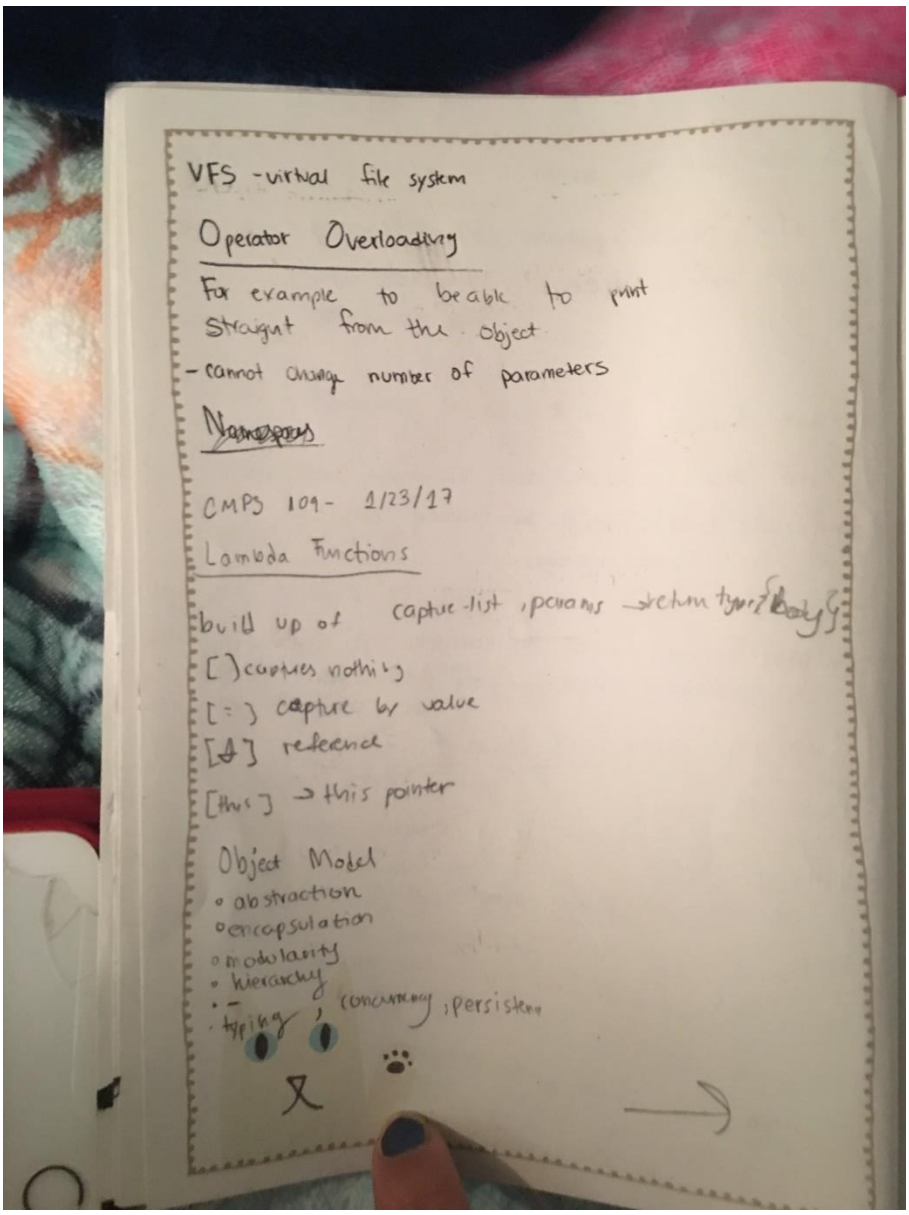


VFS - virtual file system

Operator Overloading

For example to be able to print straight from the object

- cannot change number of parameters

~~Namespaces~~

CMPS 109 - 1/23/17

Lambda Functions

built up of capture-list (params) -> return type {body};

[] captures nothing

[=] capture by value

[&] reference

[this] -> this pointer

Object Model

- abstraction
- encapsulation
- modularity
- hierarchy
- typing, concurrency, persistence

- constructors - create and set the object
- destructors - destroy / delete / free
- modifiers - altering
- selectors - access
- iterators - allows access at some time.

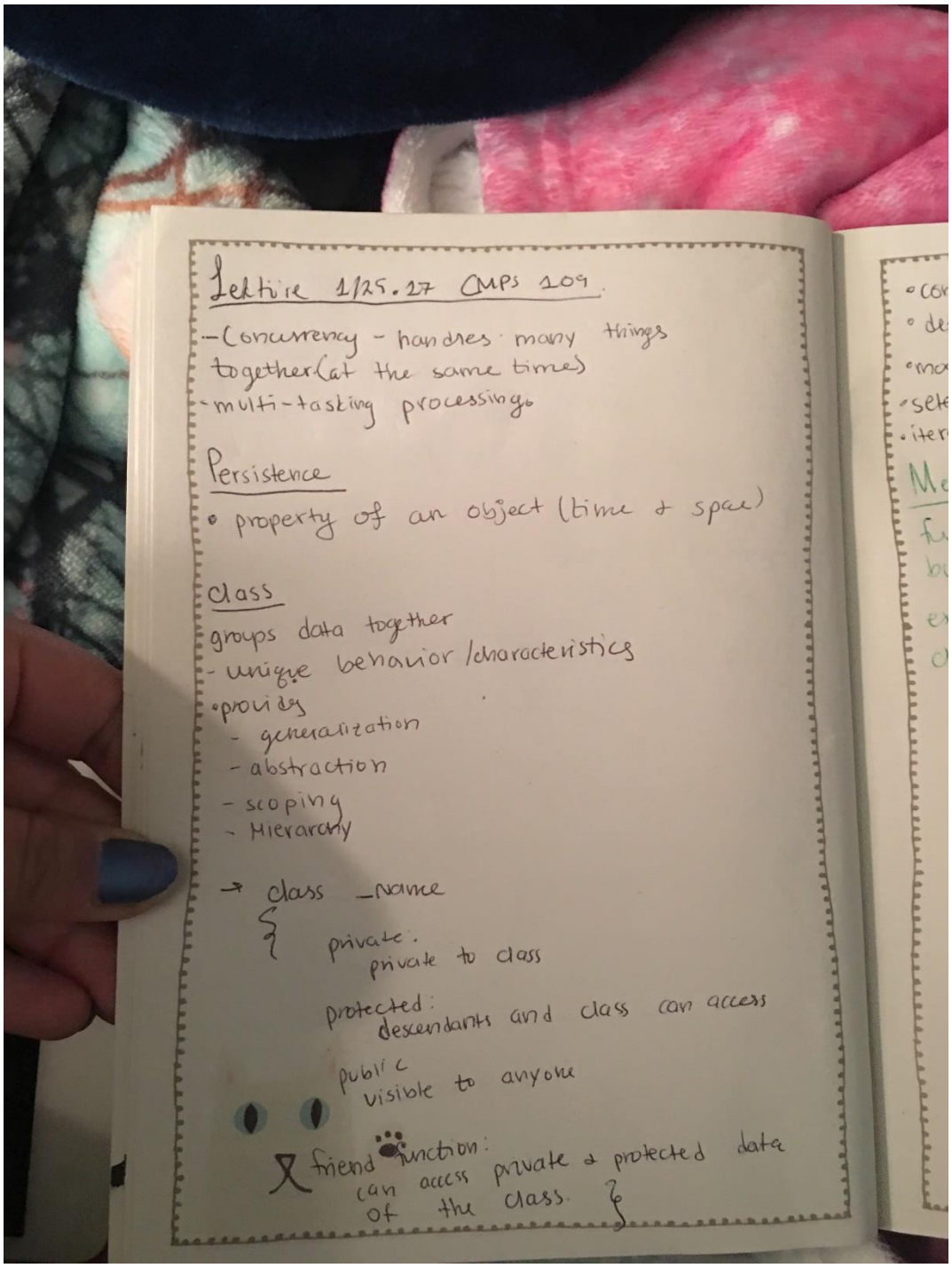
Methods Overloading

functions with same name but different set of parameters

example:

```
class Home{
    private:
    public:
    Home();
    void number(){cout<<"Number"<<endl;}
    ~Home();
};
class Number : public Home{
    public:
    Number();
    void number(){cout<<"Number"<<endl;}
    ~Number();};
int main(){
    Home x = new Home();
    Home x = new Number();
}
```

1 -> number(); 2 -> number();



Lecture 1/25.27 CMPS 109

- Concurrency - handles many things together (at the same time)
- multi-tasking processing

Persistence

- property of an object (time & space)

Class

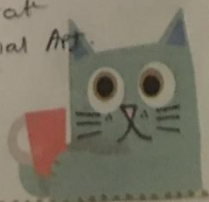
groups data together
- unique behavior / characteristics
- provides
  - generalization
  - abstraction
  - scoping
  - Hierarchy

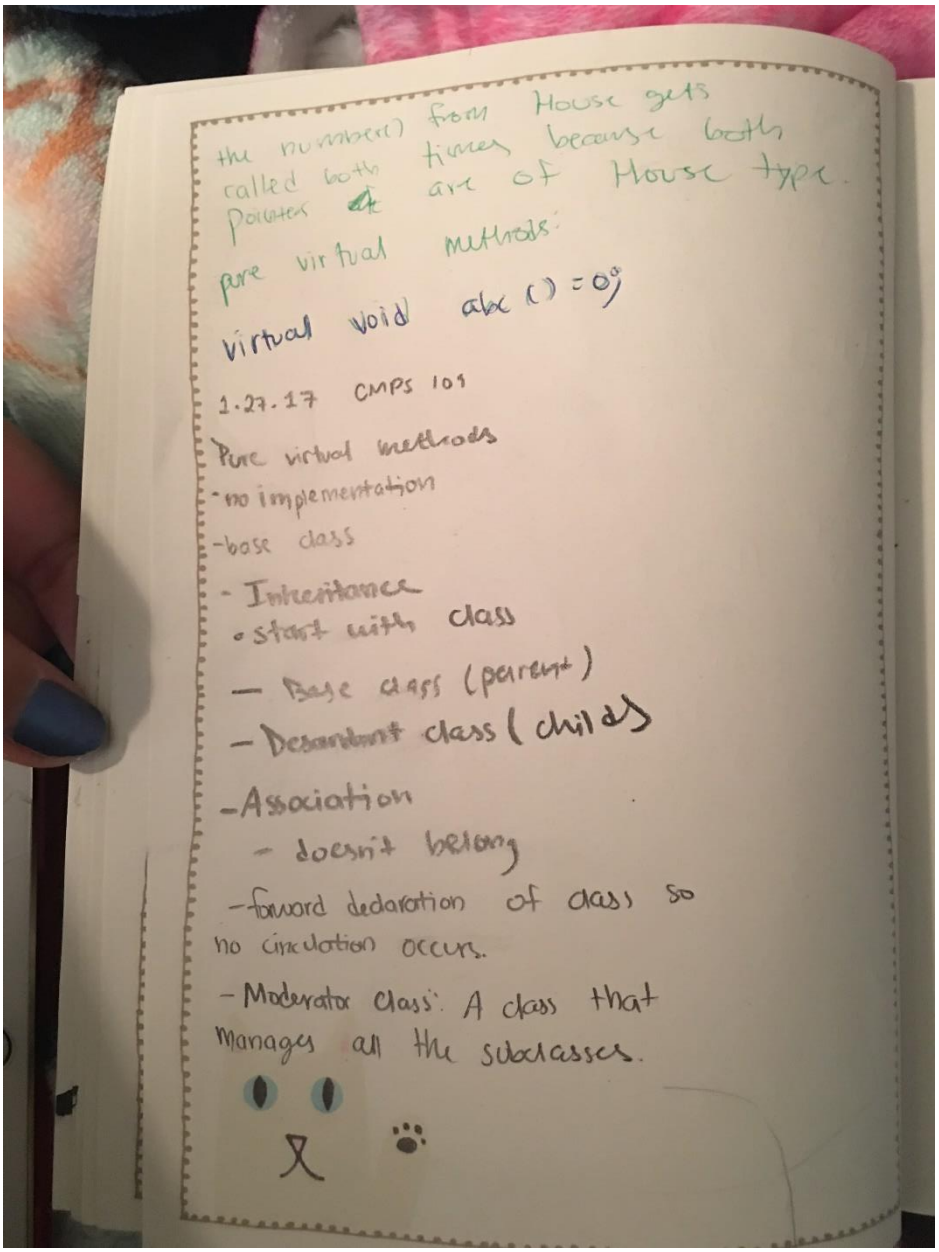
→ class _name
{
  private:
    private to class
  protected:
    descendants and class can access
  public
    visible to anyone
  friend function:
    can access private & protected data of the class.
}

- look for similarities
- make sure to understand, analyze, distinguish objects.
- needs to define object and se how they communicate between each other.
- write specification of the program.
- Encapsulation
- complementary concepts
- if applying abstraction right,
- process of compartmentalizing

elements of abstraction.

- Modularity
- sounds like abstraction but in a different way.
- Java has packages, C++ uses objects and classes, namespaces (used in modularization)
- can be compiled seperately.
- modules have to be seperate and be invoked using external API
- decomposition.





the number) from House gets called both times because both pointers are of House type.

pure virtual methods:

virtual void abc () = 0;

1.27.17 CMPS 101

Pure virtual methods
- no implementation
- base class
- Inheritance
  - start with class
  - Base class (parent)
  - Descendant class (child)
- Association
  - doesn't belong
- foward declaration of class so no circulation occurs.
- Moderator class: A class that manages all the subclasses.