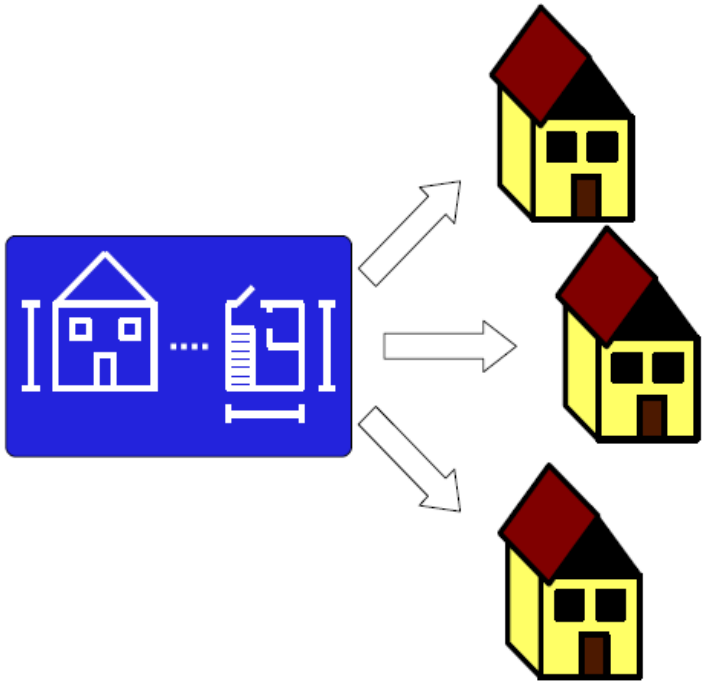
Smalltalk

- Creating a class (Account)
- Creating a subclass (Savings, Checking)

Class and Instances

http://stephane.ducasse.free.fr/FreeBooks/Gnu/ProgrammingUsingGnuSmalltalk.pdf



We can think of the blue print as a House class and the actual houses as the instances of it.

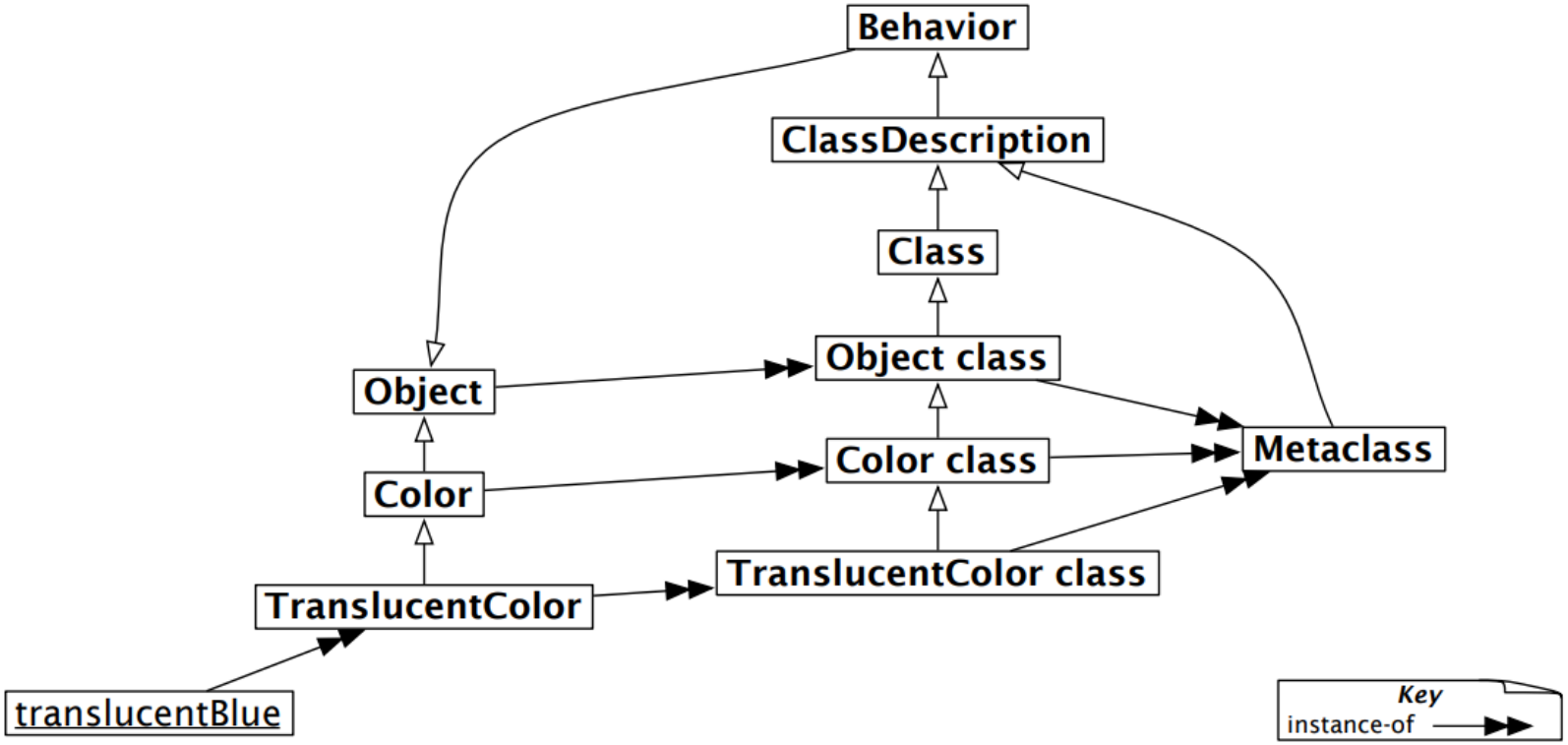
Smalltalk Rules

- **Rule 1**. Everything is an object.
- **Rule 2**. Every object is an instance of a class.
- **Rule 3**. Every class has a superclass.
- **Rule 4**. Everything happens by sending messages.
- **Rule 5**. Method lookup follows the inheritance chain.

Smalltalk Rules

- **Rule 6**. Every class is an instance of a metaclass.
- **Rule 7**. The metaclass hierarchy parallels the class hierarchy.
- **Rule 8**. Every metaclass inherits from Class and Behavior.
- **Rule 9**. Every metaclass is an instance of Metaclass.
- **Rule 10**. The metaclass of Metaclass is an instance of Metaclass.

Classes and Metaclasses
An Example



Credit: Pharo By Example

Creating a class

```
Object subclass: Account [
    | balance |
    <comment:
        'I represent a place to deposit
and withdraw money'>
]
```

Add a class method

```
Account class extend [
      new [
          | r |
          <category: 'instance creation'>
          r := super new.
          r init.
          ^r
      ]
  ]
```

Add an instance method

```
Account extend [
    init [
        <category: 'initialization'>
        balance := 0
    ]
]
```

Add another instance method

```
Account extend [
        printOn: stream [
            <category: 'printing'>
            super printOn: stream.
            stream nextPutAll: ' with
balance: '.
            balance printOn: stream
        ]
    ]
```

Add more instance methods

```
Account extend [
        spend: amount [
            <category: 'moving money'>
            balance := balance - amount
        ]
        deposit: amount [
            <category: 'moving money'>
            balance := balance + amount
        ]
    ]
```

Put everything together and testing

- See `account.st`
- Testing: `gst account.st -`
- Testing:

```
gst
st>FileStream fileIn: 'account.st'
```

Creating a subclass

```
Account subclass: Savings [
    | interest |

    init [
        <category: 'initialization'>
        interest := 0.
        ^super init
    ]
```

Creating a subclass (2)

```
    interest: amount [
        interest := interest + amount.
        self deposit: amount
    ]
    clearInterest [
        | oldinterest |
        oldinterest := interest.
        interest := 0.
        ^oldinterest
    ]
]
```

Creating another subclass

- See `checking.st`

```
Account subclass: Checking [
    | checknum checksleft history |

]
```

Method with a block

```
checksOver: amount do: aBlock [
    history keysAndValuesDo: [:key :value |
        (value > amount)
            ifTrue: [aBlock value: key]
    ]
]
```

Method with a block (2)

```
 checksOver: amount do: aBlock [
      | chosen |
      chosen := history select: [:amt| amt > amount].
      chosen keysDo: aBlock
]
```

Testing (`test.st`)

```
mycheck := Checking new.
mycheck deposit: 250.
mycheck newChecks: 100 count: 40.
mycheck writeCheck: 10.
mycheck writeCheck: 52.
mycheck writeCheck: 15.
mycheck printChecks.
(mycheck check: 101) printNl.
mycheck checksOver: 1 do: [:x | x printNl].
mycheck checksOver: 17 do: [:x | x printNl].
mycheck checksOver: 200 do: [:x | x printNl].
```

References

- Creating classes: https://www.gnu.org/software/smalltalk/manual/gst.html#Creating-classes
- Creating subclasses: https://www.gnu.org/software/smalltalk/manual/gst.html#Creating-subclasses
- GNU Smalltalk Library Reference: https://www.gnu.org/software/smalltalk/manual-base/gst-base.html