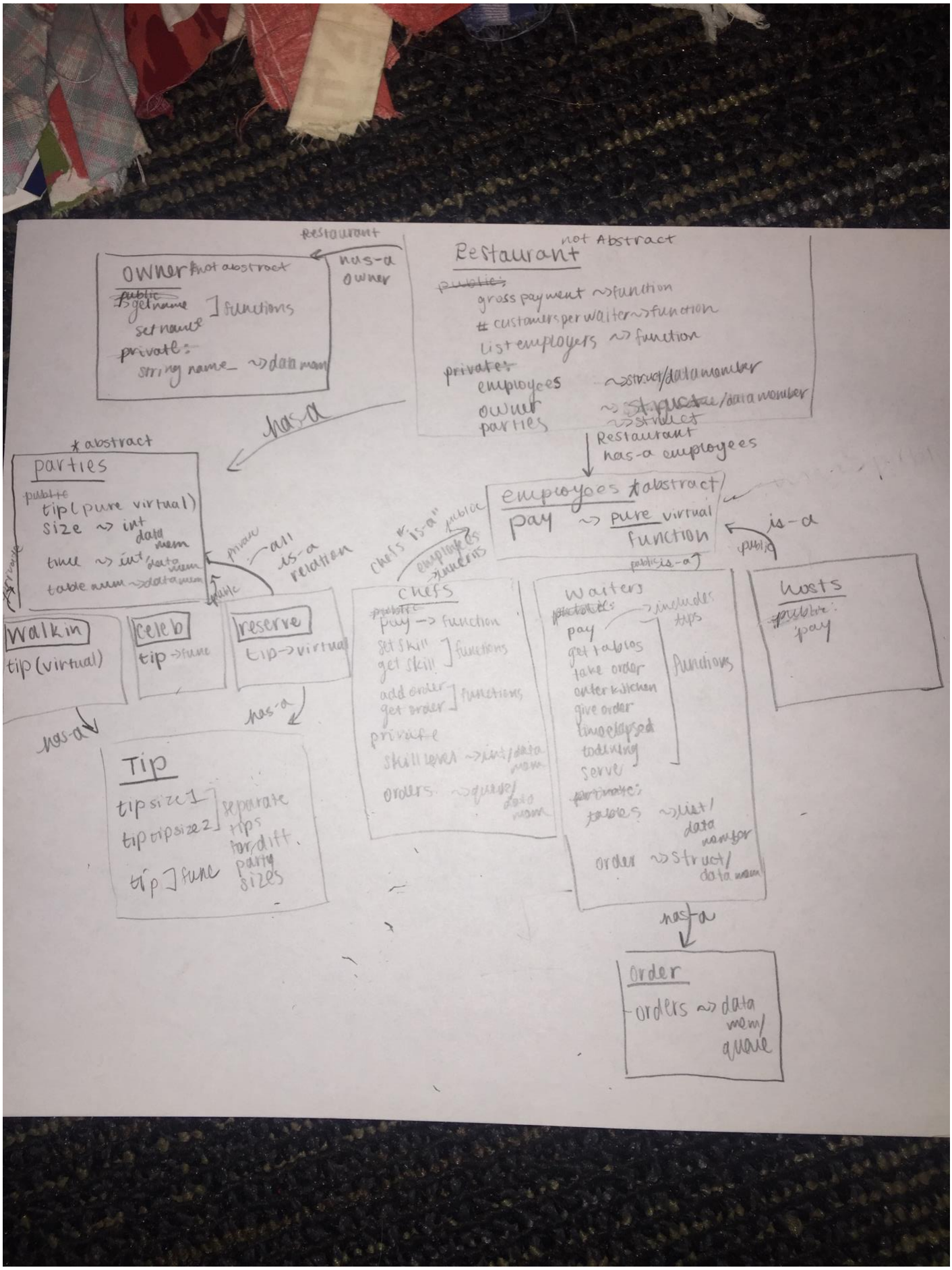

Restaurant
has-a Owner
Owner not abstract
getname
setname
functions
private:
string name_ → data mem
Restaurant not Abstract
gross payment → function
customers per waiter → function
list employers → function
private:
employees → struct/data member
owner → struct/data member
parties → struct
has-a
Restaurant has-a employees
* abstract
parties
public
tip (pure virtual)
size → int data mem
time → int data mem
table num → data mem
private
all is-a relation
employees * abstract
pay → pure virtual function
is-a
public
chefs "is-a"
public employees struct
public is-a
Walkin
tip (virtual)
Celeb
tip → func
Reserve
tip → virtual
chefs
public:
pay → function
set skill
get skill
functions
add order
get order
functions
private
skill level → int/data mem
orders → queue/data mem
Waiters
public:
pay → includes tips
get tables
take order
enter kitchen
give order
time elapsed
training
serve
functions
private:
tables → list/data member
order → struct/data mem
Hosts
public:
pay
has-a
has-a
has-a
Tip
tip size 1
tip size 2
separate tips for diff. party sizes
tip] func
Order
orders → data mem/queue

Restaurant is not an abstract class because it does not have any pure virtual functions

Parties and employees are abstract classes because they contain pure virtual functions

tip in the parties class is a pure virtual function because it has no implementation in that class, rather it is defined in the type sub-classes of parties

tip is a virtual function in walk-ins and reserved, but not in celeb because celeb has a constant tipping algorithm while walk-in and reserve tip amounts vary based on the size of the party

pay in the employees class is a pure virtual function because each pay type in the employees sub-classes are unique

is-a relationships

Chess is-a employees

waiters is-a employees

hosts is-a employees

walk-in is-a parties
celeb is-a parties
reserve is-a parties

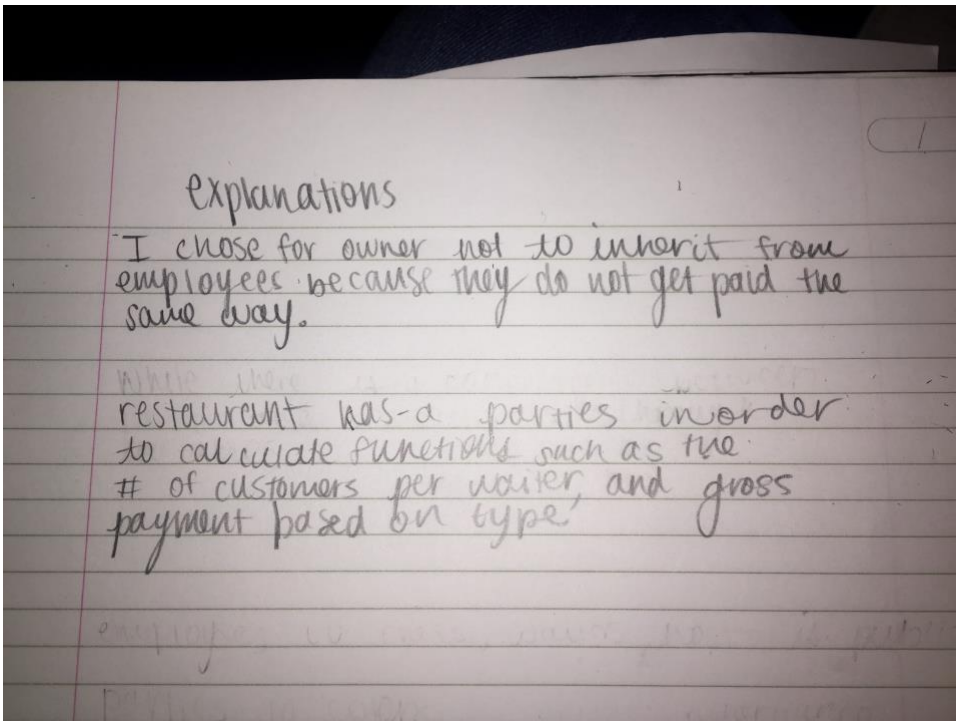
has-a relationships

Restaurant has-a owner ← string
Restaurant has-a employees
↳ struct *obj;

waiters has-a Order
↳ queue

walk in has-a tip ← double
reserved has-a tip ← double

restaurant → has-a parties.
↳ obj1, struct



Explanations

I chose for owner not to inherit from employees because they do not get paid the same way.

restaurant has a parties in order to calculate functions such as the # of customers per waiter, and gross payment based on type.