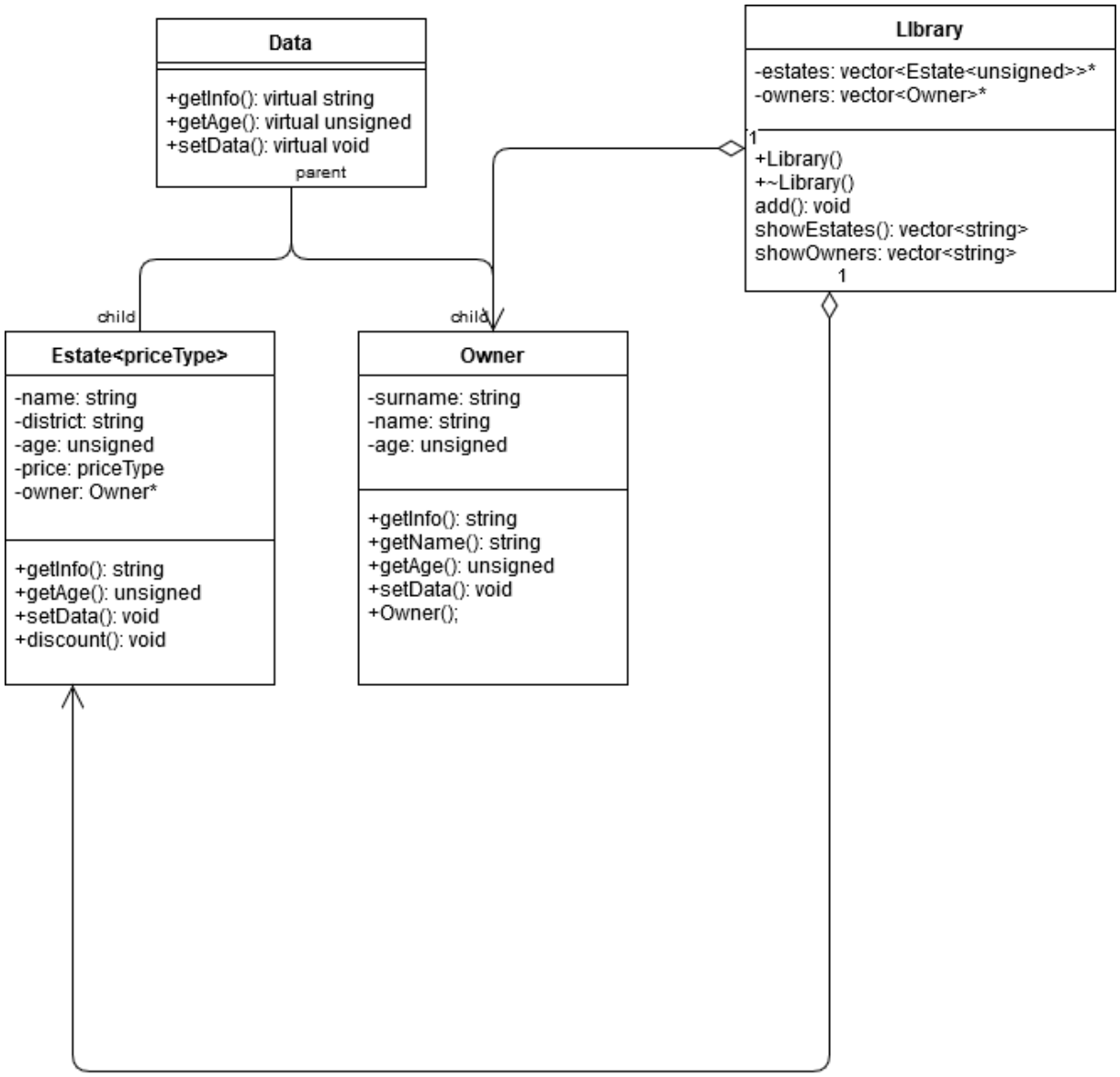
Project 3

Class diagram:



Estate:

- std::string getInfo() – returns a string containing the object's data in the following format: *„name* distr: *distrct* age: *age* price: *price* owner: *owner's name"*

- void setData(std::string nameFeed, std::string districtFeed, unsigned ageFeed, int price = 0, Owner *ownerFeed = 0)

- void setData(unsigned ageFeed)

- void discount(priceType amount) – decreases the price of the estate by the given amount. Throws std::logic_error("negative price exception")upon an attempt to make the price negative

Owner:

- std::string getInfo() – returnsa the object's data in the following format: *„surname name* age: *age"*

- std::string getName() – returns a string in the following format: *„surname name"*

- void setData(std::string nameFeed = "Johnny", std::string surnameFeed = "Unknown", unsigned ageFeed = 1);

- void setData(unsigned ageFeed);

- Owner(std::string nameFeed = "Johnny", std::string surnameFeed = "Unknown", unsigned ageFeed = 1);

Library:

- void add(Estate<unsigned> estateFeed)

- std::vector<std::string> showEstates() – returns a vector that contains strings formatted as in Estate::getInfo()

- void add(Owner estateFeed)

- std::vector<std::string> showOwners() – returns a vector that contains strings formatted as in Owner::getInfo()

functions:

- Owner operator+ (Owner person, unsigned number) - increases the owner's age by the number given

- void increaseAge(Data* info, unsigned number)

- print(std::string s)

Project 2

Clinic:

```
void mock();
Clinic& operator+=(Doctor &doc);
Clinic& operator-=(Doctor &doc);
Clinic& operator+=(Patient &pat);
Clinic& operator-=(Patient &pat);
void printDoctors();
void printDoctors(int spec);
void printPatients();
bool detailsOn(std::string const &name);
void diagnose(std::string const &name, std::string const &diagnosis);
void prescribe(std::string const &name, std::string const &prescription);
```

Doctor:

```
enum class Spclty {GP, ophthalmologist, dentist, pediatrist, unspecified};
Doctor(): specialty(Spclty::unspecified), name("unknown") {}
Doctor(std::string const &nam): specialty(Spclty::unspecified), name(nam) {}
Doctor(std::string const &nam, Spclty spec): specialty(spec), name(nam) {}
Spclty getSpclty() {return specialty;}
std::string getName() const {return name;}
bool operator==(Doctor &other);
friend std::ostream& operator<< (std::ostream& out, const Doctor &doc);
```

Patient:

```
Patient()
Patient(std::string nam);
Patient& operator= (Patient other);
Patient& diagnose (std::string const &diagnosis);
Patient& prescribe (std::string const &prescription);
```

```
std::string getName() const {return name;}
bool operator==(Patient &other);
friend std::ostream& operator<< (std::ostream& out, const Patient &pat);
Interface:
void menu();
void test() {clinic.mock();}
Clinic& addDoctor();
Clinic& addPatient();
template <typename T> Clinic& rm()
void details();
void diagnose();
void prescribe();
inline void checkInput();
```

Project 1

SetOfInts:

int getFirst();

bool empty() const;

SetOfInts& operator+= (SetOfInts other);

SetOfInts& operator+= (const int other);

SetOfInts& operator-= (SetOfInts other);

SetOfInts& operator-= (const int other);

SetOfInts& increment();

SetOfInts& decrement();

void show() const;

void show(std::string message) const;