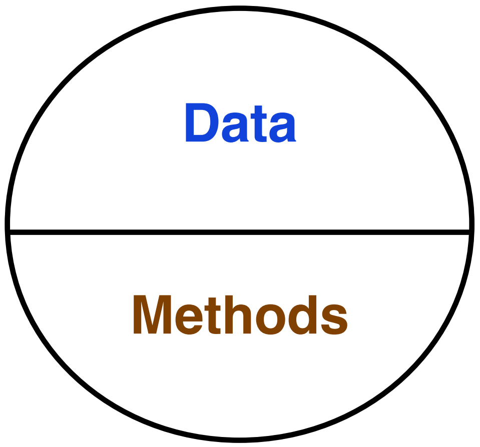
Inheritance

What is it all about?

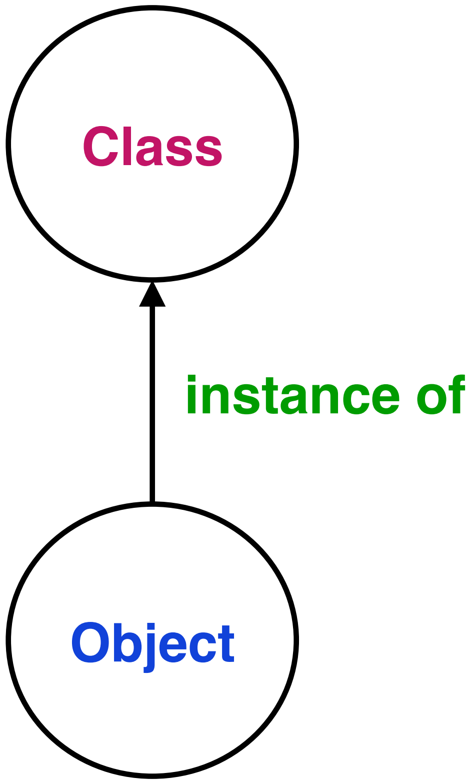
On Objects

- An **Object** is a collection of data and methods to operate on that data
  - » **Method is a procedure, function, operation**
- For a motor
  - » **turnOn   turnOff   setSpeed (someSpeed)**



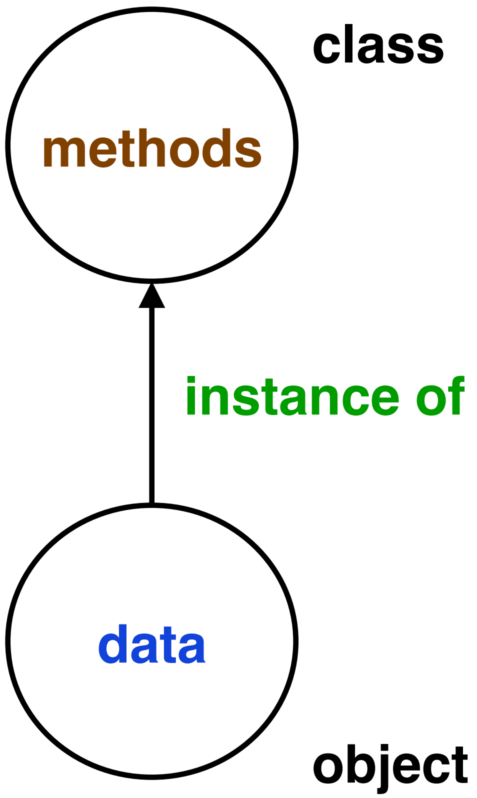
On Instances

- An object is an **instance** of a class
  - » **The class provides the template for the object**
- Template gives
  - » **Data types**
  - » **Methods**
- Can think of the object as having a copy of the methods and space for its own data



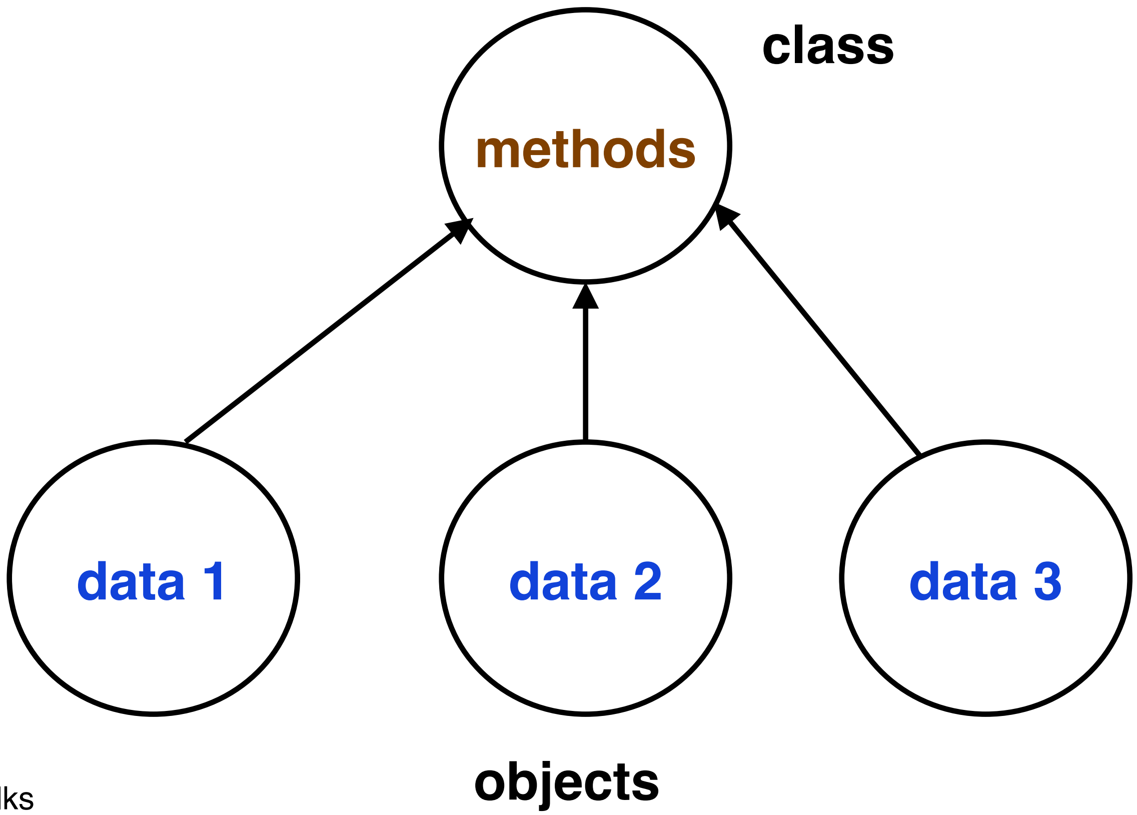
The Real Story on Space

- Only the data is unique to the object



The Real Story – 2

- Multiple Instances
  - » **Every object has its own data**
  - » **Objects share methods**



Message Definition

- A message is equivalent to a procedure call
- It is the way objects communicate with each other and request work to be done
- We think of the objects as being active
- Assume **motor** is an instance of the class **MOTOR**
  - > **Then typical expressions are:**

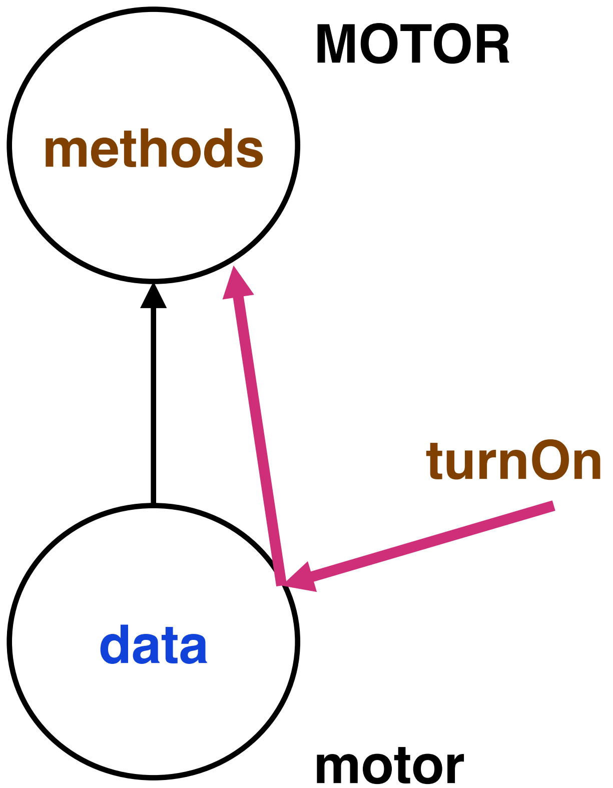
```
motor . turnOn
motor . turnOff
motor . setSpeed ( 5 )
```

Message Routing

- **MOTOR** contains method **turnOn**
- The message **turnOn** is sent to
  the object **motor**

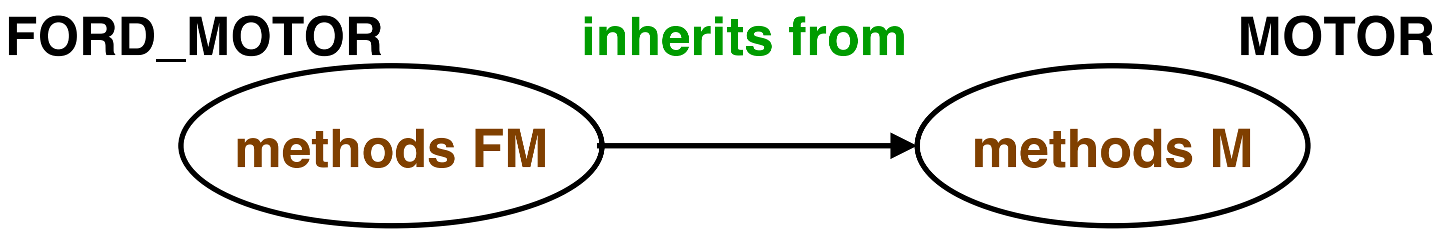
  motor . turnOn

- The data in the object is used by the method



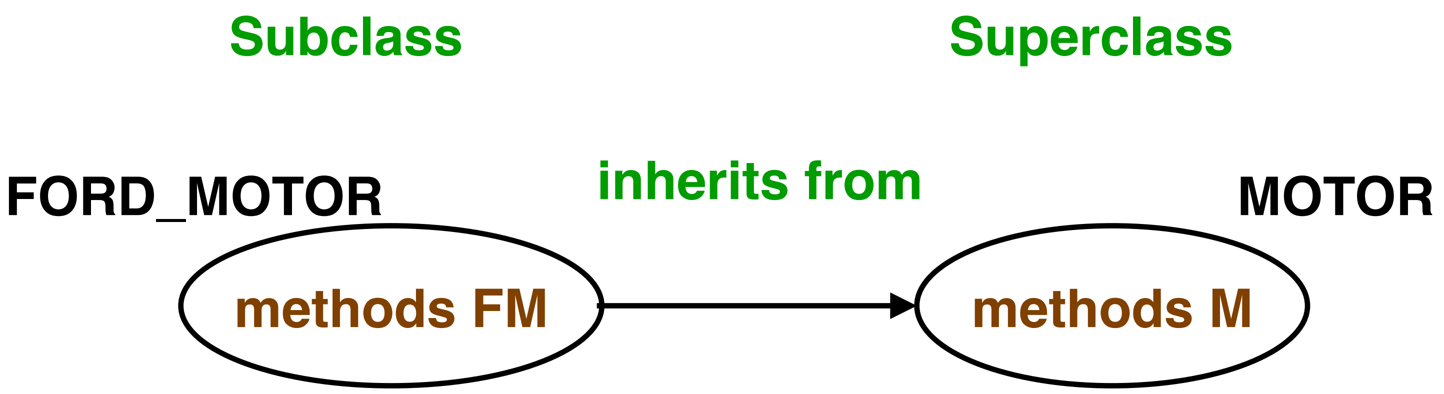
Definitions

- Inheritance
  - » A class can inherit some of its methods from another class
    - – methods FM methods M
  - > It can **define** its own methods – **add methods**
  - > It can **redefine** the methods of the class it is inheriting from – **change semantics NOT interface**

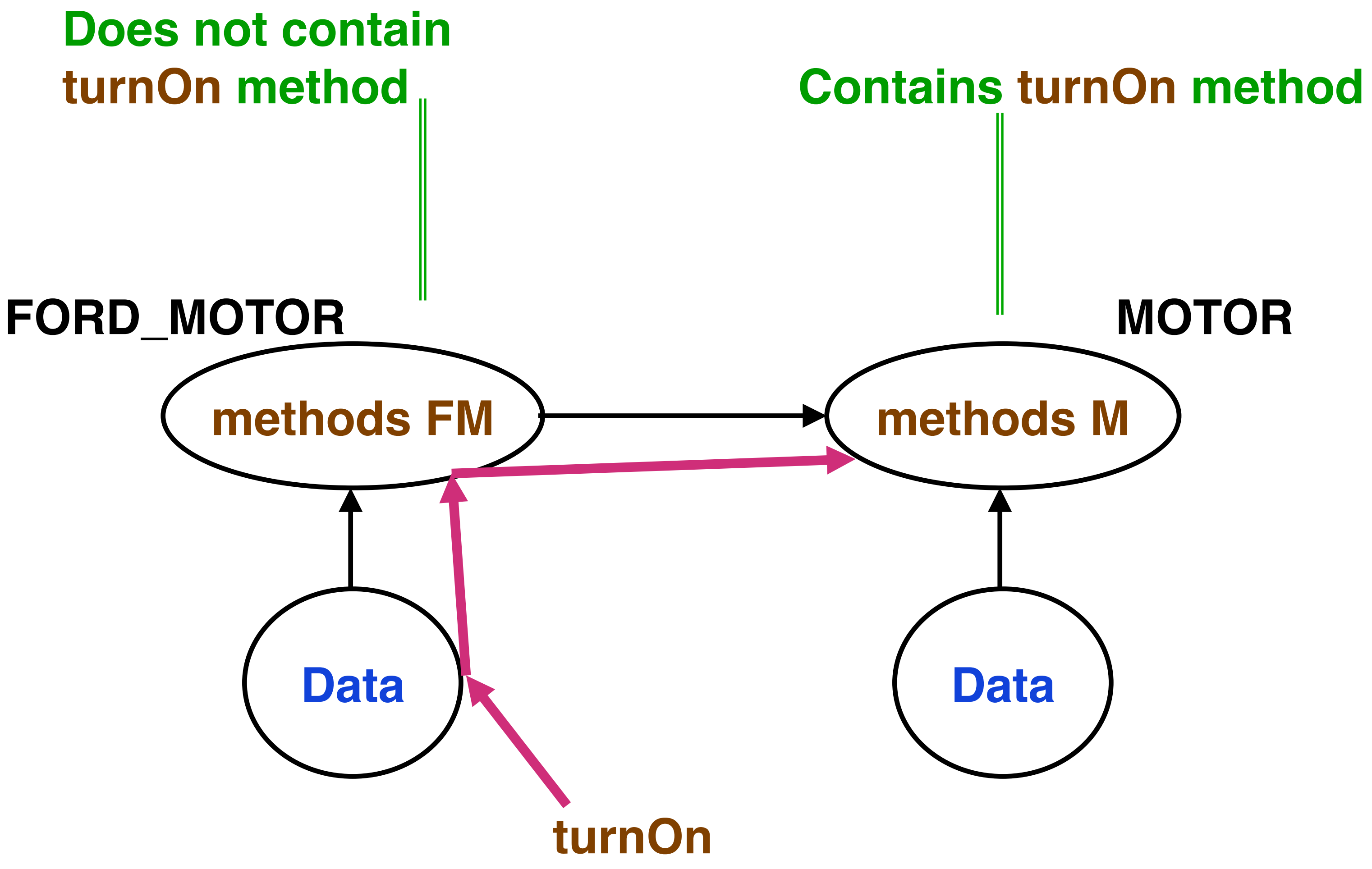


Subclass & Superclass

- Subclass
  - » Class A is a subclass of class B if A inherits from B
- Superclass
  - » Class A is a superclass of class B if B inherits from A

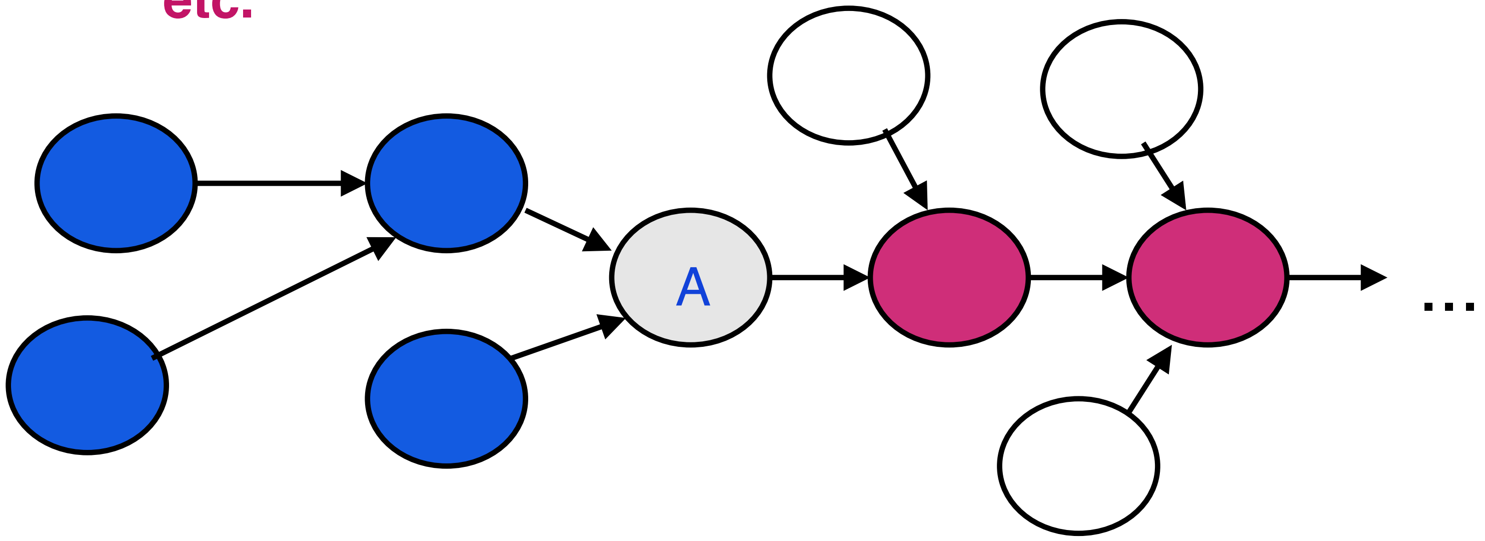


Message passing with Inheritance



Class Hierarchy

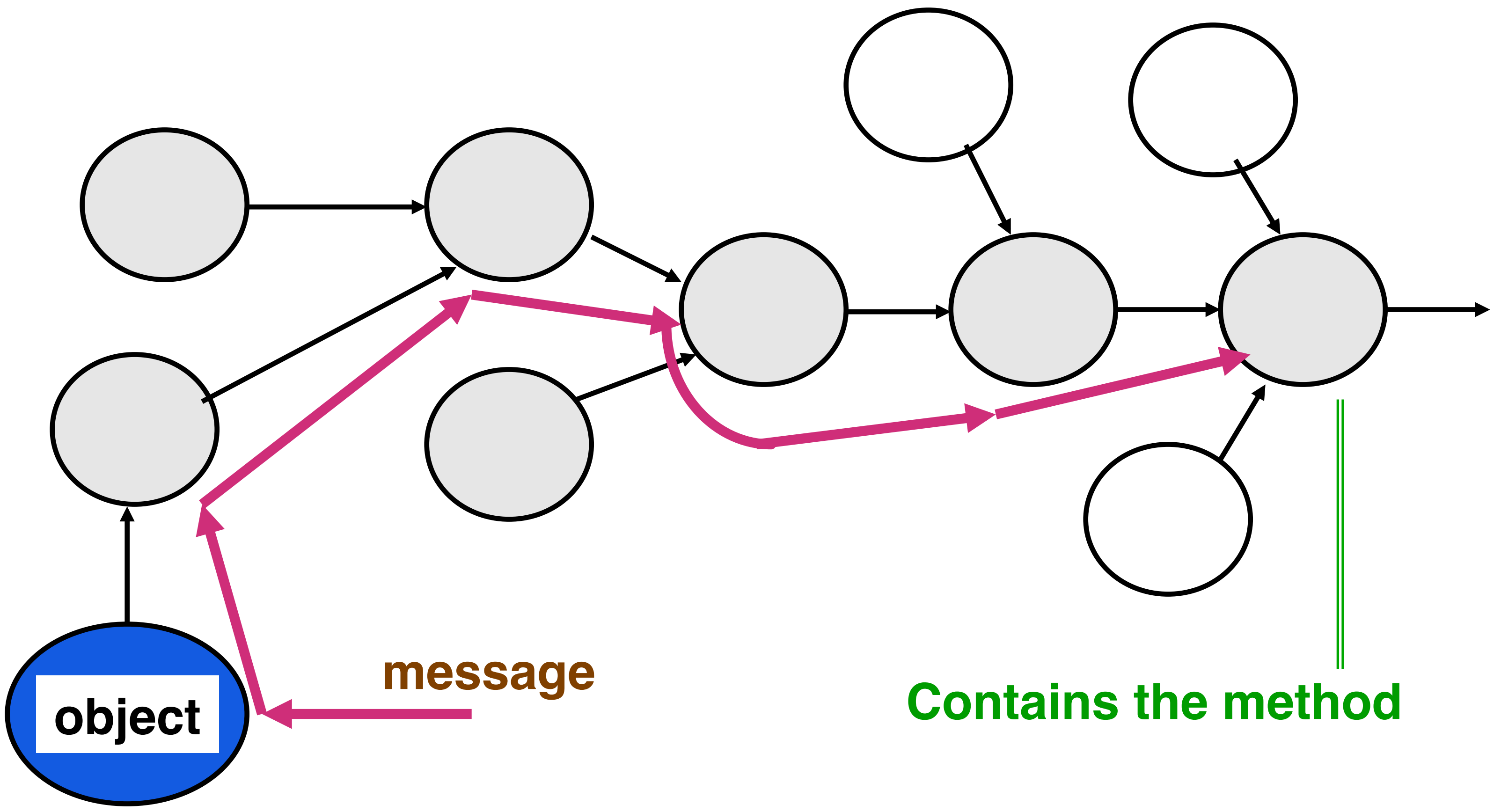
- Containing class A – includes A and the following
  - » **The transitive closure of superclasses of class A**
    - > **superclasses of A, superclasses of superclass of A, etc.**



  - » **The transitive closure of the subclasses of class A**
    - > **subclasses of A, subclasses of the subclasses of A, etc.**

Message Passing in Class Hierarchy

- Message passes up the superclass chain until method is found



The Real Story on Data

- Inheritance means a subclass has available all the methods of the transitive closure of its superclasses

The Real Story on Data – 2

- Inheritance means a subclass has available all the methods of the transitive closure of its superclasses

- This implies that an object is comprised of instances of all the data from the transitive closure of its superclasses

The Real Story on Data – 3

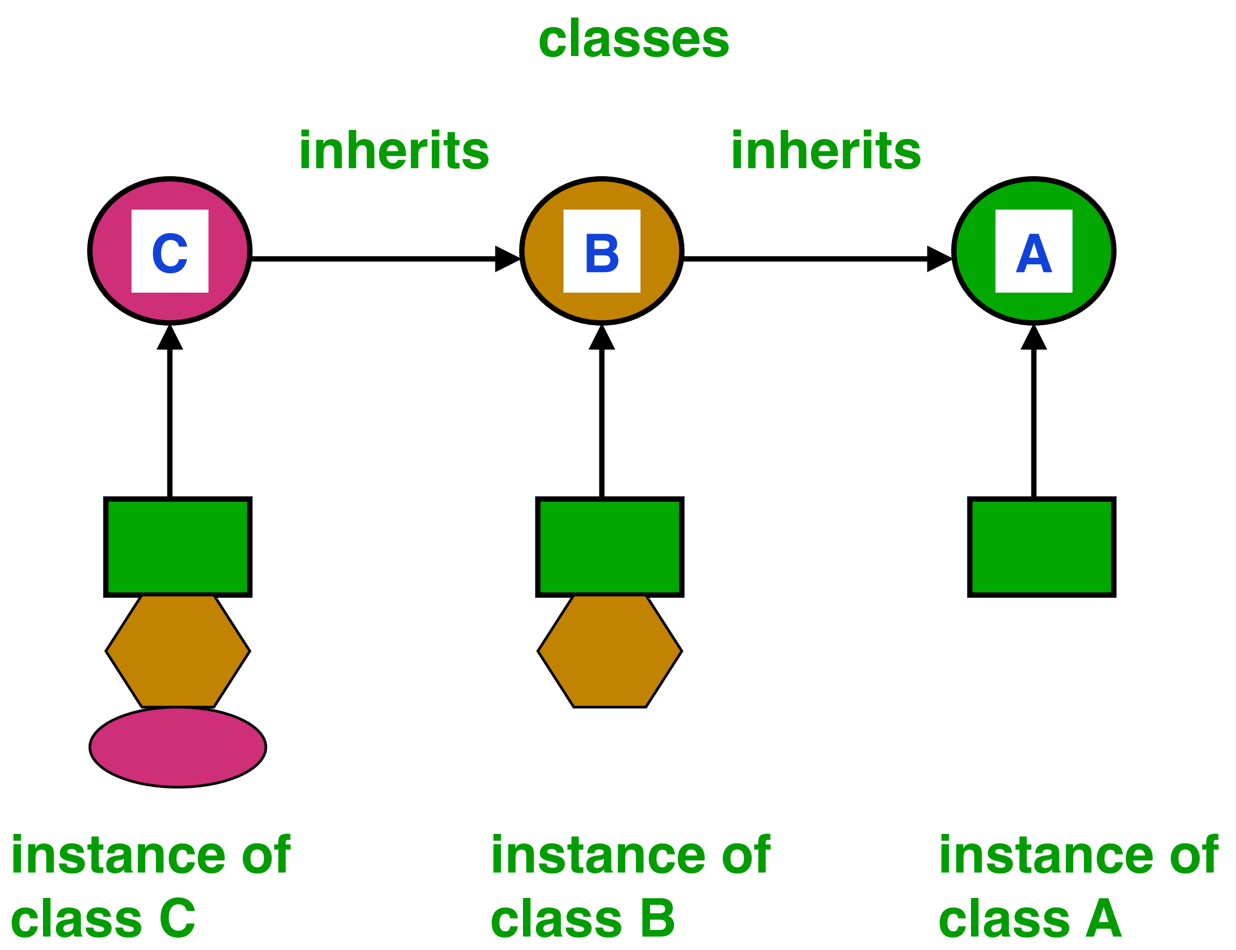
- Inheritance means a subclass has available all the methods of the transitive closure of its superclasses

- This implies that an object is comprised of instances of all the data from the transitive closure of its superclasses
  - » **Or else the methods in the superclasses would not have any data to work on**

Data Story – 2

» Instance of B has data from B and A

» Instance of C has data from C, B and A



"Is a" Relationship

- When class B inherits from class A
  - » **B inherits all the methods of A**

"Is a" Relationship – 2

- When class B inherits from class A
  - **B inherits all the methods of A**
    - **Instances of B can be sent all the messages that A responds to**

"Is a" Relationship – 3

- When class B inherits from class A
  - » **B inherits all the methods of A**
    - > **Instances of B can be sent all the messages that A responds to**
  - » **B inherits all the data from A**
    - > **Instances B have instances of all the data of A**

"Is a" Relationship – 3

- When class B inherits from class A
  - **B inherits all the methods of A**
    - **Instances of B can be sent all the messages that A responds to**
  - **B inherits all the data from A**
    - **Instances B have instances of all the data of A**
  - **As a consequence we can say**

B is an A

"Is a" Relationship – 4

- When class B inherits from class A
  - » **B inherits all the methods of A**
    - > **Instances of B can be sent all the messages that A responds to**
  - » **B inherits all the data from A**
    - > **Instances B have instances of all the data of A**
  - » **As a consequence we can say**

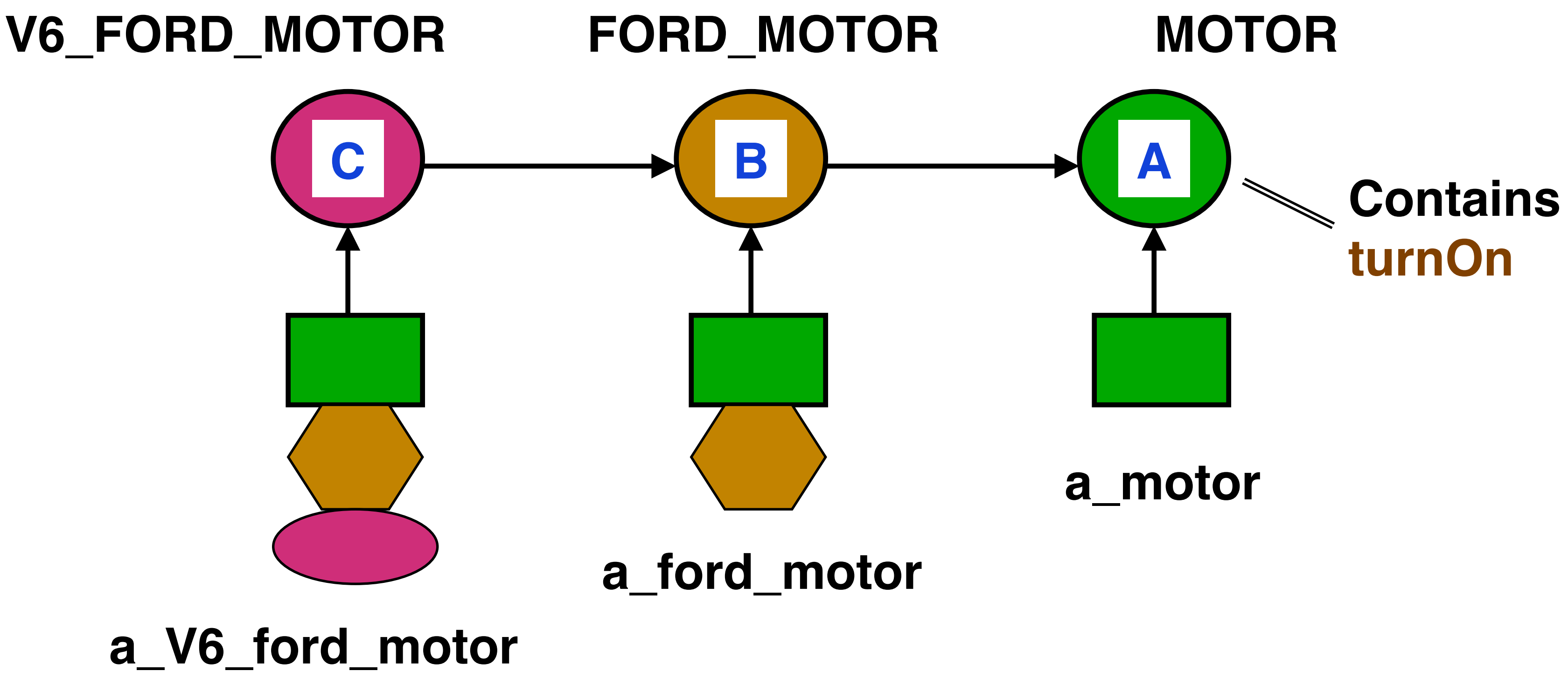
B is an A

- Every instance of B is also an instance of A
  - » **Can use B where ever an A can be used**

"Is a" Example

- Can say following because all instances are **MOTORS**

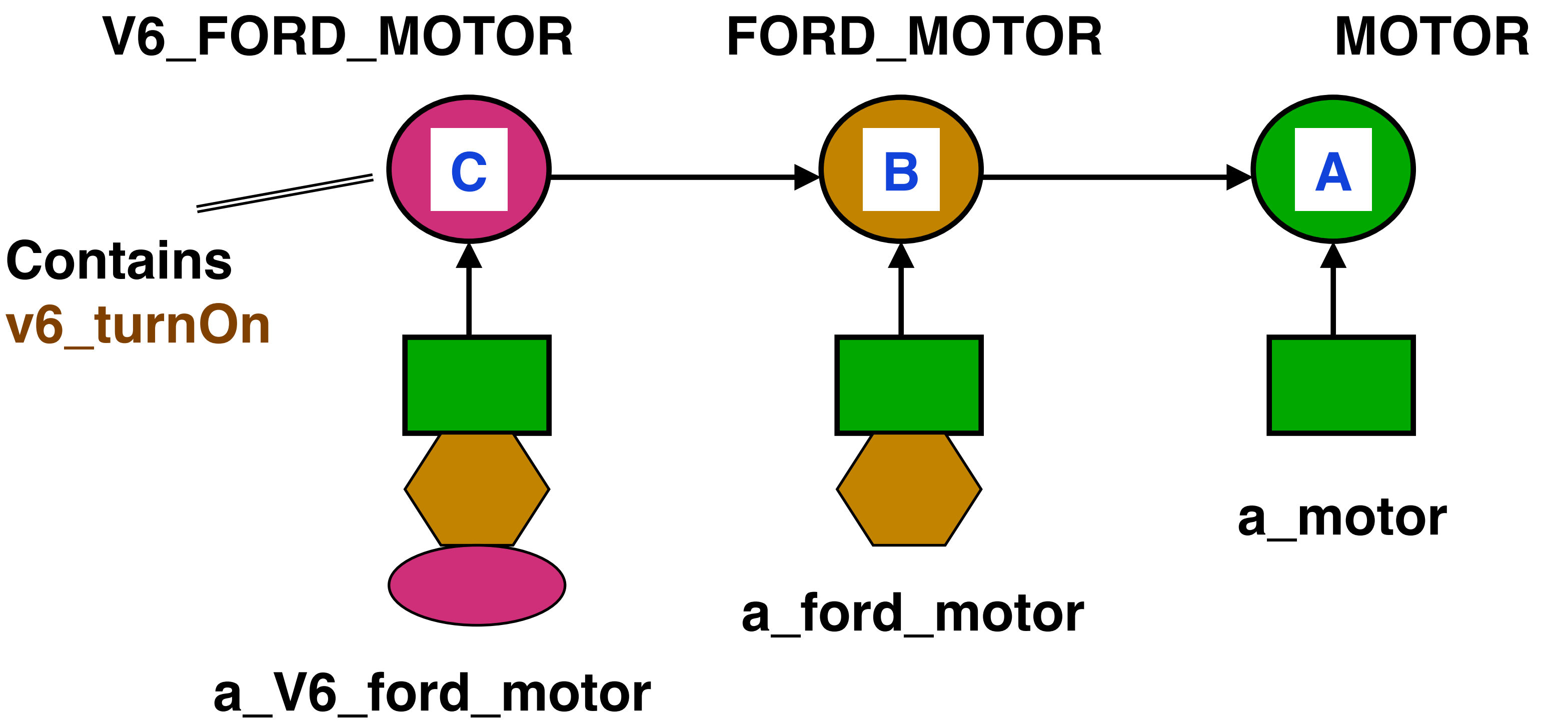
a_V6_ford_motor . turnOn
a_ford_motor . turnOn
a_motor . turnOn



"Is a" Example – 2

- Can not say following because **MOTOR** is not a **V6_FORD_MOTOR**

a_motor . v6_turnOn **Invalid, it does not compute**



What is a Meta Class?

- What sort of thing is a class?

What is a Meta Class? – 2

- What sort of thing is a class?
  - » **It is also an object !**

What is a Meta Class? – 3

- What sort of thing is a class?
  - » **It is also an object !**
  - » **Consequently it needs to be an instance of a class**

What is a Meta Class? – 4

- What sort of thing is a class?
  - **It is also an object !**
  - **Consequently it needs to be an instance of a class**
- A meta class is the class that has a class as an instance

What is a Meta Class? – 5

- What sort of thing is a class?
  - » **It is also an object !**
  - » **Consequently it needs to be an instance of a class**
- A meta class is the class that has a class as an instance
- There is only one meta class for each class

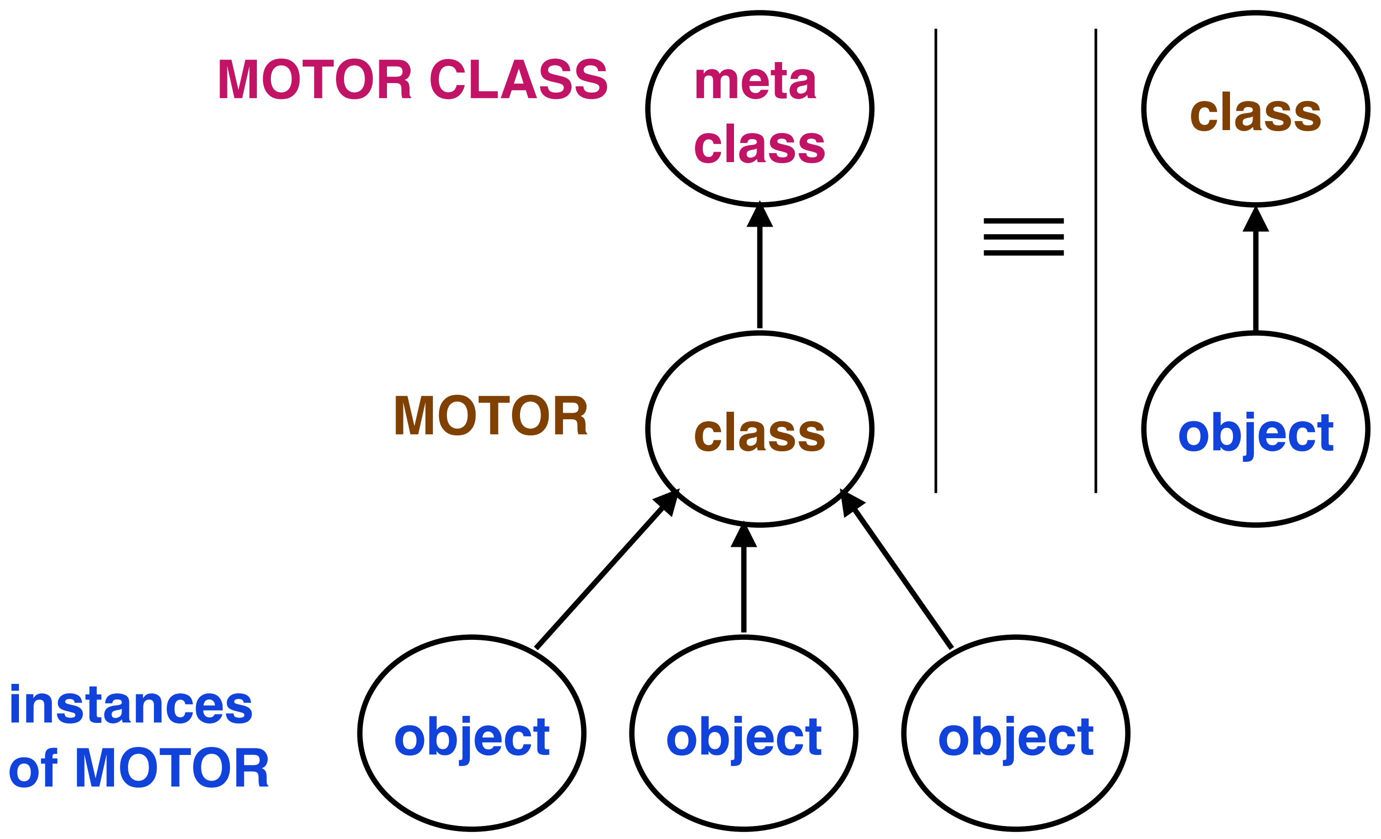
What is a Meta Class? – 6

- What sort of thing is a class?
  - » **It is also an object !**
  - » **Consequently it needs to be an instance of a class**
- A meta class is the class that has a class as an instance
- There is only one meta class for each class

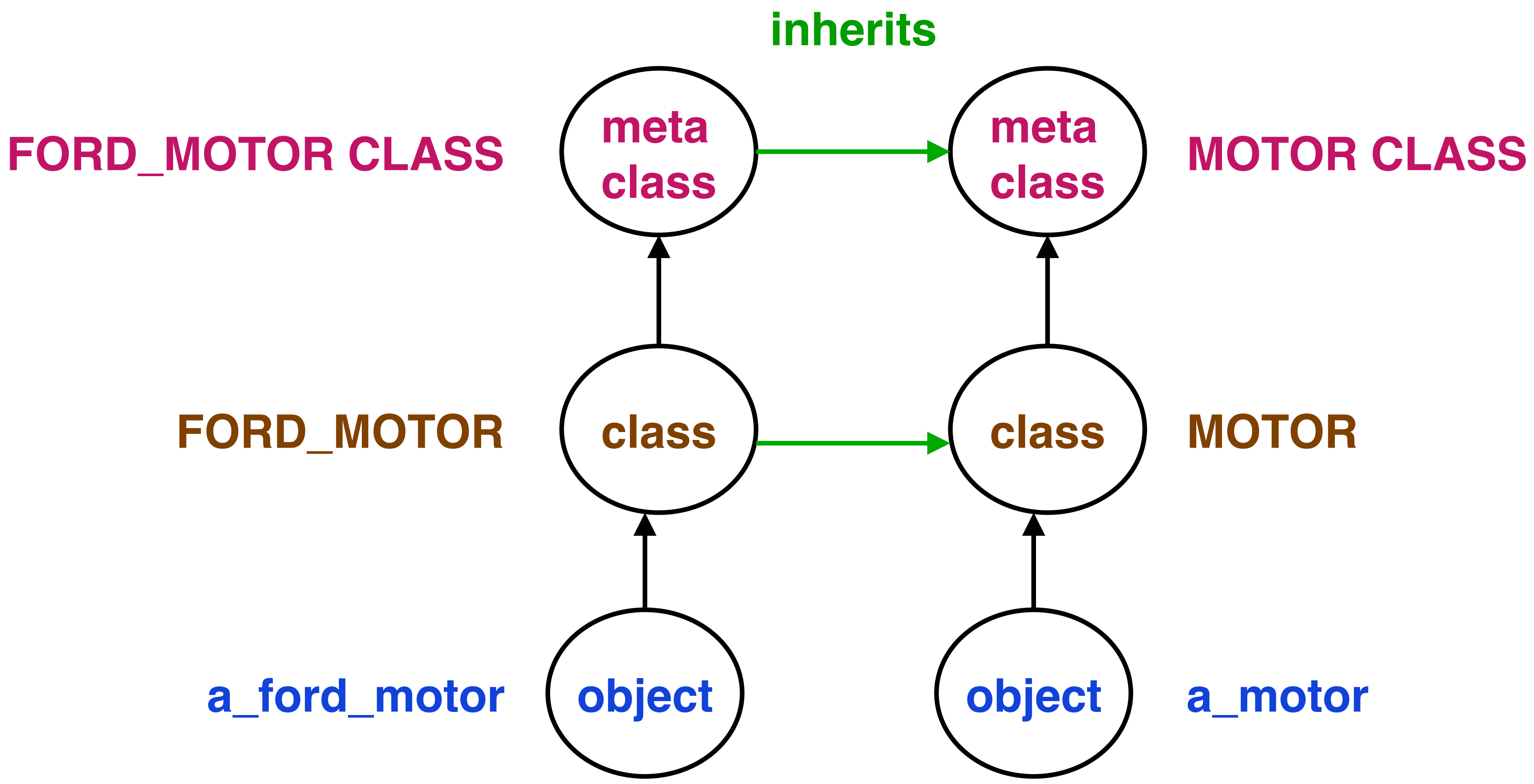
Anything you can do, I can do meta.

-- Daniel Dennett

The Small Picture – Smalltalk OO



Meta Class Inheritance – Smalltalk OO



Meta Class Creation – Smalltalk OO

- When **FORD_MOTOR** is created as a subclass of **MOTOR** then
  - » **Smalltalk automatically creates the meta class FORD_MOTOR CLASS and makes it a subclass of MOTOR CLASS**

Meta Class Creation – Smalltalk OO

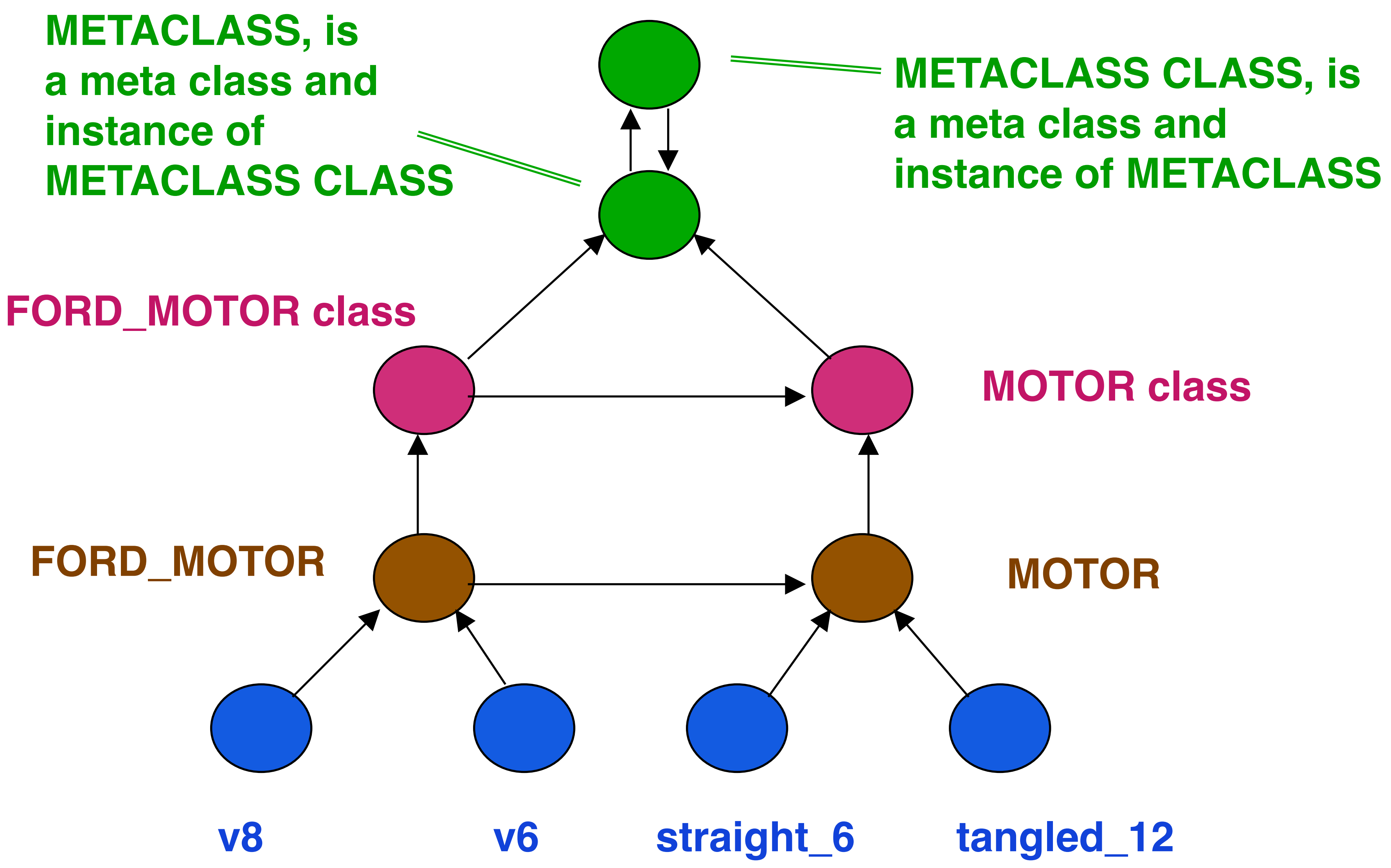
- When **FORD_MOTOR** is created as a subclass of **MOTOR** then
  - » **Smalltalk automatically creates the meta class FORD_MOTOR CLASS and makes it a subclass of MOTOR CLASS**
- Meta class are not directly accessible to the user

Meta Class Creation – Smalltalk OO

- When **FORD_MOTOR** is created as a subclass of **MOTOR** then
  - » **Smalltalk automatically creates the meta class FORD_MOTOR CLASS and makes it a subclass of MOTOR CLASS**
- Meta class are not directly accessible to the user

BUT meta classes are objects !!!

The Big Picture – Smalltalk OO


METACLASS, is a meta class and instance of METACLASS CLASS
METACLASS CLASS, is a meta class and instance of METACLASS
FORD_MOTOR class
MOTOR class
FORD_MOTOR
MOTOR
v8
v6
straight_6
tangled_12

Meta Classes Benefits

- Benefit
  - » **Uniform treatment of all objects**
    - > **Classes are first class citizens**

Meta Classes Benefits & Drawbacks

- Benefit
  - » **Uniform treatment of all objects**
    - > **Classes are first class citizens**
- Drawback
  - » **No strong typing**
    - > **More difficult to create error free software**

Other Mechanisms

- Provide a set of features available to all classes
  - » **Eiffel – Put them in a universal ANY class**
  - » **Java – Put them in a special class CLASS**

Other Mechanisms – 2

- Operations that characterize a class rather than object
  - **Most obvious is object creation**
    - **Eiffel – use special construct create**
    - **Java – use special construct new**
  - **Others can be put into universal class**
    - **Eiffel – ANY**
    - **Java ???**

Other Mechanisms – 3

- Obtain information about a class
  - » Eiffel
    - > stored in one instance of E_CLASS per class
  - » Java
    - > class Class<T>
      - – Instances represent classes and interfaces
      - – Use object.getClass() to access the Class
        - object.getClass().getName() to get the name of the class to which object belongs