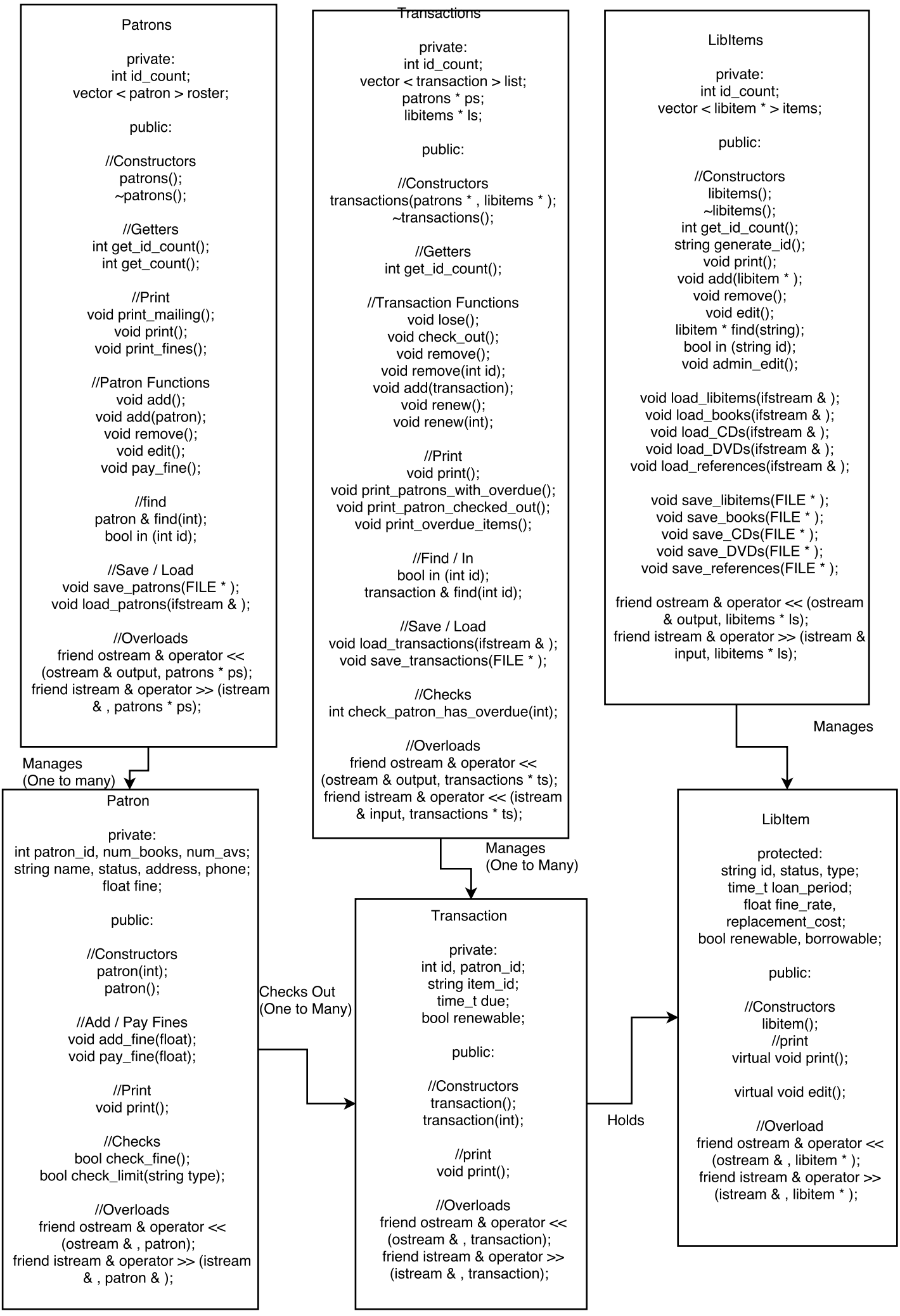
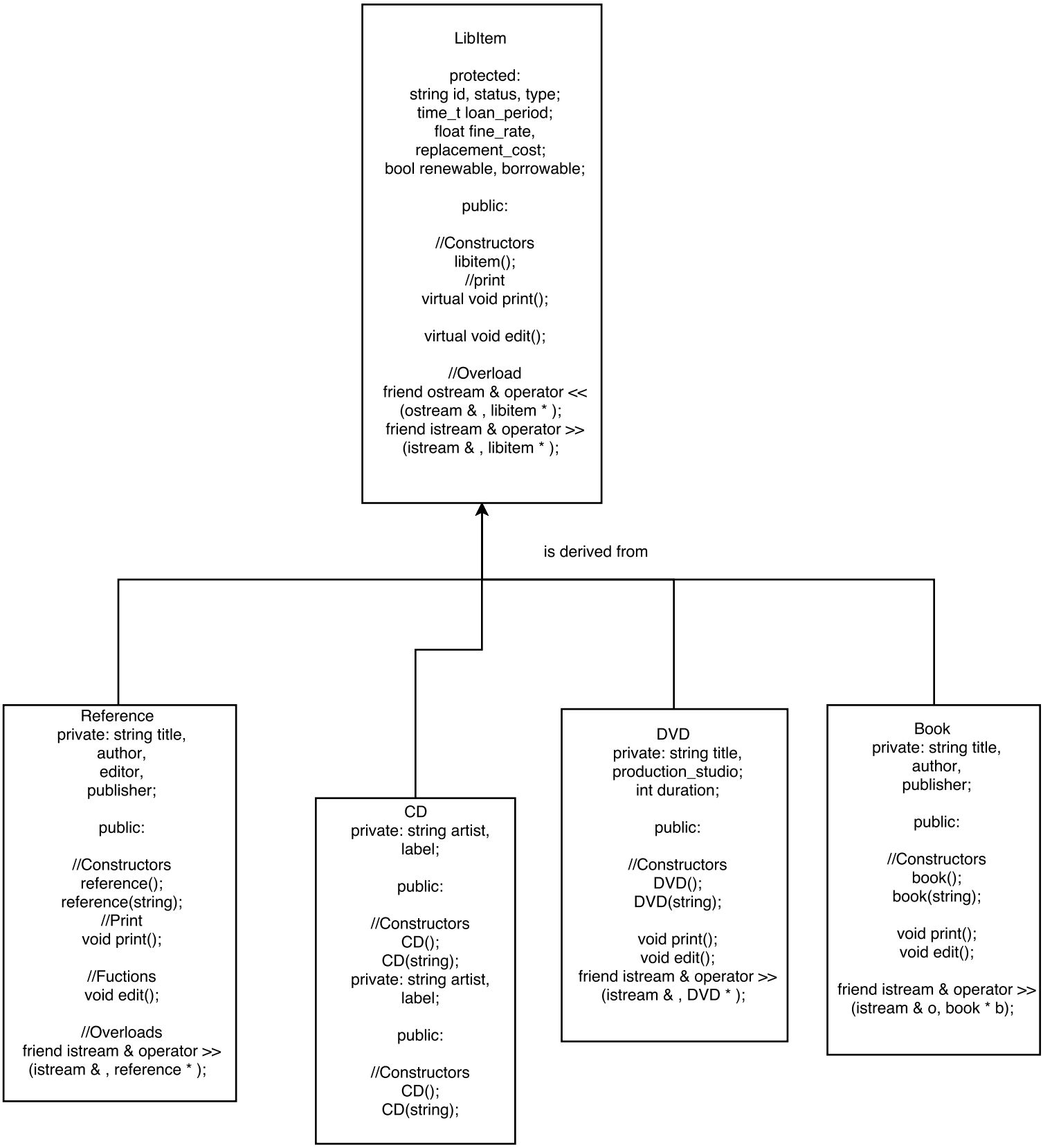

Patrons
private:
int id_count;
vector < patron > roster;
public:
//Constructors
patrons();
~patrons();
//Getters
int get_id_count();
int get_count();
//Print
void print_mailing();
void print();
void print_fines();
//Patron Functions
void add();
void add(patron);
void remove();
void edit();
void pay_fine();
//find
patron & find(int);
bool in (int id);
//Save / Load
void save_patrons(FILE *);
void load_patrons(ifstream &);
//Overloads
friend ostream & operator <<
(ostream & output, patrons * ps);
friend istream & operator >> (istream
& , patrons * ps);
Transactions
private:
int id_count;
vector < transaction > list;
patrons * ps;
libitems * ls;
public:
//Constructors
transactions(patrons * , libitems *);
~transactions();
//Getters
int get_id_count();
//Transaction Functions
void lose();
void check_out();
void remove();
void remove(int id);
void add(transaction);
void renew();
void renew(int);
//Print
void print();
void print_patrons_with_overdue();
void print_patron_checked_out();
void print_overdue_items();
//Find / In
bool in (int id);
transaction & find(int id);
//Save / Load
void load_transactions(ifstream &);
void save_transactions(FILE *);
//Checks
int check_patron_has_overdue(int);
//Overloads
friend ostream & operator <<
(ostream & output, transactions * ts);
friend istream & operator << (istream
& input, transactions * ts);
LibItems
private:
int id_count;
vector < libitem * > items;
public:
//Constructors
libitems();
~libitems();
int get_id_count();
string generate_id();
void print();
void add(libitem *);
void remove();
void edit();
libitem * find(string);
bool in (string id);
void admin_edit();
void load_libitems(ifstream &);
void load_books(ifstream &);
void load_CDs(ifstream &);
void load_DVDs(ifstream &);
void load_references(ifstream &);
void save_libitems(FILE *);
void save_books(FILE *);
void save_CDs(FILE *);
void save_DVDs(FILE *);
void save_references(FILE *);
friend ostream & operator << (ostream
& output, libitems * ls);
friend istream & operator >> (istream &
input, libitems * ls);
Manages
(One to many)
Patron
private:
int patron_id, num_books, num_avs;
string name, status, address, phone;
float fine;
public:
//Constructors
patron(int);
patron();
//Add / Pay Fines
void add_fine(float);
void pay_fine(float);
//Print
void print();
//Checks
bool check_fine();
bool check_limit(string type);
//Overloads
friend ostream & operator <<
(ostream & , patron);
friend istream & operator >> (istream
& , patron &);
Manages
(One to Many)
Transaction
private:
int id, patron_id;
string item_id;
time_t due;
bool renewable;
public:
//Constructors
transaction();
transaction(int);
//print
void print();
//Overloads
friend ostream & operator <<
(ostream & , transaction);
friend istream & operator >>
(istream & , transaction);
Checks Out
(One to Many)
Manages
LibItem
protected:
string id, status, type;
time_t loan_period;
float fine_rate,
replacement_cost;
bool renewable, borrowable;
public:
//Constructors
libitem();
//print
virtual void print();
virtual void edit();
//Overload
friend ostream & operator <<
(ostream & , libitem *);
friend istream & operator >>
(istream & , libitem *);
Holds


LibItem
protected:
string id, status, type;
time_t loan_period;
float fine_rate,
replacement_cost;
bool renewable, borrowable;
public:
//Constructors
libitem();
//print
virtual void print();
virtual void edit();
//Overload
friend ostream & operator <<
(ostream & , libitem *);
friend istream & operator >>
(istream & , libitem *);
is derived from
Reference
private: string title,
author,
editor,
publisher;
public:
//Constructors
reference();
reference(string);
//Print
void print();
//Fuctions
void edit();
//Overloads
friend istream & operator >>
(istream & , reference *);
CD
private: string artist,
label;
public:
//Constructors
CD();
CD(string);
private: string artist,
label;
public:
//Constructors
CD();
CD(string);
DVD
private: string title,
production_studio;
int duration;
public:
//Constructors
DVD();
DVD(string);
void print();
void edit();
friend istream & operator >>
(istream & , DVD *);
Book
private: string title,
author,
publisher;
public:
//Constructors
book();
book(string);
void print();
void edit();
friend istream & operator >>
(istream & o, book * b);