages
- Data members

Using classes to build ADTs
Defining the List Class