I provide the classes I design in the project for your reference. The purpose of it is to help you get some ideas how the classes should be developed.

Classes

- userInterface
- tenantInputScreen
- rentInputScreen
- expenseInputScreen
- tenant
- tenanList
- rentRecord
- rentRow
- annualReport
- expense
- expenseRecord

The following classes had aggregation relationship:

tenantList and tenant
rentRecord and rentRow
expenseRecord and expense

Other classes have dependency relationships. You find the dependency relationships among these classes by reviewing the classes below.

Here are the classes in C++

```
//landlord.h
//header file for landlord.cpp -- contains class declarations, etc.
#pragma warning (disable:4786)  //for set (microsoft only)
#include <iostream>
#include <vector>
#include <set>
```

```
#include <string>
#include <algorithm>          //for sort()
#include <numeric>            //for accumulate()
using namespace std;
//////////////////////////global methods//////////////////////
void getaLine(string& inStr);   // get line of text
char getaChar();              // get a character

//////////////////////////class tenant//////////////////////
class tenant
  {
  private:
  string name;    // tenant's name
  int aptNumber;  // tenant's apartment number
  // other tenant information (phone, etc.) could go here

  public:
  tenant(string n, int aNo);
  ~tenant();
  int getAptNumber();
  // needed for use in 'set'
  friend bool operator < (const tenant&, const tenant&);
  friend bool operator == (const tenant&, const tenant&);
  // for I/O
  friend ostream& operator << (ostream&, const tenant&);
};  // end class tenant
//////////////////////class compareTenants//////////////////
class compareTenants  //function object --  compares tenants
  {
  public:
  bool operator () (tenant*, tenant*) const;
  };

//////////////////////class tenantList//////////////////////
class tenantList
  {
  private:
  // set of pointers to tenants
  set<tenant*, compareTenants> setPtrsTens;
  set<tenant*, compareTenants>::iterator iter;
```

```
  public:
  ~tenantList();             // destructor (deletes tenants)
  void insertTenant(tenant*);  // put tenant on list
  int getAptNo(string);       // return apartment number
  void display();            // display tenant list
  };  // end class tenantList

////////////////////class tenantInputScreen//////////////////
class tenantInputScreen
  {
  private:
  tenantList* ptrTenantList;
  string tName;
  int aptNo;

  public:
  tenantInputScreen(tenantList* ptrTL) : ptrTenantList(ptrTL)
    { /* empty */ }
  void getTenant();
  };  //end class tenantInputScreen

////////////////////////class rentRow////////////////////////
// one row of the rent record: an address and 12 rent amounts
class rentRow
  {
  private:
  int aptNo;
  float rent[12];

  public:
  rentRow(int);              // 1-arg ctor
  void setRent(int, float);    // record rent for one month
  float getSumOfRow();        // return sum of rents in row
  // needed to store in 'set'
  friend bool operator < (const rentRow&, const rentRow&);
  friend bool operator == (const rentRow&, const rentRow&);
  // for output
  friend ostream& operator << (ostream&, const rentRow&);
  };  // end class rentRow

//////////////////////////////////////////////////////////////
```

```
class compareRows  //function object --  compares rentRows
  {
  public:
    bool operator () (rentRow*, rentRow*) const;
  };

////////////////////////class rentRecord//////////////////////
class rentRecord
  {
  private:
  // set of pointers to rentRow objects (one per tenant)
  set<rentRow*, compareRows> setPtrsRR;
  set<rentRow*, compareRows>::iterator iter;

  public:
  ~rentRecord();
  void insertRent(int, int, float);
  void display();
  float getSumOfRents();       // sum all rents in record
  };  // end class rentRecord

////////////////////////class rentInputScreen/////////////////
class rentInputScreen
  {
  private:
  tenantList* ptrTenantList;
  rentRecord* ptrRentRecord;

  string renterName;
  float rentPaid;
  int month;
  int aptNo;

  public:
  rentInputScreen(tenantList* ptrTL, rentRecord* ptrRR) :
                ptrTenantList(ptrTL), ptrRentRecord(ptrRR)
    { /*empty*/ }
  void getRent();          //rent for one tenant and one month
  };  // end class rentInputScreen

////////////////////////////class expense//////////////////////
```

```
class expense
  {
  public:
  int month, day;
  string category, payee;
  float amount;
  expense()
    { }
  expense(int m, int d, string c, string p, float a) :
       month(m), day(d), category(c), payee(p), amount(a)
    { /*empty */ }
  // needed for use in 'set'
  friend bool operator < (const expense&, const expense&);
  friend bool operator == (const expense&, const expense&);
  // needed for output
  friend ostream& operator << (ostream&, const expense&);
  };  // end class expense
//////////////////////////////////////////////////////////////
class compareDates  //function object--compares expenses
  {
  public:
    bool operator () (expense*, expense*) const;
  };
//////////////////////////////////////////////////////////////
class compareCategories  //function object--compares expenses
  {
  public:
    bool operator () (expense*, expense*) const;
  };

//////////////////////////class expenseRecord//////////////////////
class expenseRecord
  {
  private:
  // vector of pointers to expenses
  vector<expense*> vectPtrsExpenses;
  vector<expense*>::iterator iter;

  public:
  ~expenseRecord();
  void insertExp(expense*);
```

```
  void display();
  float displaySummary();      // used by annualReport
  };  // end class expenseRecord

//////////////////////class expenseInputScreen//////////////////
class expenseInputScreen
  {
  private:
  expenseRecord* ptrExpenseRecord;

  public:
  expenseInputScreen(expenseRecord*);
  void getExpense();
  };  // end class expenseInputScreen

//////////////////////class annualReport//////////////////////
class annualReport
  {
  private:
  rentRecord* ptrRR;
  expenseRecord* ptrER;
  float expenses, rents;

  public:
  annualReport(rentRecord*, expenseRecord*);
  void display();
  };  // end class annualReport

//////////////////////class userInterface//////////////////////
class userInterface
  {
  private:
  tenantList*        ptrTenantList;
  tenantInputScreen*   ptrTenantInputScreen;
  rentRecord*        ptrRentRecord;
  rentInputScreen*     ptrRentInputScreen;
  expenseRecord*      ptrExpenseRecord;
  expenseInputScreen*  ptrExpenseInputScreen;
  annualReport*       ptrAnnualReport;
  char ch;
```

```
  public:
  userInterface();
  ~userInterface();
  void interact();
  };  // end class userInterfac
//////////////////////////end file landlord.h//////////////////////
```