Inheritance

- the __Base__ (derived/base) class is the __Parent__(parent/child)
- the __Derived__ (derived/base) class is the __Child__(parent/child)
- a __Child__ (parent/child) has an is-a relationship with the __Parent__ (parent/child)

(More) Concretely

- the __Animal__ class is the __Parent__
- the __Mamal__ class is the __Child__
- a __Turtle__ is a(n) __Reptile__

What is not inherited?

anything defined under private.
Constructor destructor and overridden
classes. Child-To-Parent references.

What is inherited?

Public and Protected

How does privacy interact with inheritance?

Anything that is private cant be
accessed through child classes

Animal

```
class Animal {
public:
    Animal(string sound): sound_(sound) {}
    string MakeSound() {return sound_; }
    virtual int GetPower() {return 0; }
private:
    std::string sound_;
}
```

Reptile

```
class Reptile : public Animal {
public:
     Reptile(std::string sound):
     Animal(sound + "rawr") {}

     int GetPower() {return 2; }
}
```

Mammal

```
class Mammal : public Animal {
public:
     Mammal():
     Animal("fuzzy fuzz") {}
     int GetPower() {return 3; }
}
```

Turtle

```
class Turtle : public Reptile {
public:
     Turtle(): Reptile("turtle turtle") {}
     int GetPower() {return 7; }
}
```

```
// We could instantiate some Animals as follows:
Turtle t;
Mammal gopher;
Animal cow = new Animal("moo");

std::cout << t.MakeSound() << std::endl;
std::cout << gopher.MakeSound() << std::endl;
std::cout << cow->MakeSound() << std::endl;
```

What is the output of the above code?

turtle turtle rawr
fuzzy fuzz
moo

Would the below code work? why/why not?

```
std::vector<Animal> vec = {t, gopher, *(cow)};
```

No, because they need to be defined
as animal objects

Dynamic Dispatch

What is dynamic dispatch? How does it relate to the `virtual` keyword?

Dynamic dispatch is process that takes the child method through the virtual keyword.

```
// Now, let's instantiate some more objects as follows:
Animal * t2 = new Turtle();
Animal * m2 = new Mammal();
Animal * r2 = new Reptile("hiss");
```

Would the below code work? why/why not?

```
std::vector<Animal *> vec = {t2, m2, r2};
```

Answer:

Yes this works, no errors

What method(s) are called in the following code?

```
// which method is being called for these function calls?
for (int i = 0; i < vec.size(); i++) {
   std::cout << vec[i]->MakeSound() << std::endl;
}
```

method(s) called

animal -> MakeSound()

What method(s) are called in the following code?

```
// which method is being called for these function calls?
for (int i = 0; i < vec.size(); i++) {
   std::cout << vec[i]->GetPower() << std::endl;
}
```

method(s) called

It would run the child's version
It would run the one in child's version

What would happen if `GetPower()` had not been marked `virtual`?

It would return the parent function.