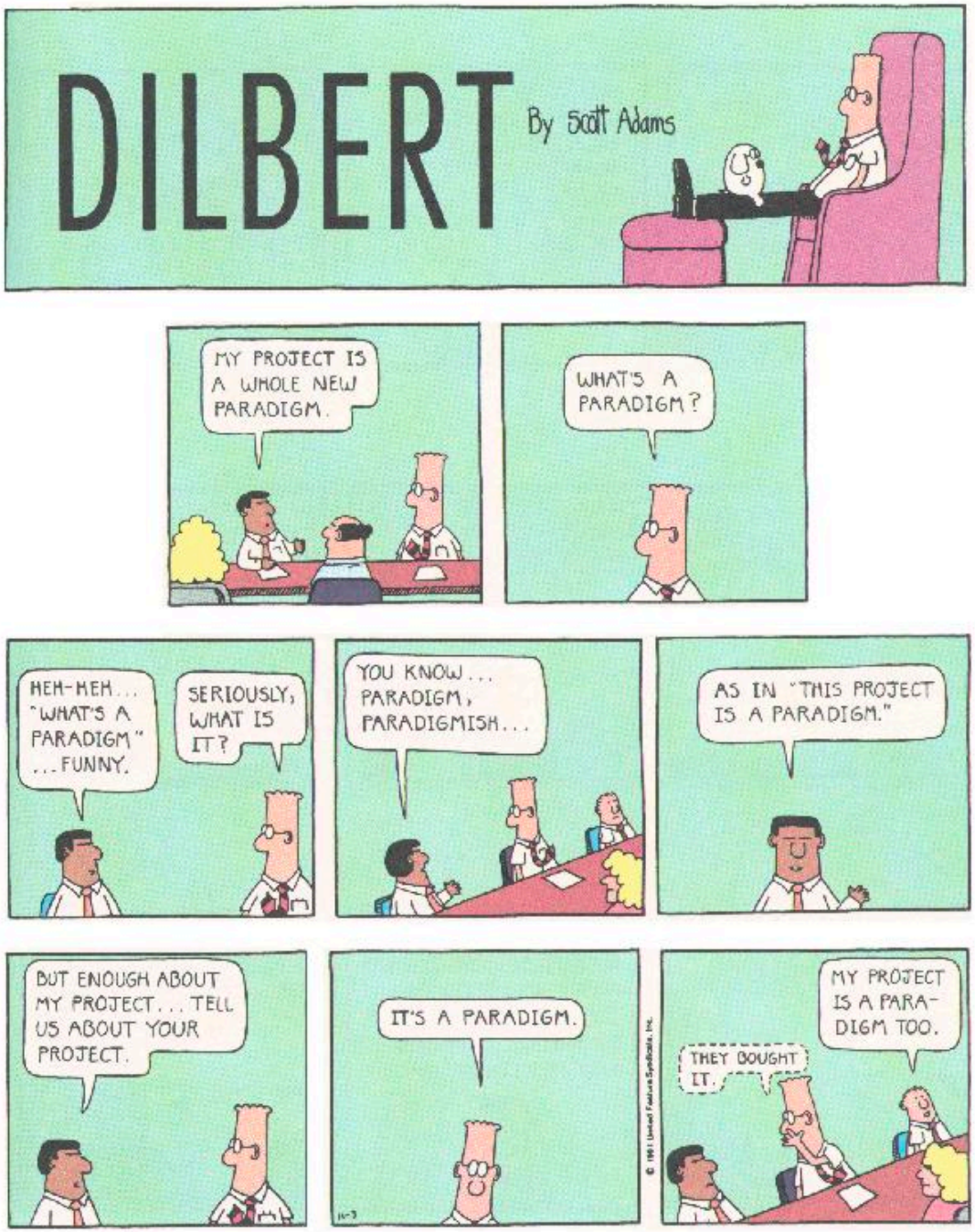
SEG 2105 - LECTURE 02

Object Orientation

WHAT IS

OBJECT ORIENTATION


DILBERT
By Scott Adams
MY PROJECT IS A WHOLE NEW PARADIGM.
WHAT'S A PARADIGM?
HEH-HEH... "WHAT'S A PARADIGM" ...FUNNY.
SERIOUSLY, WHAT IS IT?
YOU KNOW... PARADIGM, PARADIGMISH...
AS IN "THIS PROJECT IS A PARADIGM."
BUT ENOUGH ABOUT MY PROJECT... TELL US ABOUT YOUR PROJECT.
IT'S A PARADIGM.
MY PROJECT IS A PARA-DIGM TOO.
THEY BOUGHT IT.

paradigm noun

par·a·digm | \ ˈper-ə-ˌdīm, ˈpa-rə- *also* -ˌdim\

Definition of *paradigm*

1 : EXAMPLE, PATTERN

especially : an outstandingly clear or typical example or archetype

// ... regard science as the *paradigm* of true knowledge.
— G. C. J. Midgley

2 : an example of a conjugation or declension showing a word in all its inflectional forms

3 : a philosophical and theoretical framework of a scientific school or discipline within which theories, laws, and generalizations and the experiments performed in support of them are formulated

// the Freudian *paradigm* of psychoanalysis

broadly : a philosophical or theoretical framework of any kind

https://upload.wikimedia.org/wikipedia/commons/f/f7/Programming_paradigms.svg

https://en.wikipedia.org/wiki/Programming_paradigm

https://www.merriam-webster.com/dictionary/paradigm

OBJECT ORIENTED (STRUCTURED)

STRUCTURED (IMPERATIVE)

EVENT DRIVEN (PROCEDURAL)

FUNCTIONAL (DECLARATIVE)

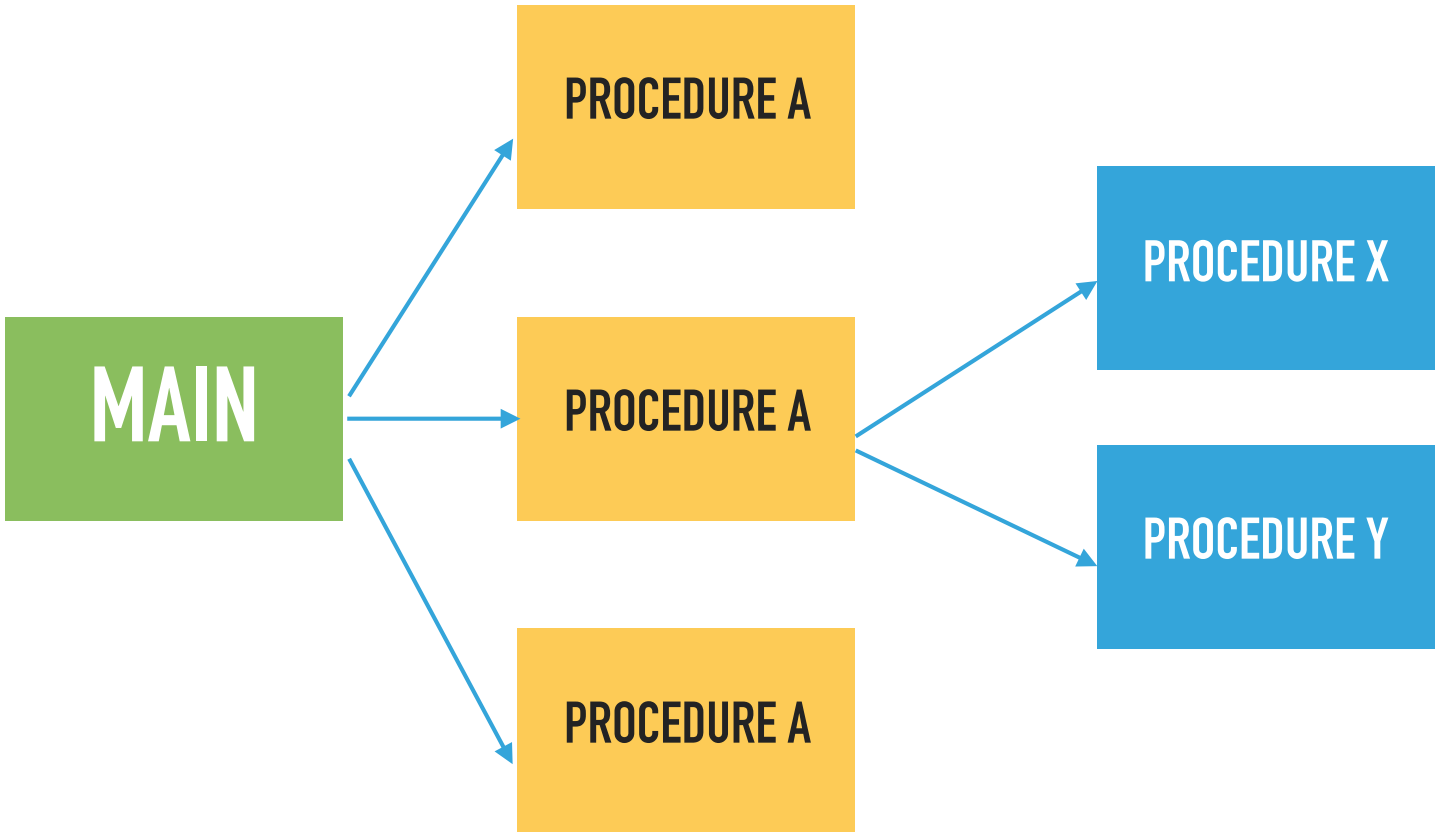
DECLARATIVE (LOGIC)

PROCEDURAL (IMPERATIVE)

FEW LANGUAGES ARE PURE

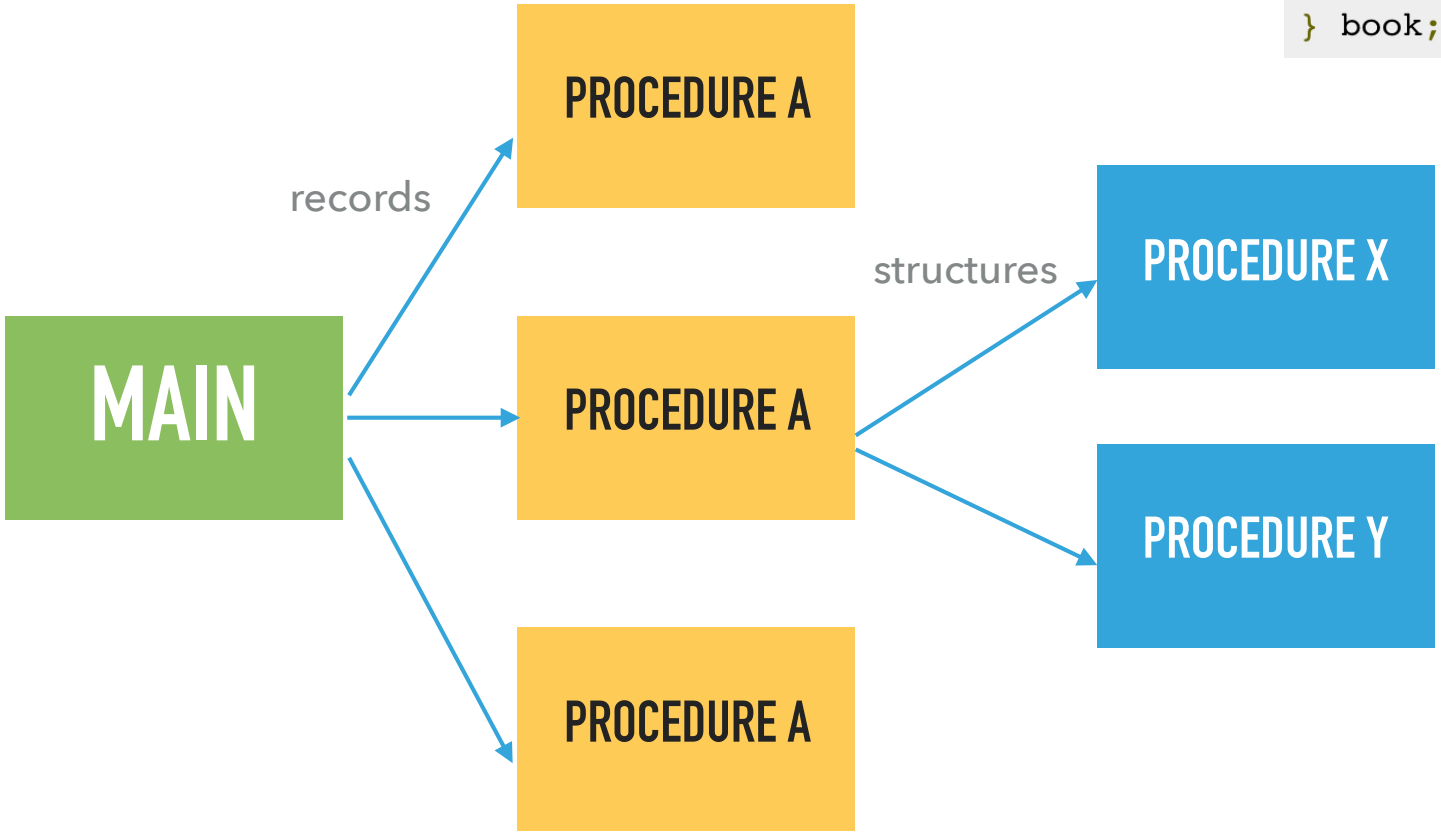
NOW BACK TO OO

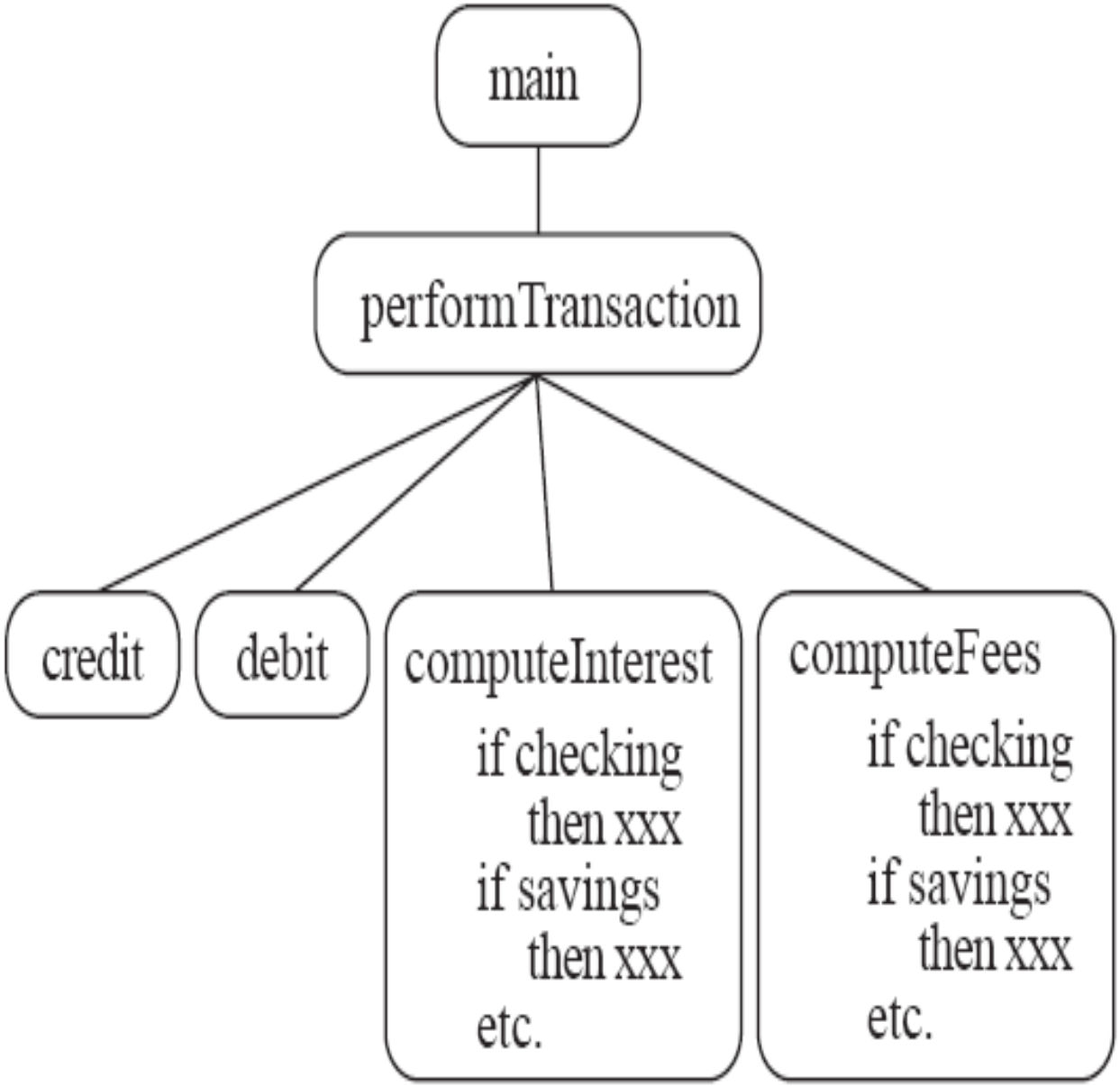
Procedural paradigm



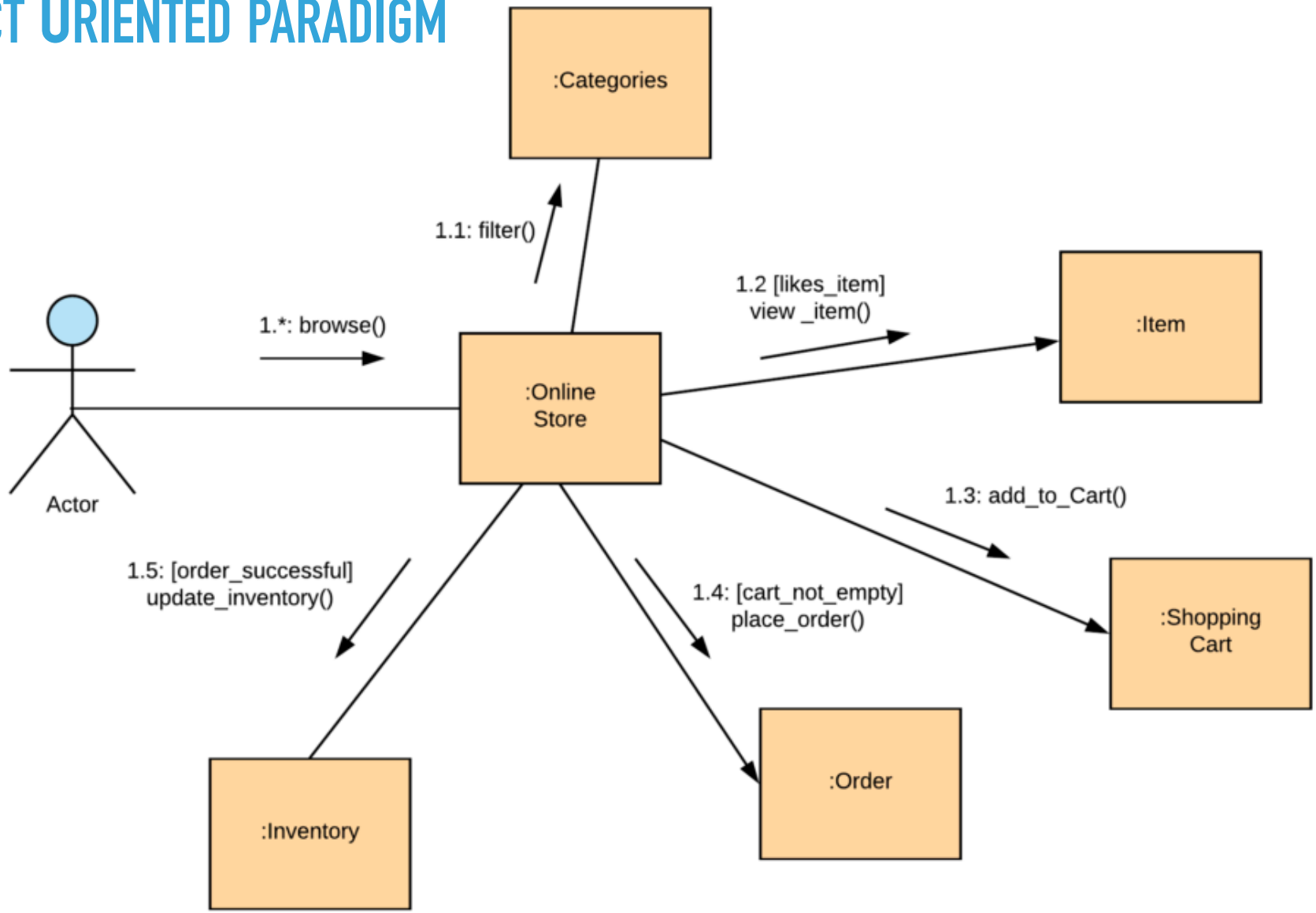
PROCEDURAL PARADIGM

```
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

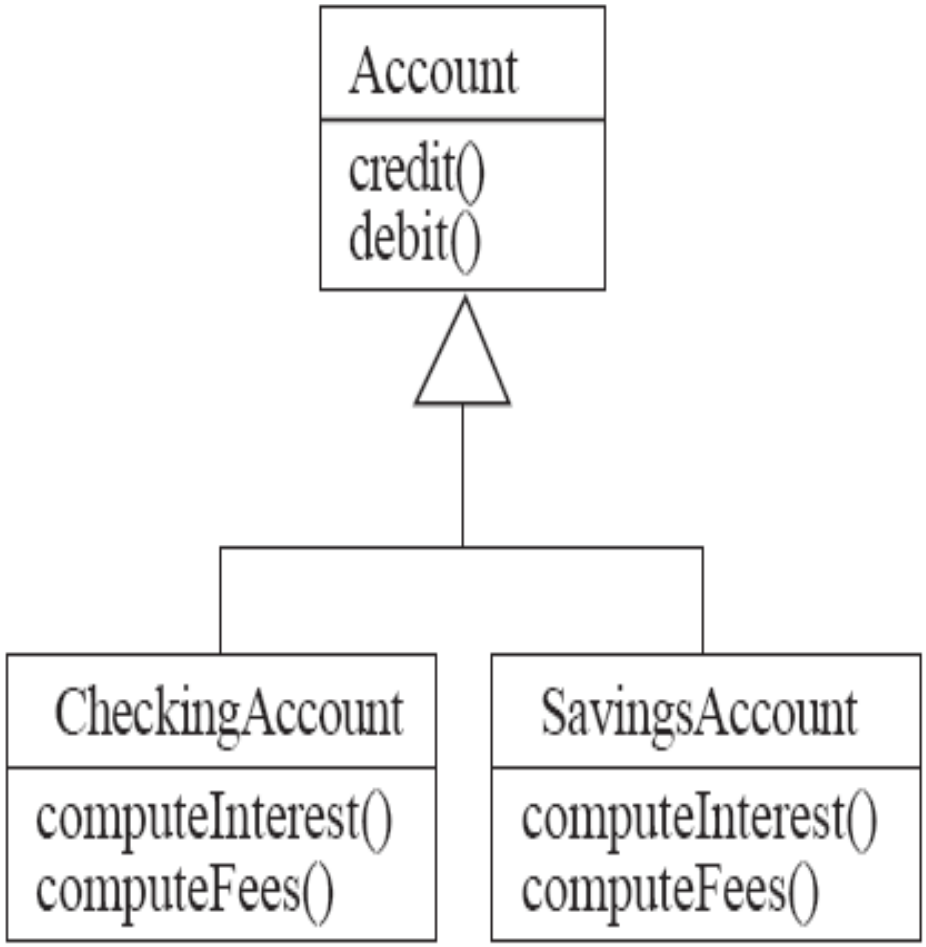



main
performTransaction
credit
debit
computeInterest
if checking
then xxx
if savings
then xxx
etc.
computeFees
if checking
then xxx
if savings
then xxx
etc.

OBJECT ORIENTED PARADIGM

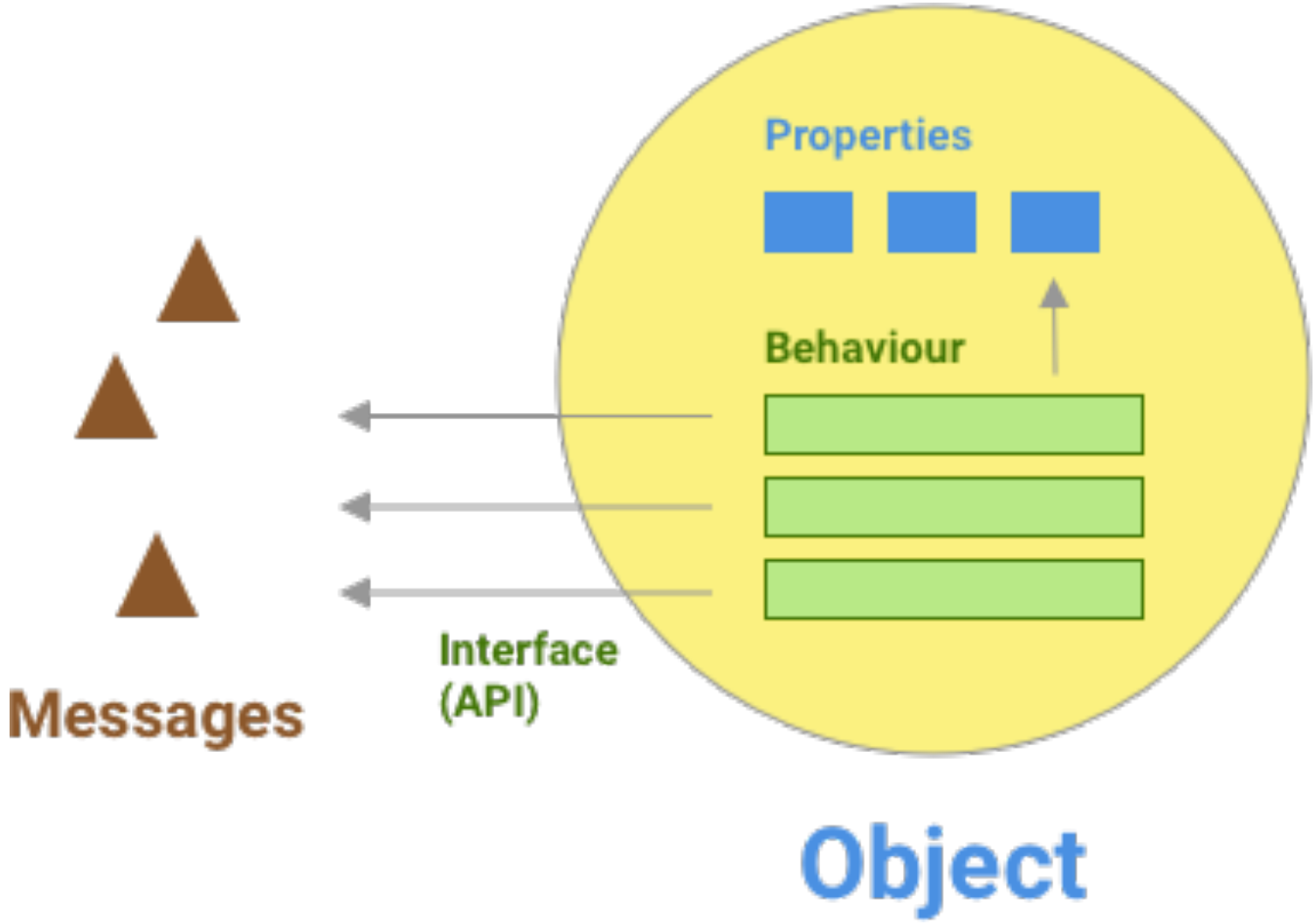


https://tallyfy.com/uml-diagram/

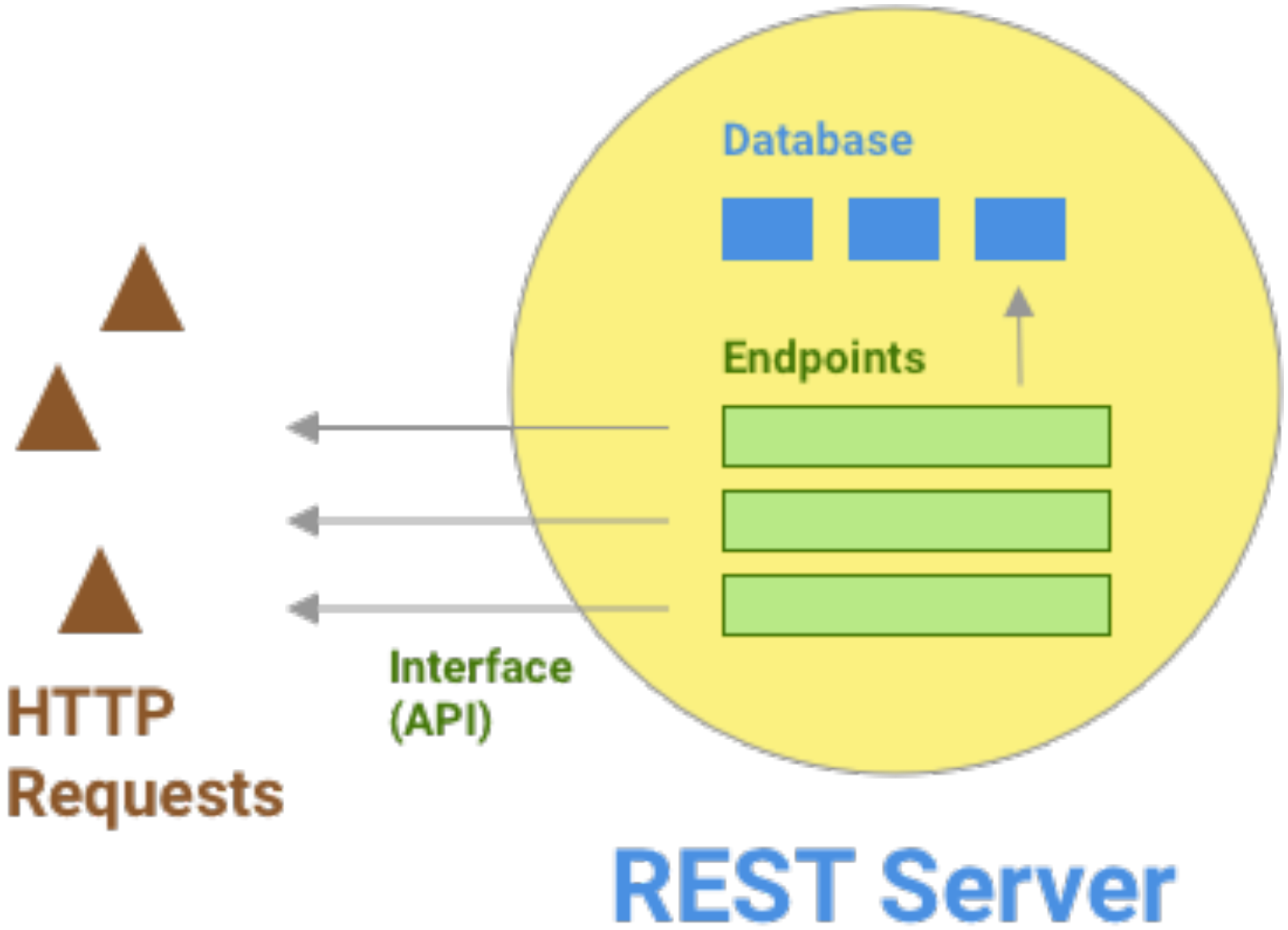


Umple Example

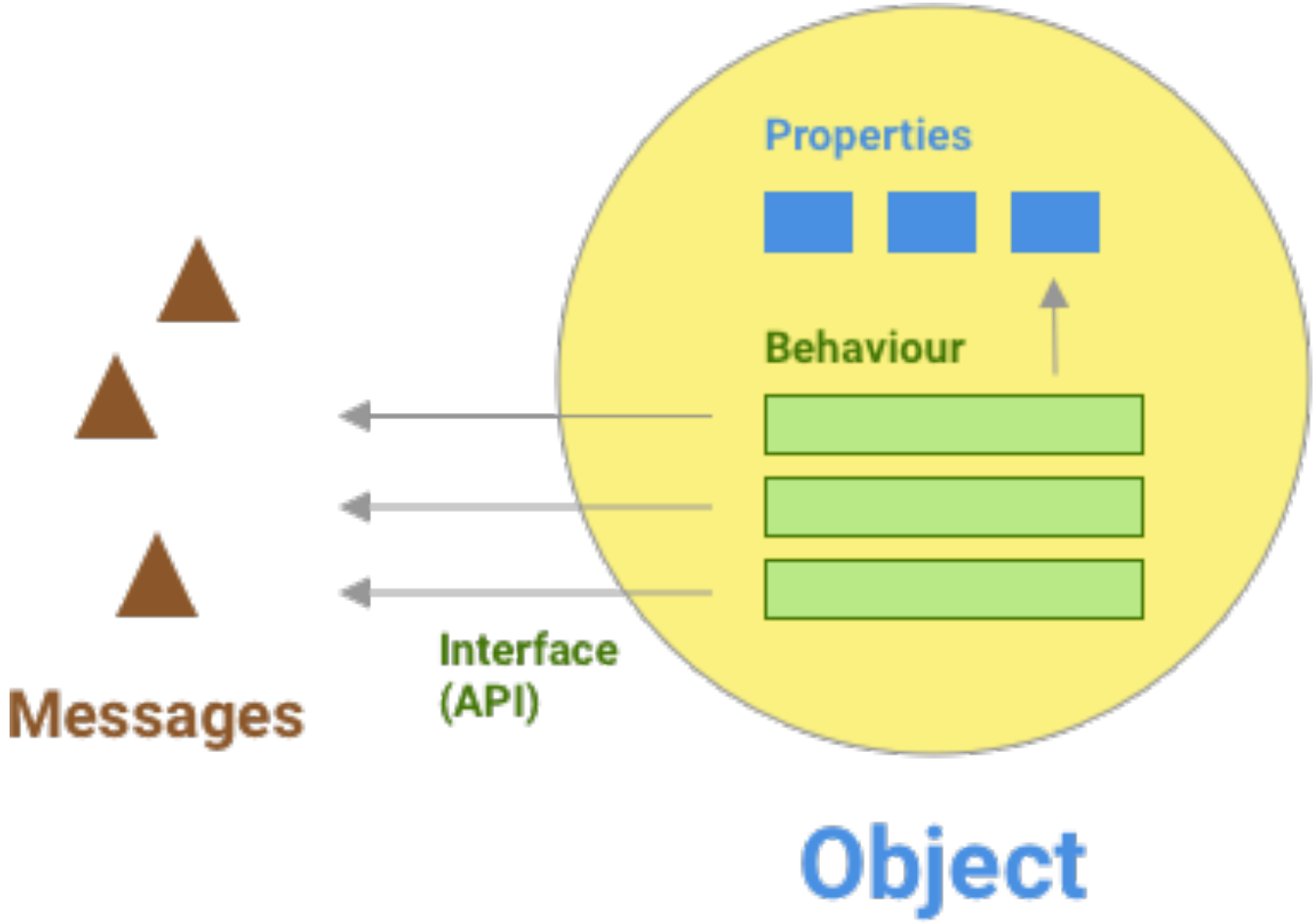
Object oriented paradigm



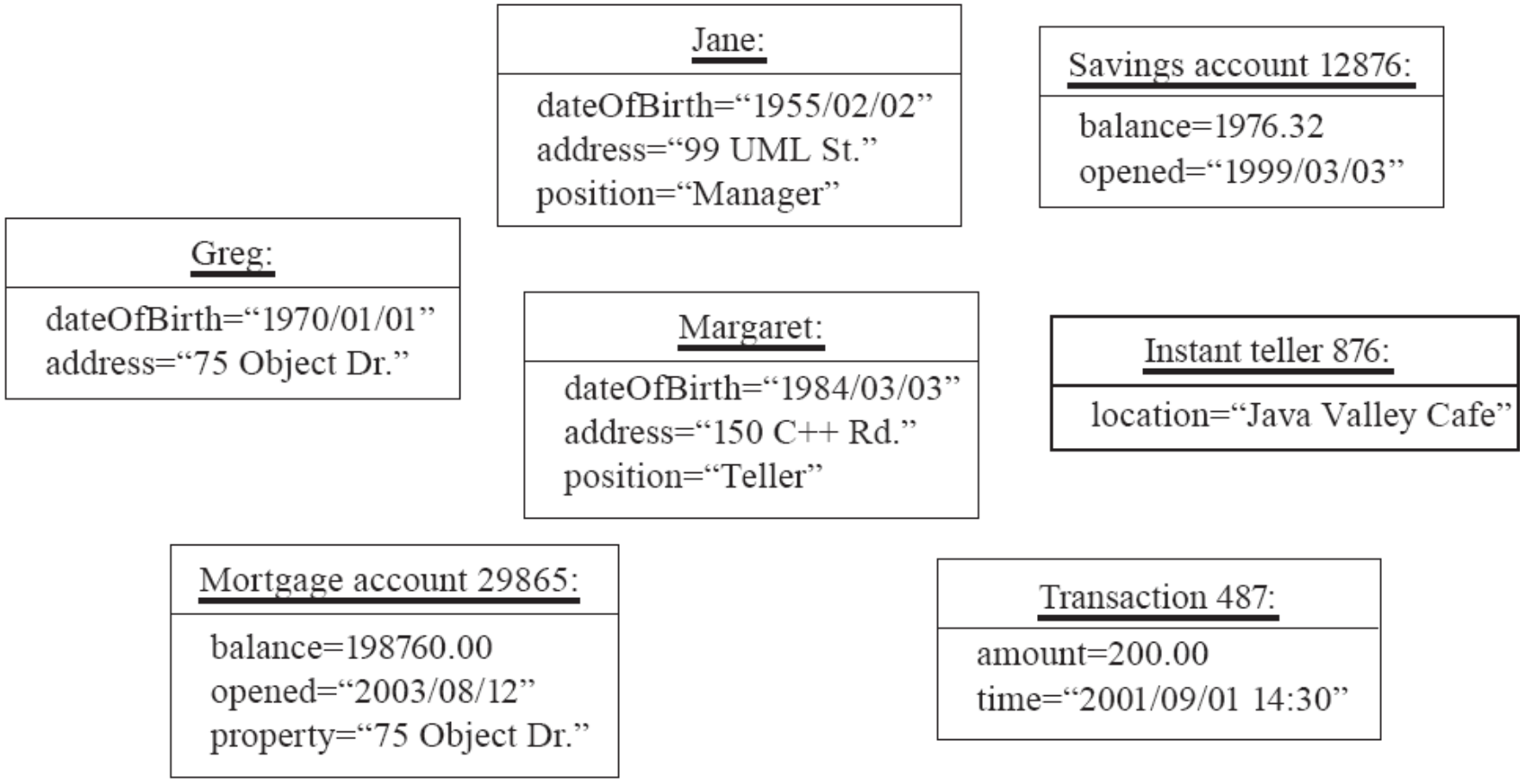
Quite similar to RESTful APIs



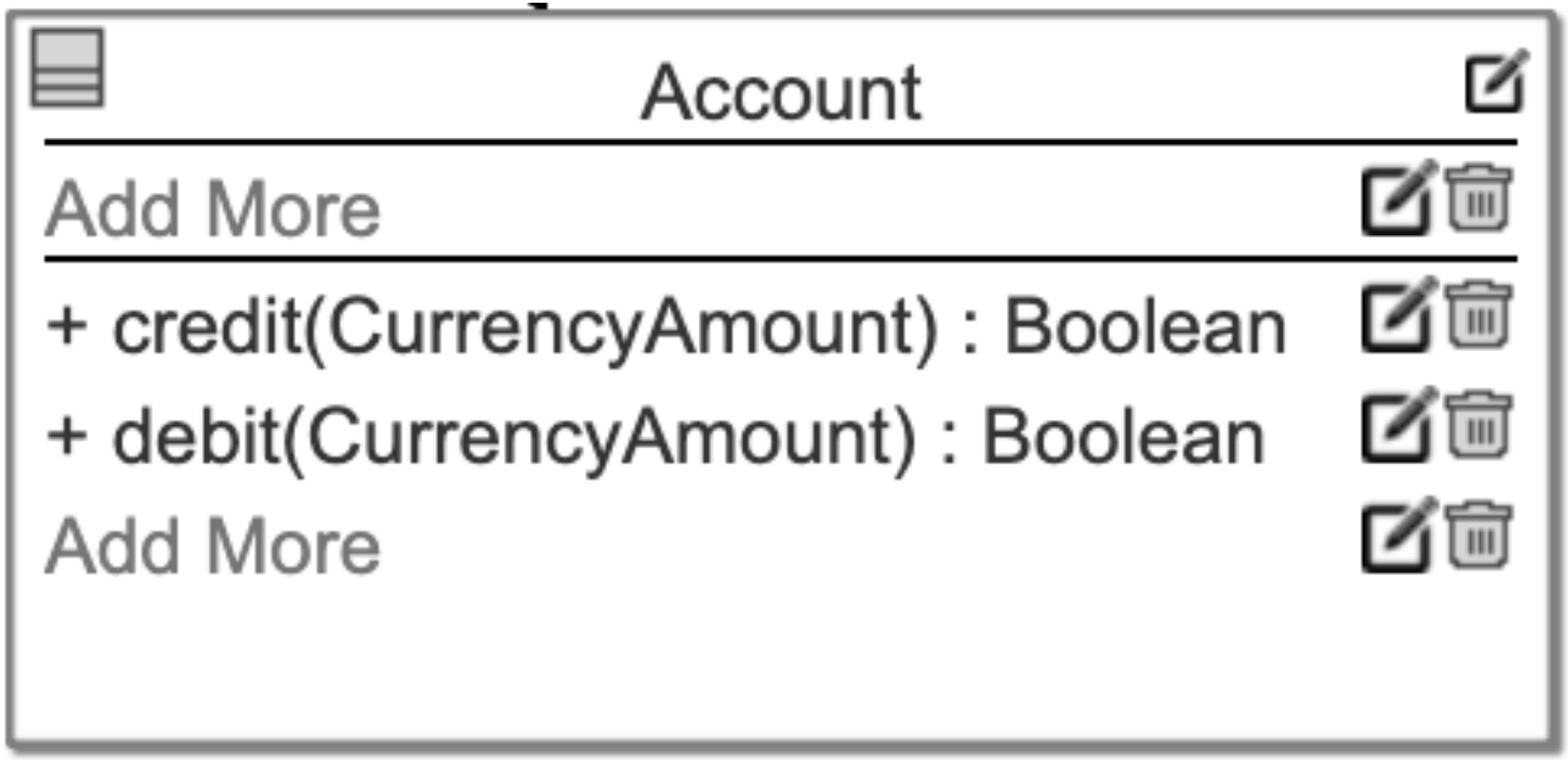
Objects



OBJECT DIAGRAM



Classes

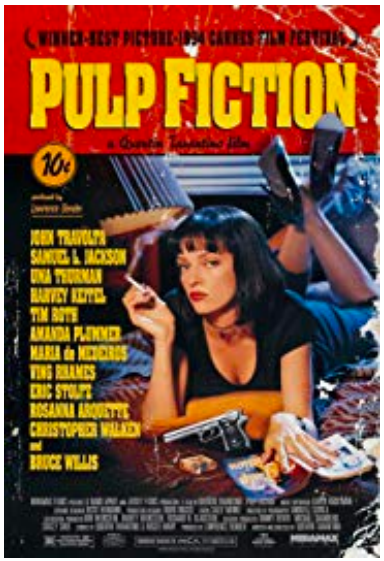

Account
Add More
+ credit(CurrencyAmount) : Boolean
+ debit(CurrencyAmount) : Boolean
Add More

AM I A...

CLASS OR OBJECT?

FILM

FILM



CLASS; INSTANCES ARE SPECIFIC FILMS LIKE PULP FICTION

https://www.imdb.com/title/tt0110912/

REEL OF FILM

REEL OF FILM

CLASS: INSTANCES ARE PHYSICAL REELS

https://cinemacommentary.com/

REEL (OF FILM) SW19876

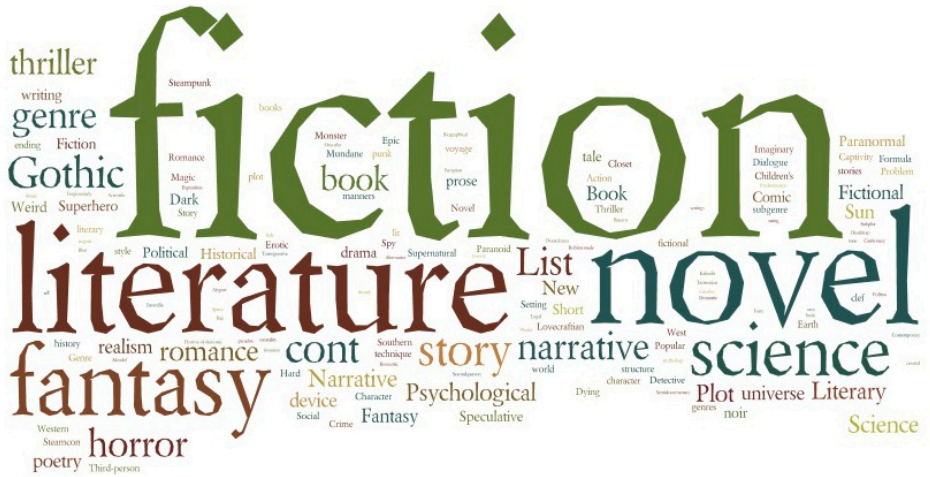
REEL (OF FILM) SW1987

INSTANCE OF REELOFFFILM

https://cinemacommentary.com/

SCIENCE FICTION

SCIENCE FICTION



INSTANCE OF THE CLASS GENRE.

https://eslchestnut.com/2013/07/09/book-genres-summer-reading/

SCIENCE FICTION FILM

SCIENCE FICTION FILM

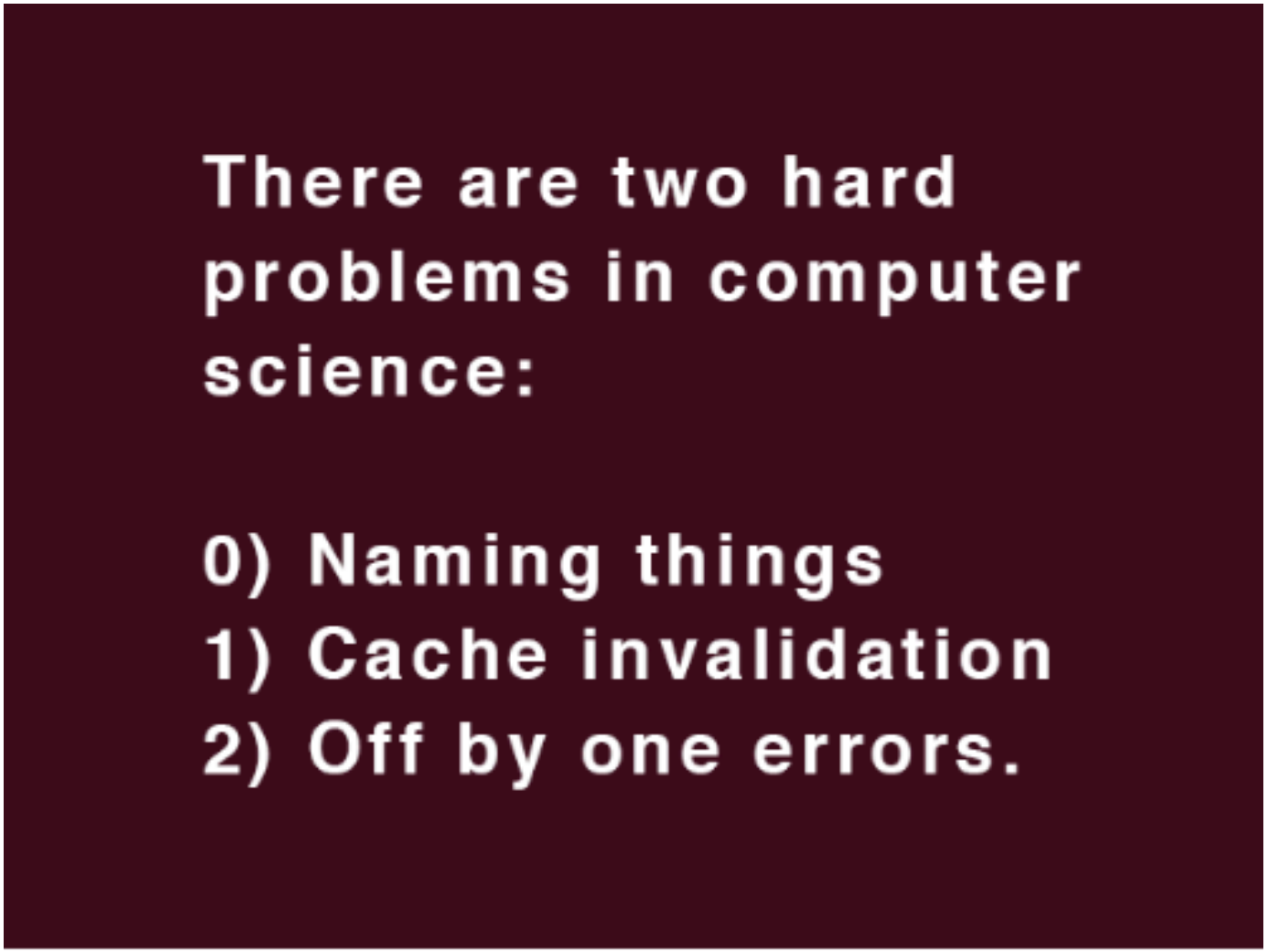
CLASS: INSTANCES INCLUDE ‘VERONICA MARS’

https://www.imdb.com/title/tt0412253/

SHOWING OF ‘VERONICA MARS SEASON 4’ ON HULU UNTIL JAN 2020.

SHOWING OF ‘VERONICA MARS SEASON 4’ ON HULU UNTIL JAN 2020.

INSTANCE OF ShowingOfTvSeries



Hard Problems

https://me.me/i/there-are-two-hard-problems-in-computer-science-0-naming-17965806

Naming classes

- Use capital letters
  - E.g. BankAccount not bankAccount
- Use singular nouns
- Use the right level of generality
  - E.g. Municipality, not City
- Make sure the name has only one meaning
  - E.g. ‘bus’ has several meanings

Instance Variables

Example instance variables

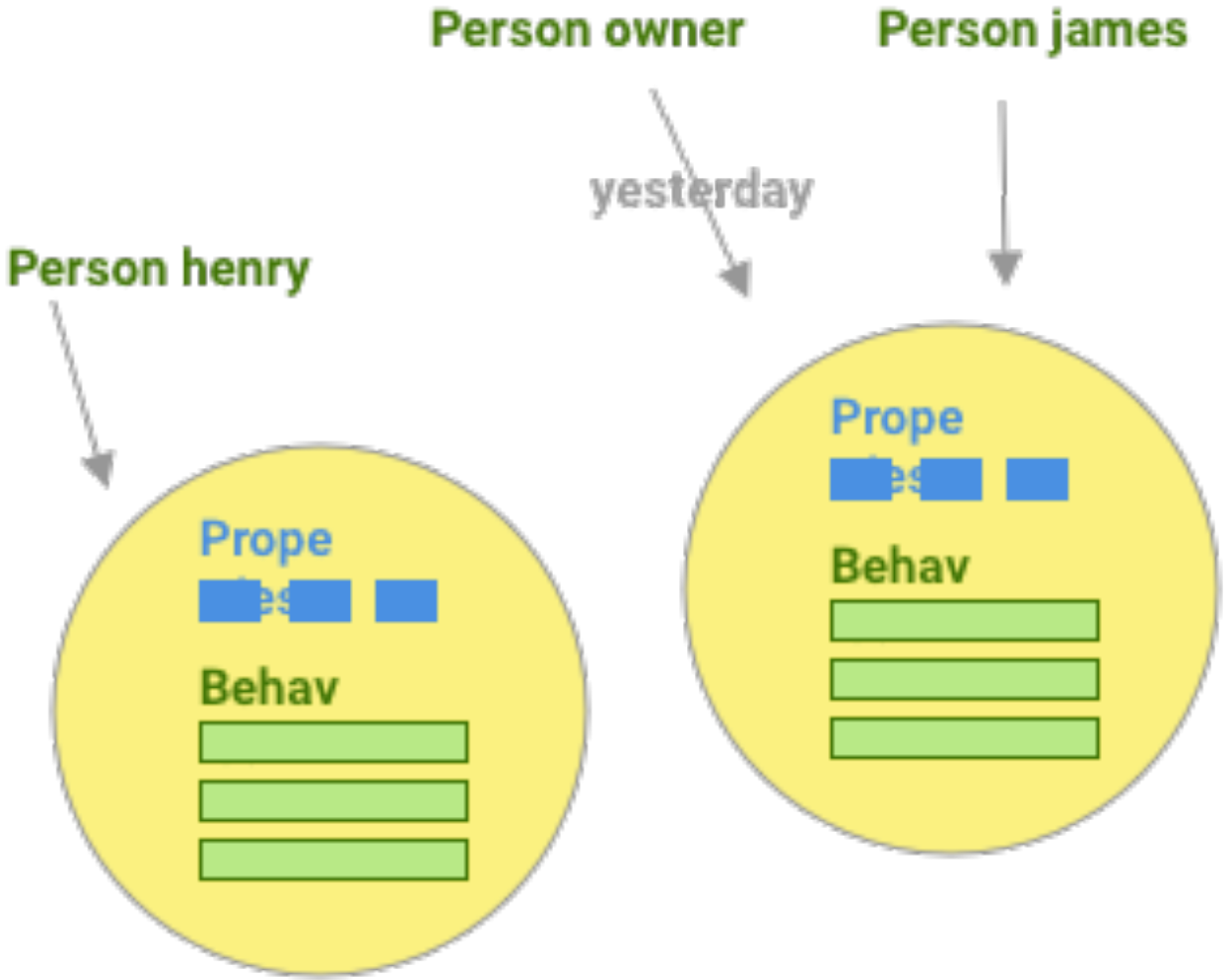
name

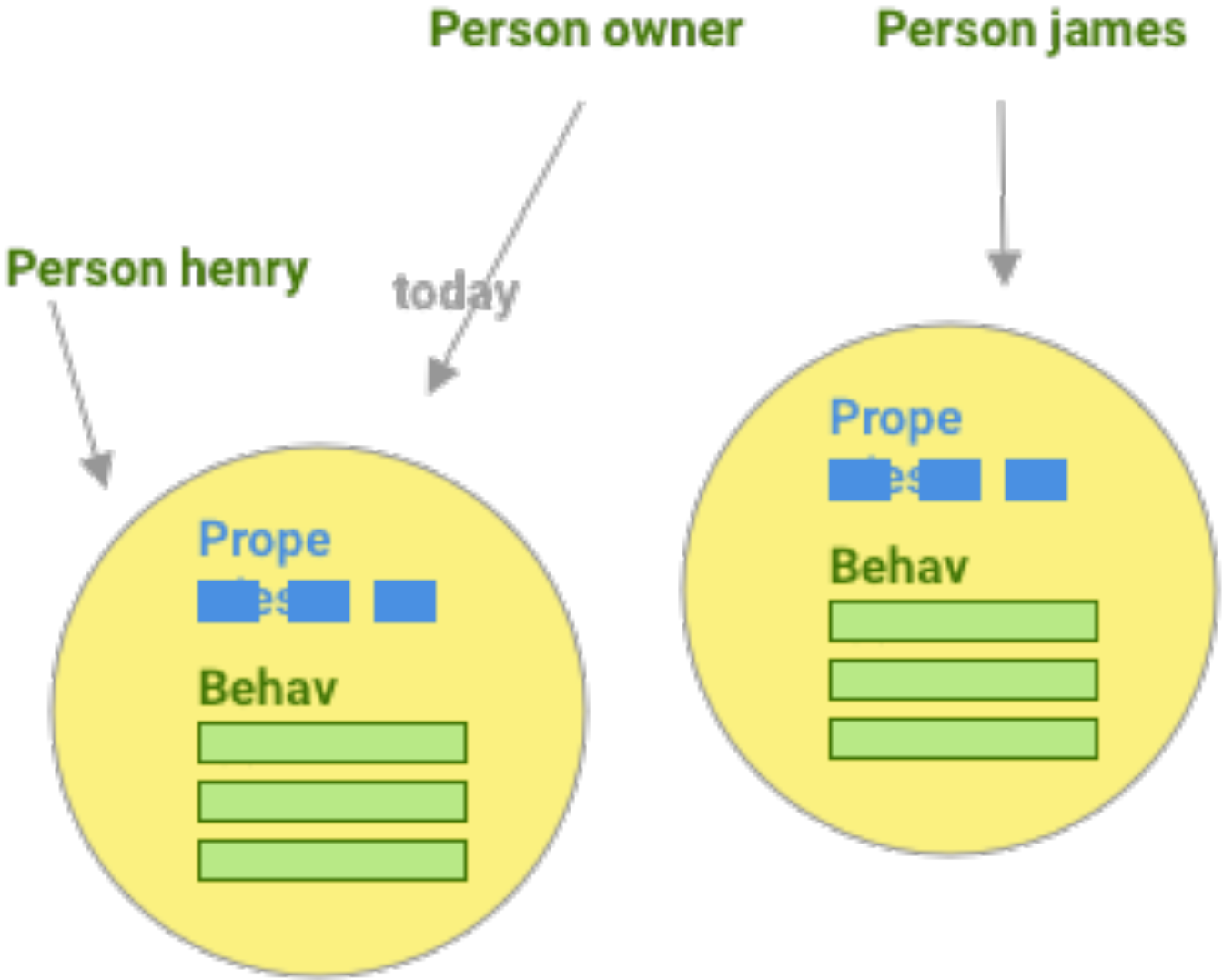
dateOfBirth

courses_taken

supervisor

VARIABLES != OBJECTS




Person owner
Person james
Person henry
today
Prope
Behav
Prope
Behav

Class variables

Static variables

Shared by all instances

Global (good for constants)

Global (bad otherwise)

```java
public class Circle {
  private static double PI = 3.14159;

  private double radius;

  public Circle(double aRadius) {
    radius = aRadius;
  }

  public double radius() {
    return radius;
  }

  public double diameter() {
    return radius * 2;
  }

  public double circumference() {
    return 2 * PI * diameter();
  }
}
```

```
class Circle {
  const double PI = 3.14159;

  double radius;
  double diameter = { radius *2; }
  double circumference = { 2 * PI * diameter }
}
```

Operation

- A **higher-level procedural abstraction** that specifies a type of behaviour
- **Independent** of any code which **implements** that behaviour

```
double area()
```

METHOD

- A **procedural abstraction** used to implement the behaviour of a class

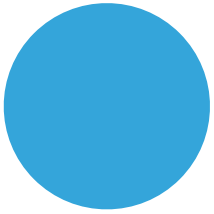
```
public double area() {
    return PI * radius * radius;
}
```

POLYMORPHISM

- Several **different classes** can have methods with the **same name**
- They implement the **same abstract operation** in ways suitable to each class

```
public double area() {
  return side * side;
}
```

```
public double area() {
  return PI * radius * radius;
}
```

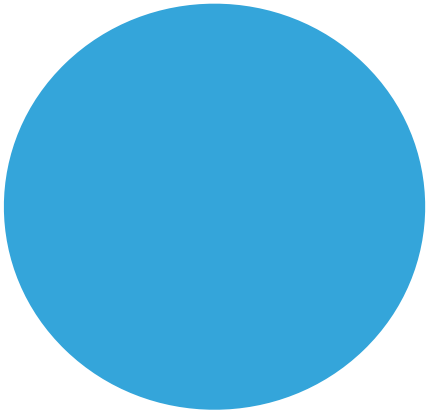


Polymorphism

- A property of object oriented software by which an abstract operation may be performed in different ways in different classes.
  - Requires that there be multiple **methods** of the **same name**
  - The choice of which one to execute **depends on the object** that is in a variable
  - Reduces **if-else** or **switch** statements

```
public double area() {
  return side * side;
}

public double perimeter() {
  return 4 * side;
}
```

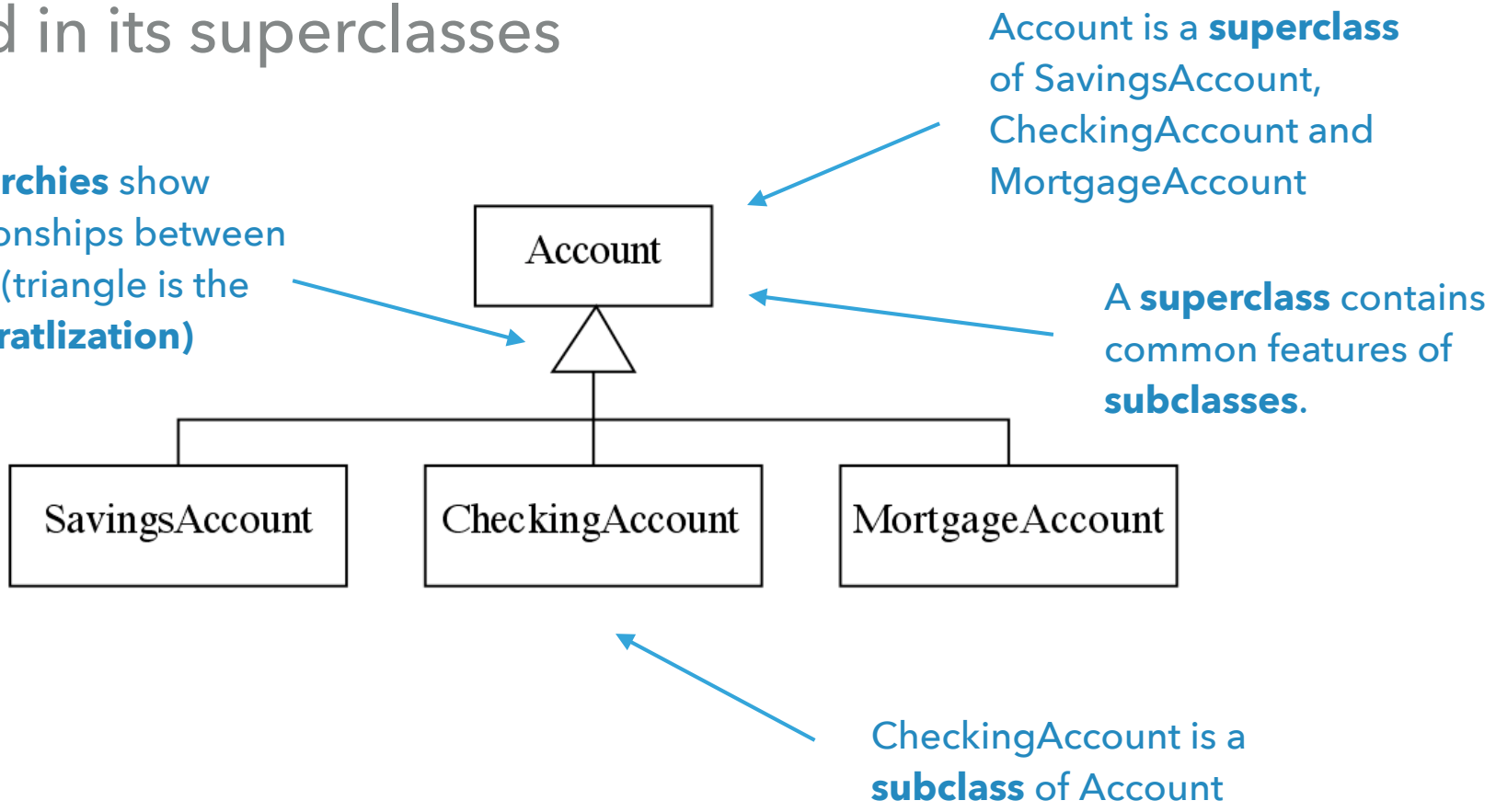


```
public double area() {
  return PI * radius * radius;
}

public double perimeter() {
  return circumference();
}
```

INHERITANCE

- The implicit possession by all subclasses of features defined in its superclasses



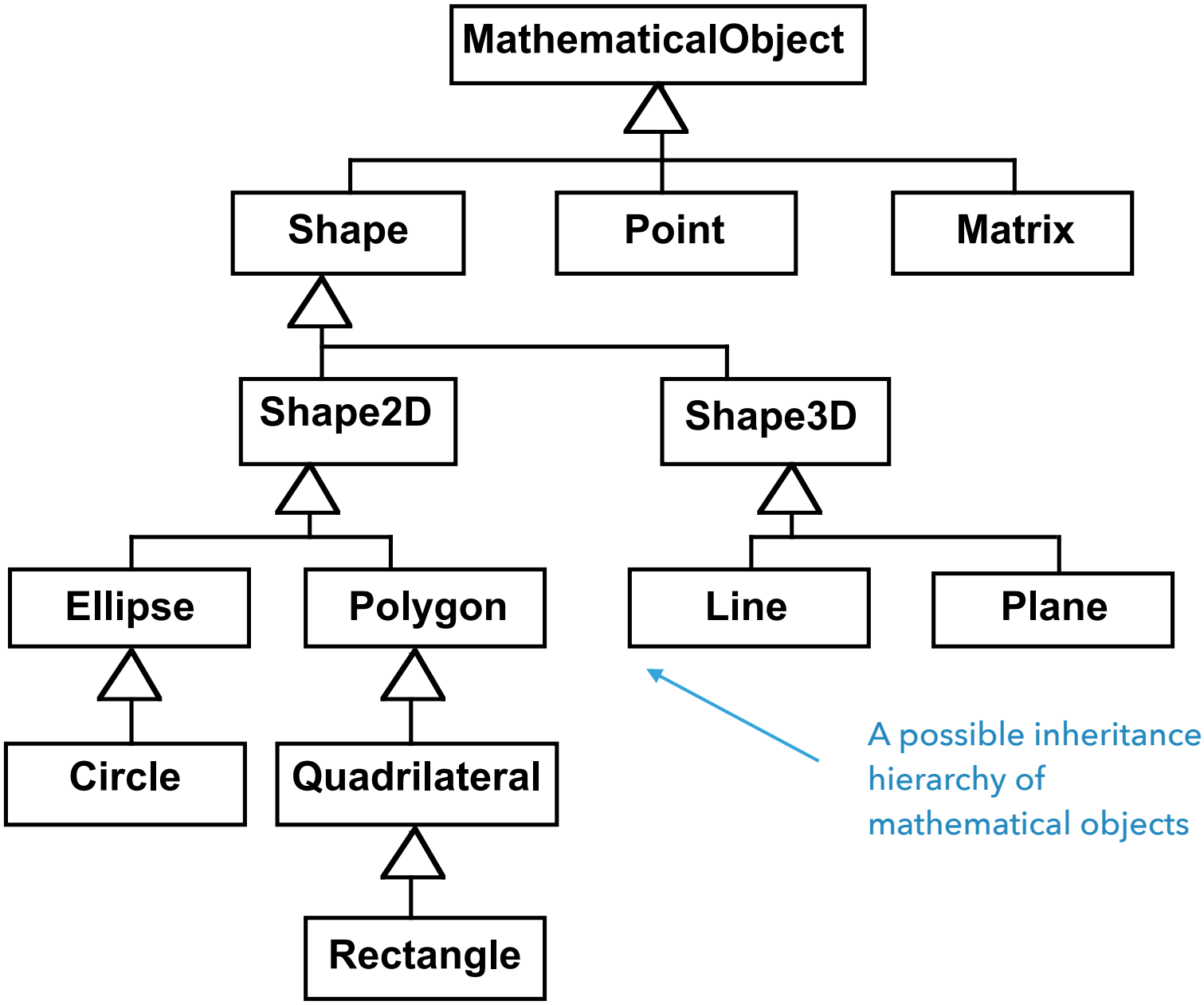
Open in Umple

The Isa Rule

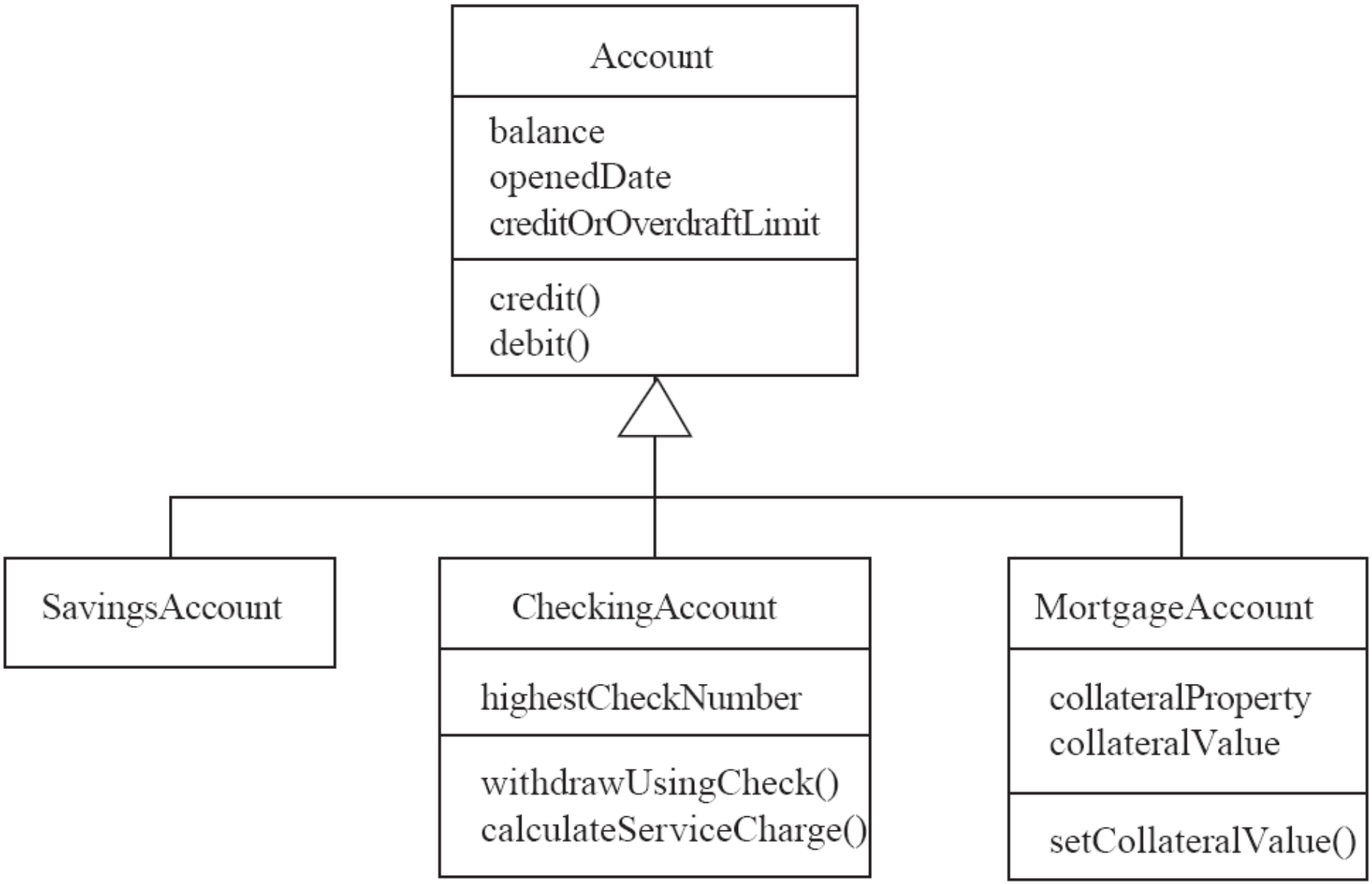
Always check generalizations to ensure they obey the isa rule

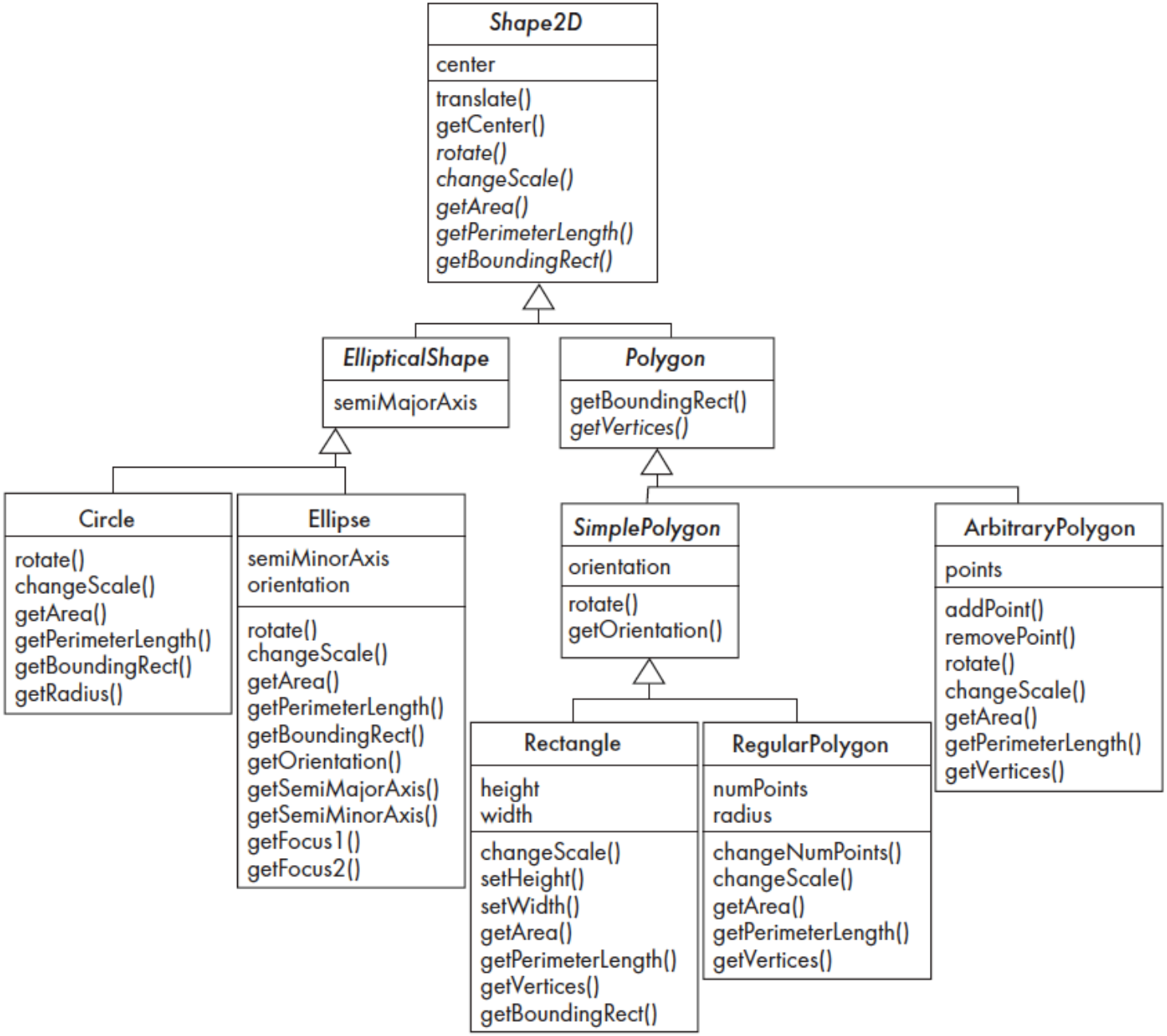
- "A checking account **is a**n account"
- "A village **is a** municipality"

Should 'Province' be a subclass of 'Country'?

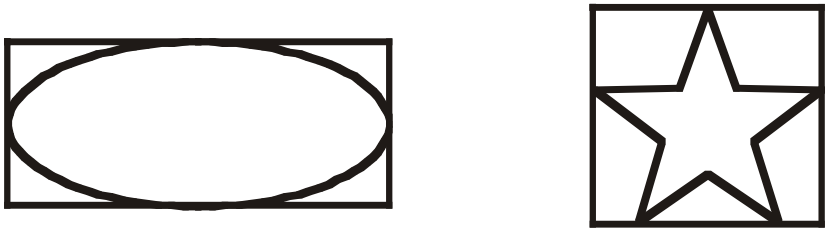

MathematicalObject
Shape
Point
Matrix
Shape2D
Shape3D
Ellipse
Polygon
Line
Plane
Circle
Quadrilateral
Rectangle
A possible inheritance hierarchy of mathematical objects

Make Sure all Inherited Features Make Sense in Subclasses

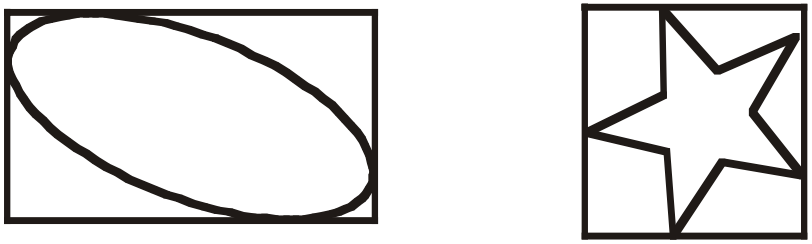

Account
balance
openedDate
creditOrOverdraftLimit
credit()
debit()
SavingsAccount
CheckingAccount
highestCheckNumber
withdrawUsingCheck()
calculateServiceCharge()
MortgageAccount
collateralProperty
collateralValue
setCollateralValue()


Shape2D
center
translate()
getCenter()
rotate()
changeScale()
getArea()
getPerimeterLength()
getBoundingRect()
EllipticalShape
semiMajorAxis
Polygon
getBoundingRect()
getVertices()
Circle
rotate()
changeScale()
getArea()
getPerimeterLength()
getBoundingRect()
getRadius()
Ellipse
semiMinorAxis
orientation
rotate()
changeScale()
getArea()
getPerimeterLength()
getBoundingRect()
getOrientation()
getSemiMajorAxis()
getSemiMinorAxis()
getFocus1()
getFocus2()
SimplePolygon
orientation
rotate()
getOrientation()
ArbitraryPolygon
points
addPoint()
removePoint()
rotate()
changeScale()
getArea()
getPerimeterLength()
getVertices()
Rectangle
height
width
changeScale()
setHeight()
setWidth()
getArea()
getPerimeterLength()
getVertices()
getBoundingRect()
RegularPolygon
numPoints
radius
changeNumPoints()
changeScale()
getArea()
getPerimeterLength()
getVertices()

Original objects
(showing bounding rectangle)



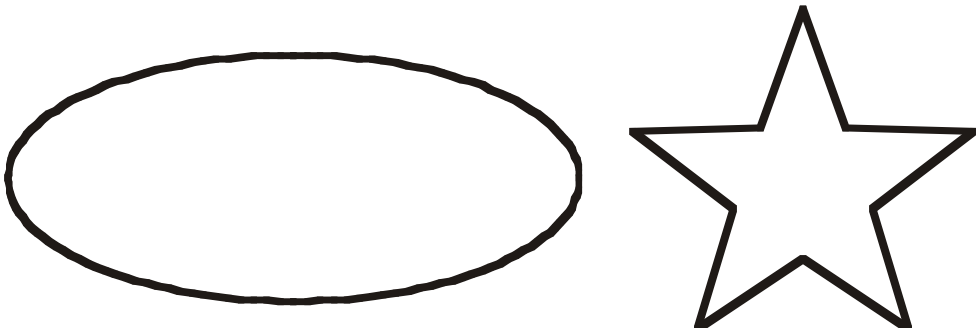
Rotated objects
(showing bounding rectangle)



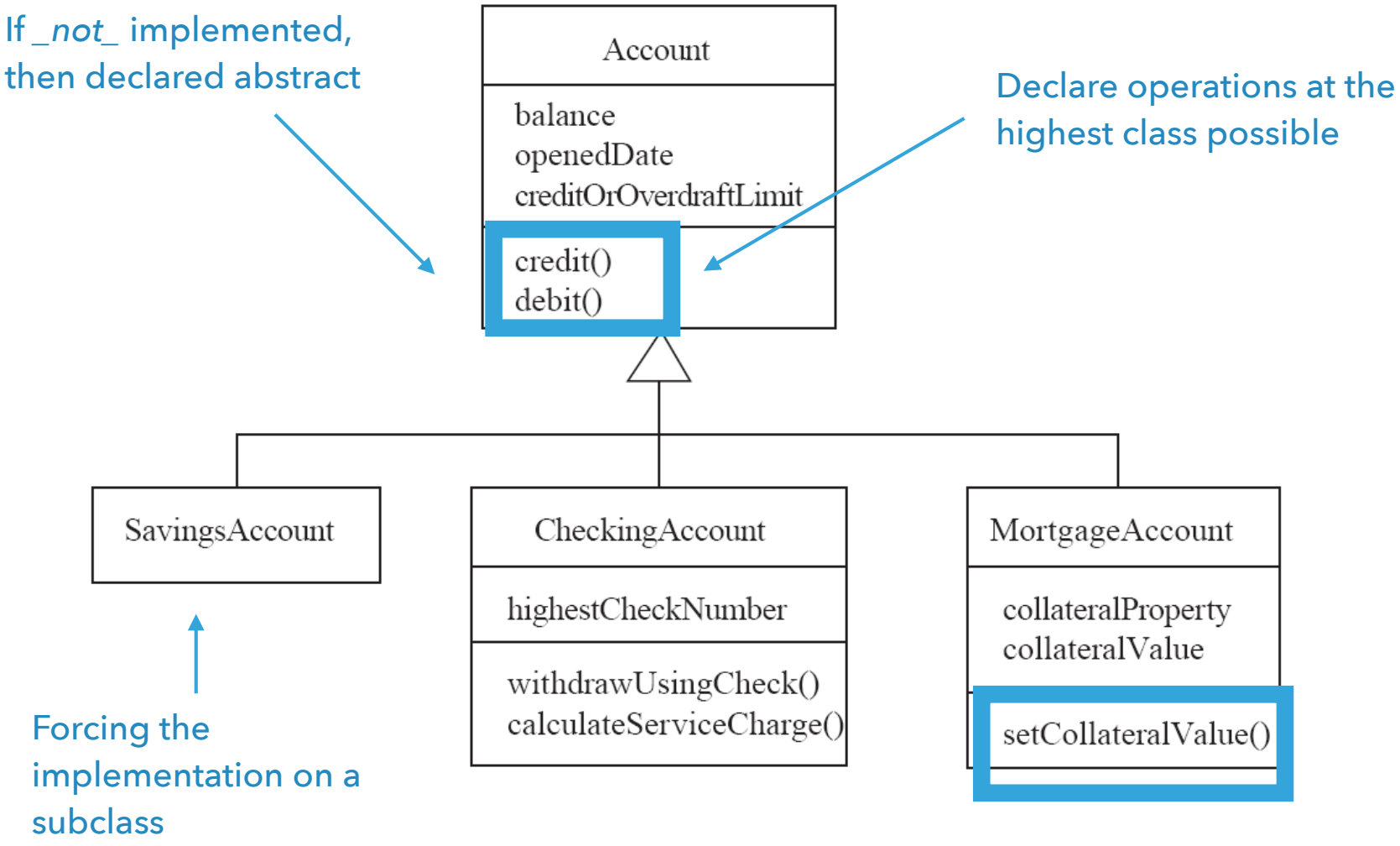
Translated objects
(showing original)

Scaled objects
(50%)

Scaled objects
(150%)



ABSTRACT CLASSES AND METHODS



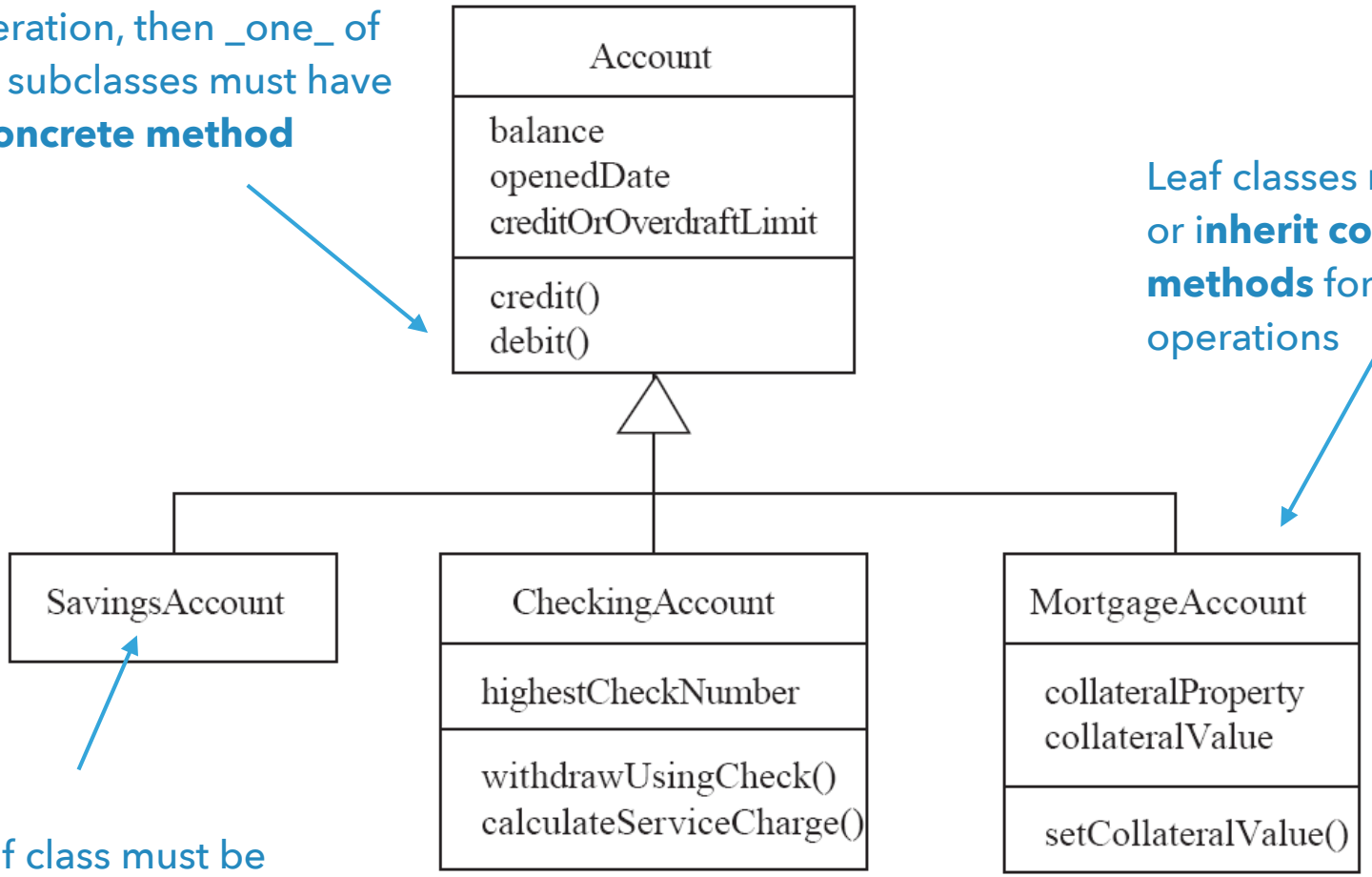
```
Shape.java:1: error: Shape is not abstract and does
not override abstract method perimeter() in Shape
public class Shape {
       ^
1 error
```

If class contains abstract methods, it must also be declared abstract

- **No instances** can be created
- The opposite of an abstract class is a **concrete class**

```
Circle.java:1: error: Circle is not abstract and does
 not override abstract method area() in Shape
public class Circle extends Shape {
       ^
```

If we have an **abstract** operation, then _one_ of the subclasses must have a **concrete method**



Leaf classes **must have** or i**nherit concrete methods** for all operations

Leaf class must be concrete

Overriding

- A method would be inherited, but a subclass contains a new version instead
- AVOID!!!
- But if asked on an exam, some reasons would be
  - For extension (SavingsAccount charges a fee for a debit)
  - For optimization (getPerimeterLength for Circle simpler than Ellipse)
  - For restriction (method might not make sense in subclass)

Method precedence (which one to run)

1. If there is a **concrete method** for the operation in the current class, run that method.
2. Otherwise, check in the i**mmediate superclass** to see if there is a method there; if so, run it.
3. **Repeat step 2**, looking in successively higher superclasses until a concrete method is found and run.
4. If **no method** is found, then there is an **error**
   - Java / C++ won't compile (compilation error)
   - Ruby / PHP throw a runtime error

DYNAMIC BINDING

- Occurs when **decision** about which method to run can only be **made at run time**

2+ polymorphic methods available

```
public static void main(String[] args) {
  double input = Double.parseDouble(args[0]);
  Shape s = new Circle(input);

  System.out.println("s1 Area: " + s.area());
  System.out.println("s1 Perimeter: " + s.perimeter());
}
```

Variable declared as it's superclass

Variable declared as it's superclass

KEY TERMINOLOGY

Abstraction

- Object -> something in the world
- Class -> objects
- Superclass -> subclasses
- Operation -> methods
- Attributes and associations -> instance variables

Modularity

- Code is divided into classes, and classes into methods

Encapsulation

- Details can be hidden in classes
- This gives rise to information hiding:
- Programmers do not need to know all the details of a class

NOVEMBER 11, 2002 *by* JOEL SPOLSKY

The Law of Leaky Abstractions

TOP 10, ROCK STAR DEVELOPER, NEWS

There's a key piece of magic in the engineering of the Internet which you rely on every single day. It happens in the TCP protocol, one of the fundamental building blocks of the Internet.

TCP is a way to transmit data that is *reliable*. By this I mean: if you send a message over a network using TCP, it will arrive, and it won't be garbled or corrupted.

https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/