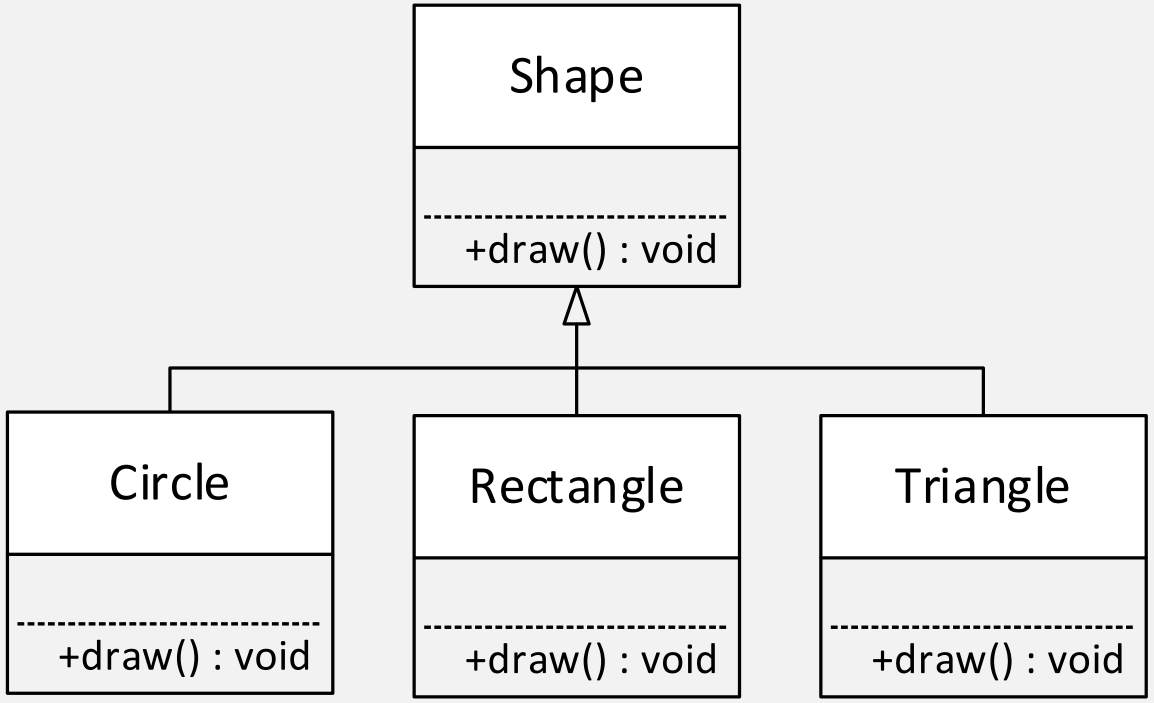
PURE VIRTUAL FUNCTIONS AND ABSTRACT CLASSES

And their connection to polymorphism

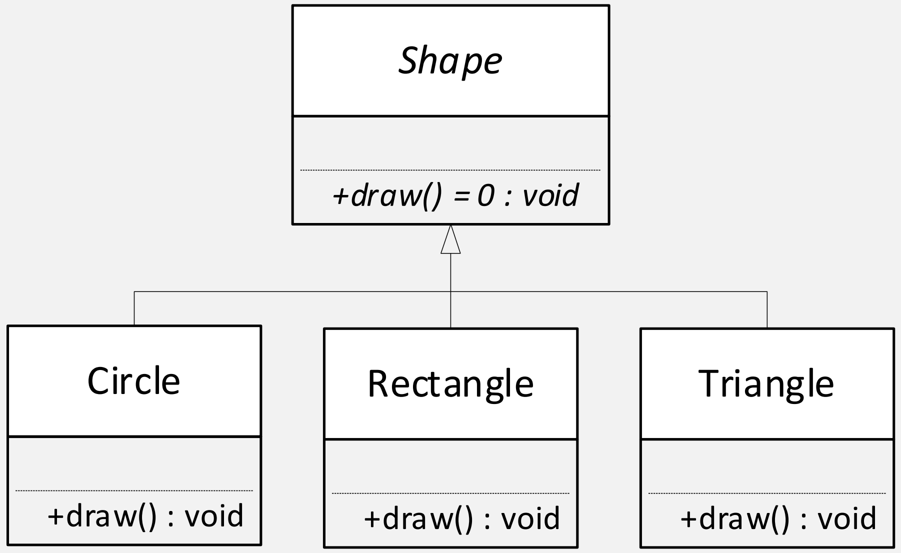
Delroy A. Brinkerhoff

POLYMORPHISM AND ALGORITHMS



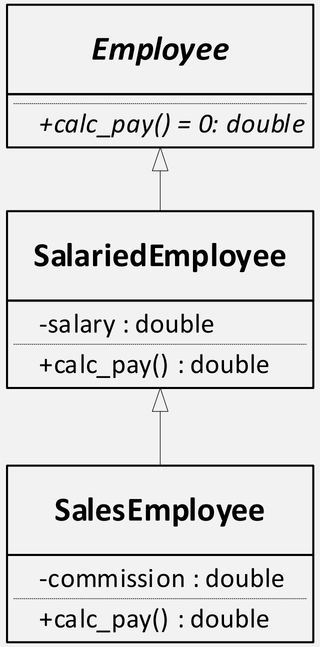
PURE VIRTUAL FUNCTIONS MAKES A CLASS ABSTRACT

- Pure virtual functions
  - Don’t have a body (prototype = 0)
  - <u>Must</u> be overridden in all subclasses
- pure virtual functions makes a class abstract
- Abstract classes cannot be instantiated
- Abstract classes and functions are denoted with italic characters in the UML



POLYMORPHISM AND ABSTRACTION

- Polymorphism does not require pure virtual functions or abstract classes, but they are often used together to create general programming solutions.
- Calculating pay
    - salaried: salary / 24
    - sales: salary / 24 + commission
- `Employee e = new __Employee;`
- `e->calc_pay();`



CHAINING FUNCTION CALLS

- Employee

```
virtual double calc_pay() = 0;
```

- SalariedEmployee

```
double calc_pay()
{
    return salary / 24;
}
```

- SalesEmployee

```
double calc_pay()
{
    return SalariedEmployee::calc_pay() + commission;
}
```

ABSTRACT CLASSES CAN

- not be instantiated
- be a superclass (i.e., a parent or base class)
- be used as a datatype (`Employee* e;`)
- participate in (i.e., be the target of) an upcast (`e = new SalesEmployee;`)
- participate in polymorphism
- have concrete features (both variables and functions) that can be inherited by subclasses