Inheritance

- the _____ (derived/base) class is the _____(parent/child)
- the _____ (derived/base) class is the _____(parent/child)
- a _____ (parent/child) has an is-a relationship with the _____ (parent/child)

(More) Concretely

- the _____ class is the _____
- the _____ class is the _____
- a _____ is a(n) _____

What is not inherited?

What is inherited?

How does privacy interact with inheritance?

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

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; }
}
```

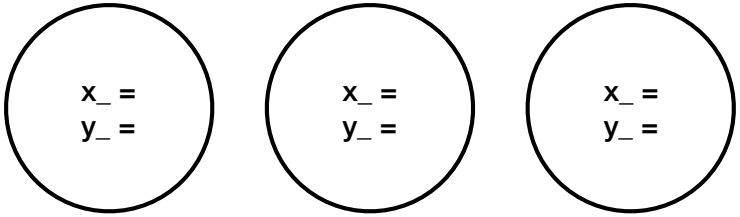
```
Turtle t;
Mammal m;
Animal * a = new Turtle()

// which method is being called for these function calls?
std::cout << t.MakeSound() << std::endl;
std::cout << m.MakeSound() << std::endl;
std::cout << a->MakeSound() << std::endl;


// what about for these ones?
std::cout << t.GetPower() << std::endl;
std::cout << m.GetPower() << std::endl;
std::cout << a->GetPower() << std::endl;
```

Non static fields

Point.h

```
int x_;
int y_;
```

Point instances



Static fields

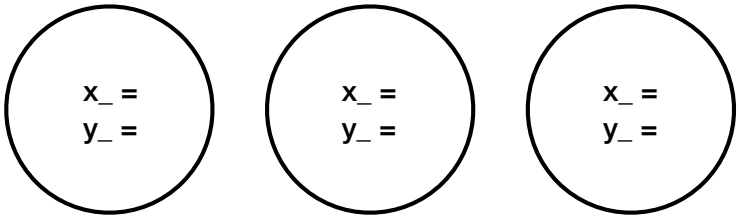
Point.h

```
static int x_;
static int y_;
```

Point.cpp

```
int Point::x_ =      ;
int Point::y_ =      ;
```

Point instances



Non static methods

Point.h

```
double Distance(const Point & other) const;
```

Static methods

Point.h

```
static double Distance(const Point & p1, const Point & p2);
```