Felipe Lima - 109290055

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 ___base___ class is the ___parent___
- the ___derived___ class is the ___child___
- a ___child___ is a(n) ___parent___

What is not inherited?

Constructors
Desctructors
overloaded operators

What is inherited?

Methods and attributes
fields

How does privacy interact with inheritance?

public: everyone has access

private: only the actual class has access

protected: child classes and actual class have access

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 fuzzy
moo

Would the below code work? why/why not?

no, they have different types

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

Dynamic Dispatch

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

"Dynamic dispatch means that the binding of the method is determined at run time depending on the type of the object pointed to by the object call." - condor depaul

Calls the respective object that is pointed by the object call - (Most derived version)

virtual keyword is used to dynamic dispatch is used by c++

```
// 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, they are all "Animal" type.

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

Makesoud in Animal all three times.

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

In each respective classes because of the virtual keyword

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

It would call GetPower() in "Animal" class.