Foundations of Software Engineering

Introduction to Unified Modeling Language

Dr. Petru Florin Mihancea

Tell me about ...

Tell me about ...

```java
public interface Expression {

    Expression computeDerivative();

}
```

```java
abstract public class BinaryExpression implements Expression {

    protected Expression left,right;

    public BinaryExpression(Expression st, Expression dr) {
        this.left = st;
        this.right = dr;
    }

    public void setLeft(Expression left) {
        this.left = left;
    }

    public void setRight(Expression right) {
        this.right = right;
    }
}
```

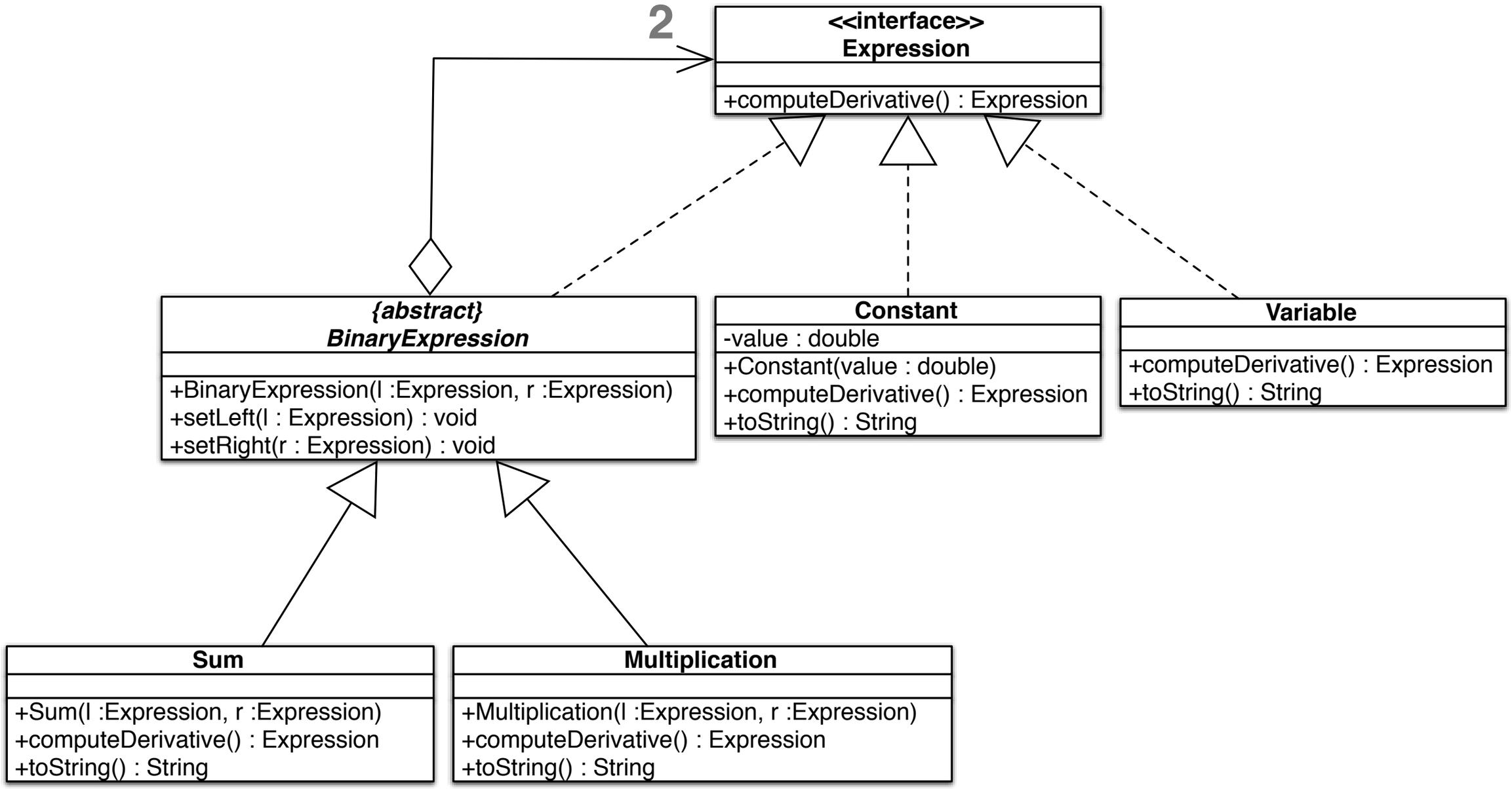
```java
public class Multiplication extends BinaryExpression {

    public Multiplication(Expression st, Expression dr) {
        super(st,dr);
    }

    public Expression computeDerivative() {
        Expression t1 = new Multiplication(left,right.computeDerivative());
        Expression t2 = new Multiplication(left.computeDerivative(),right);
        return new Sum(t1,t2);
    }

    public String toString() {
        return "(" + left.toString() + " * " + right.toString() + ")";
    }
}
```

```java
public class Variable implements Expression {

    public Expression computeDerivative() {
        return new Constant(1);
    }

    public String toString() {
        return "x";
    }
}
```

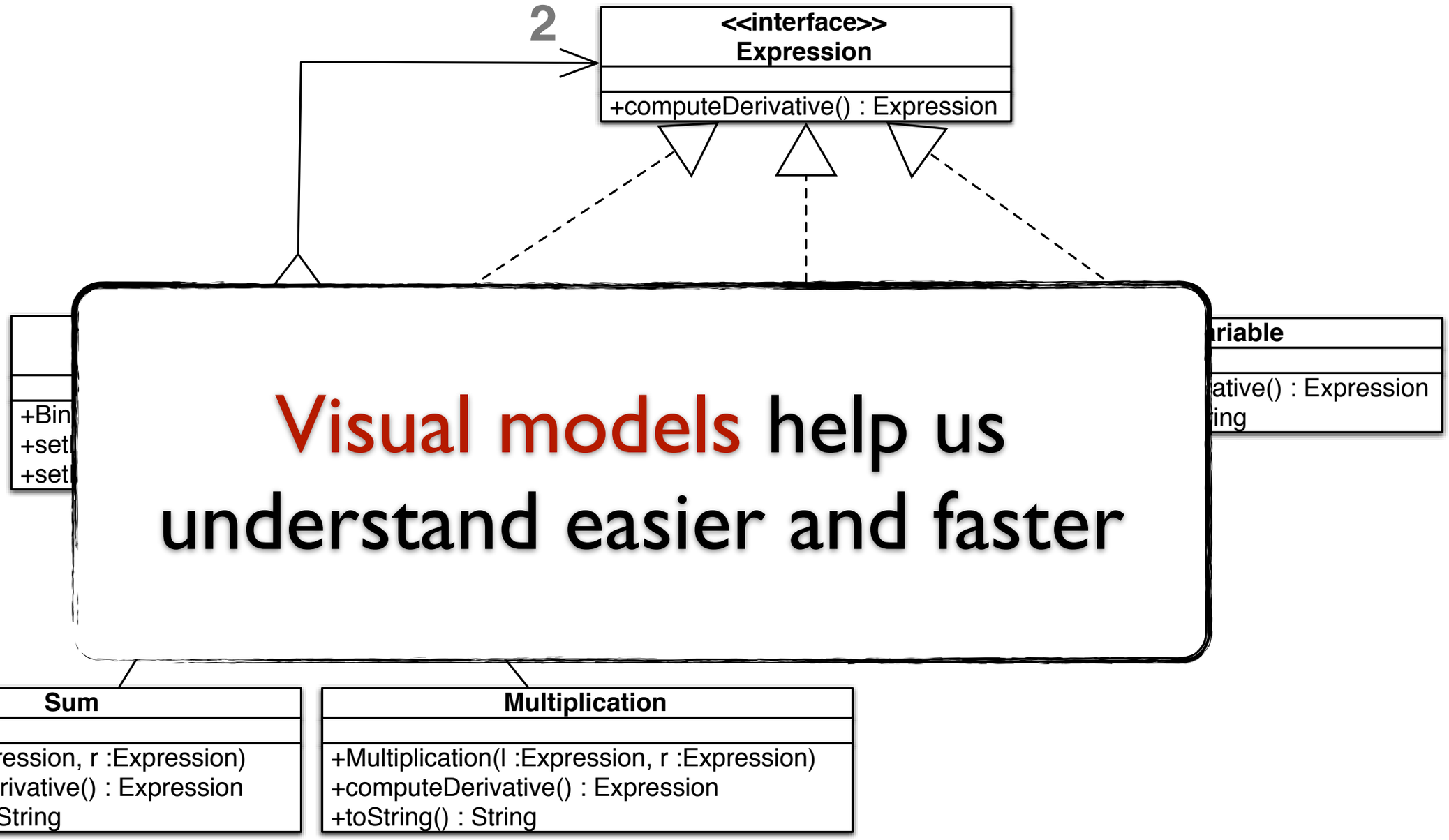
```java
public class Sum extends BinaryExpression {

    public Sum(Expression st, Expression dr) {
        super(st,dr);
    }

    public Expression computeDerivative() {
        return new Sum(left.computeDerivative(),right.computeDerivative());
    }

    public String toString() {
        return "(" + left.toString() + " + " + right.toString() + ")";
    }
}
```

```java
public class Constant implements Expression {

    private double value;

    public Constant(double a) {
        this.value = a;
    }

    public Expression computeDerivative() {
        return new Constant(0);
    }

    public String toString() {
        return value + "";
    }

}
```

Tell me about ...



Tell me about ...



Visual models help us understand easier and faster

Unified Modeling Language

Unified Modeling Language

Family of **graphical** notations

for **modeling** an (OO) system

Unified Modeling Language

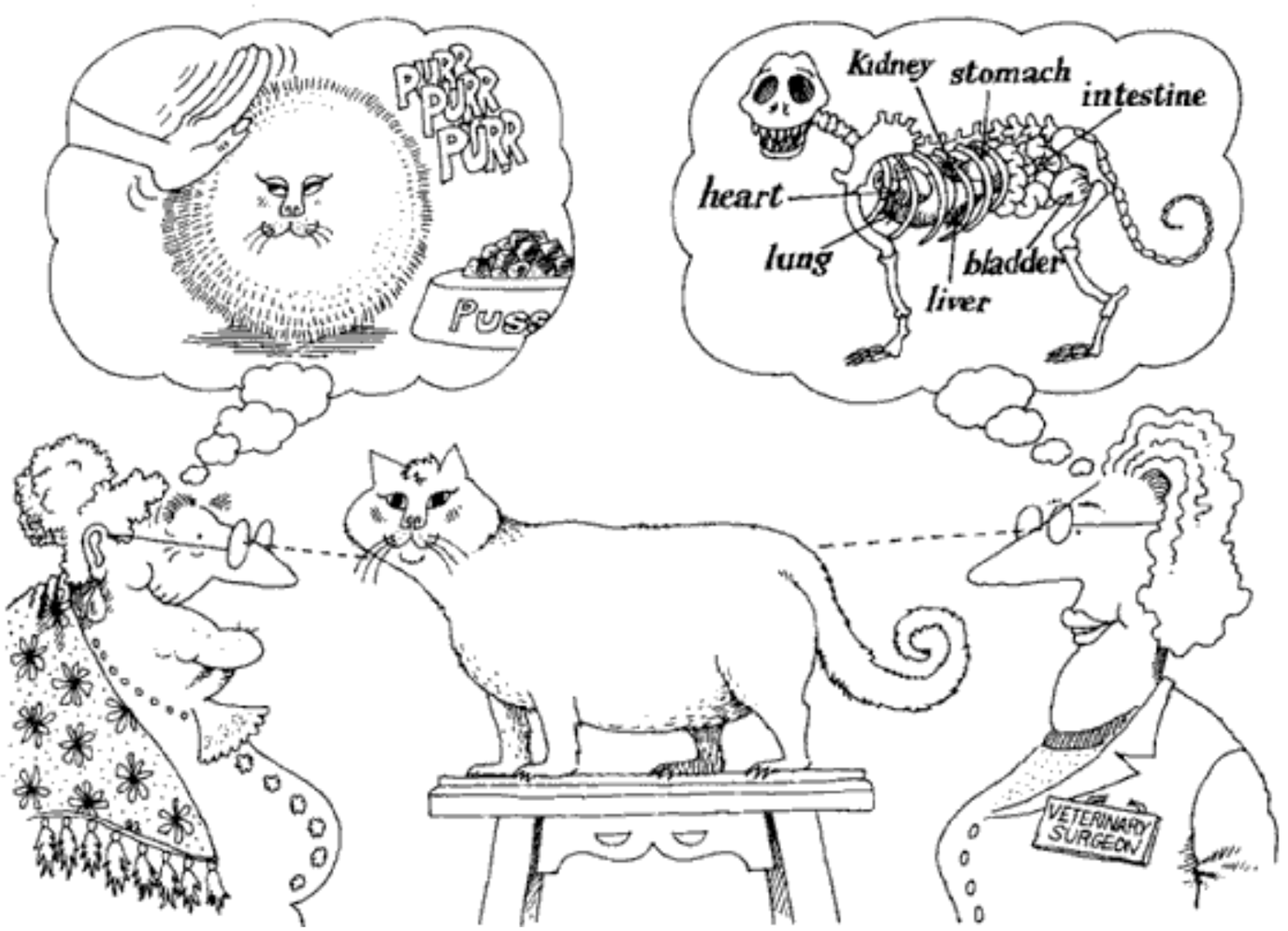
Family of **graphical** notations

for **modeling** an (OO) system

Goal

Learn the notations

Types of UML models

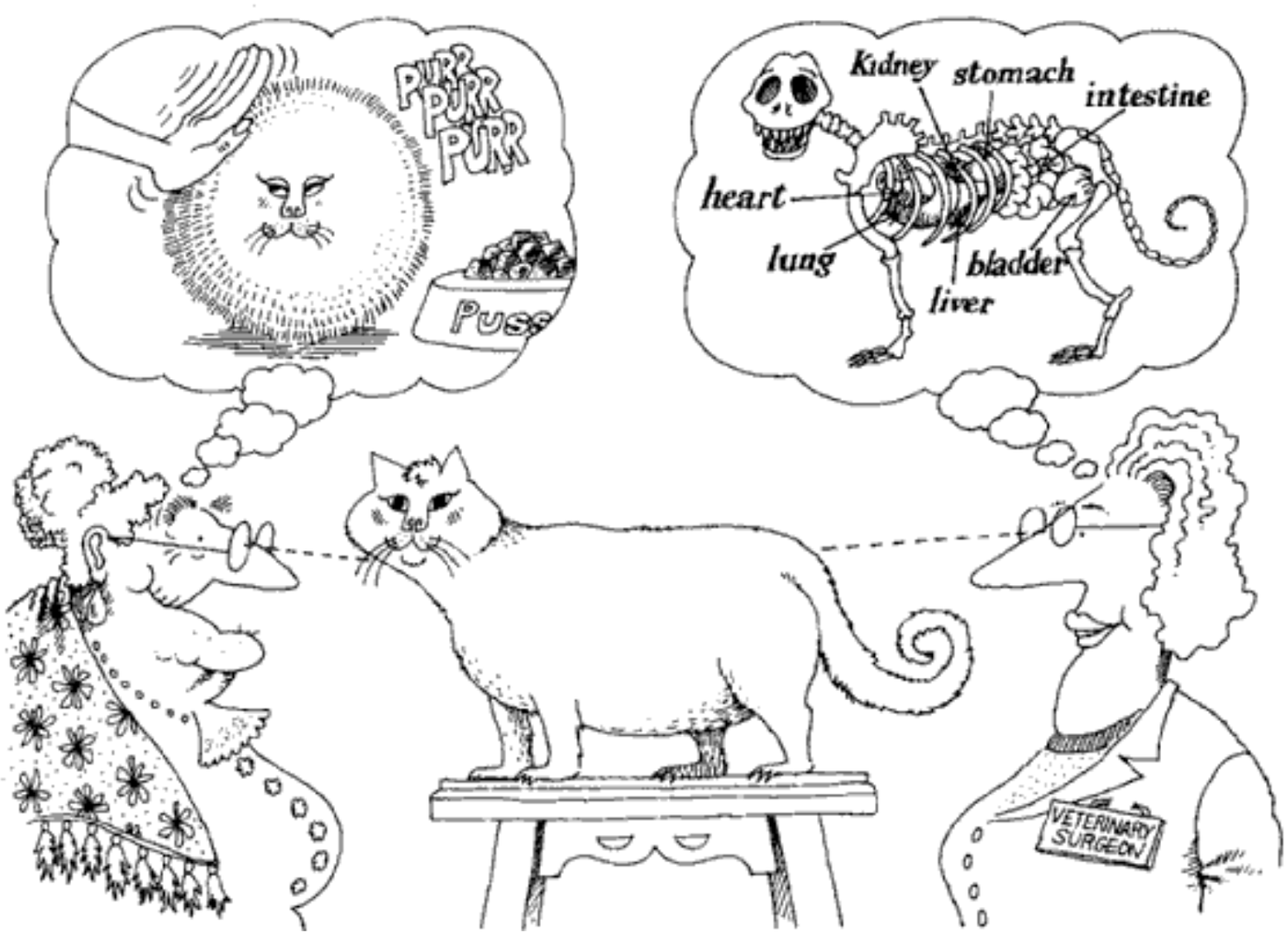


Booch - OO Analysis and Design

Types of UML models

Structural

e.g. Class diagram (CD)



Booch - OO Analysis and Design

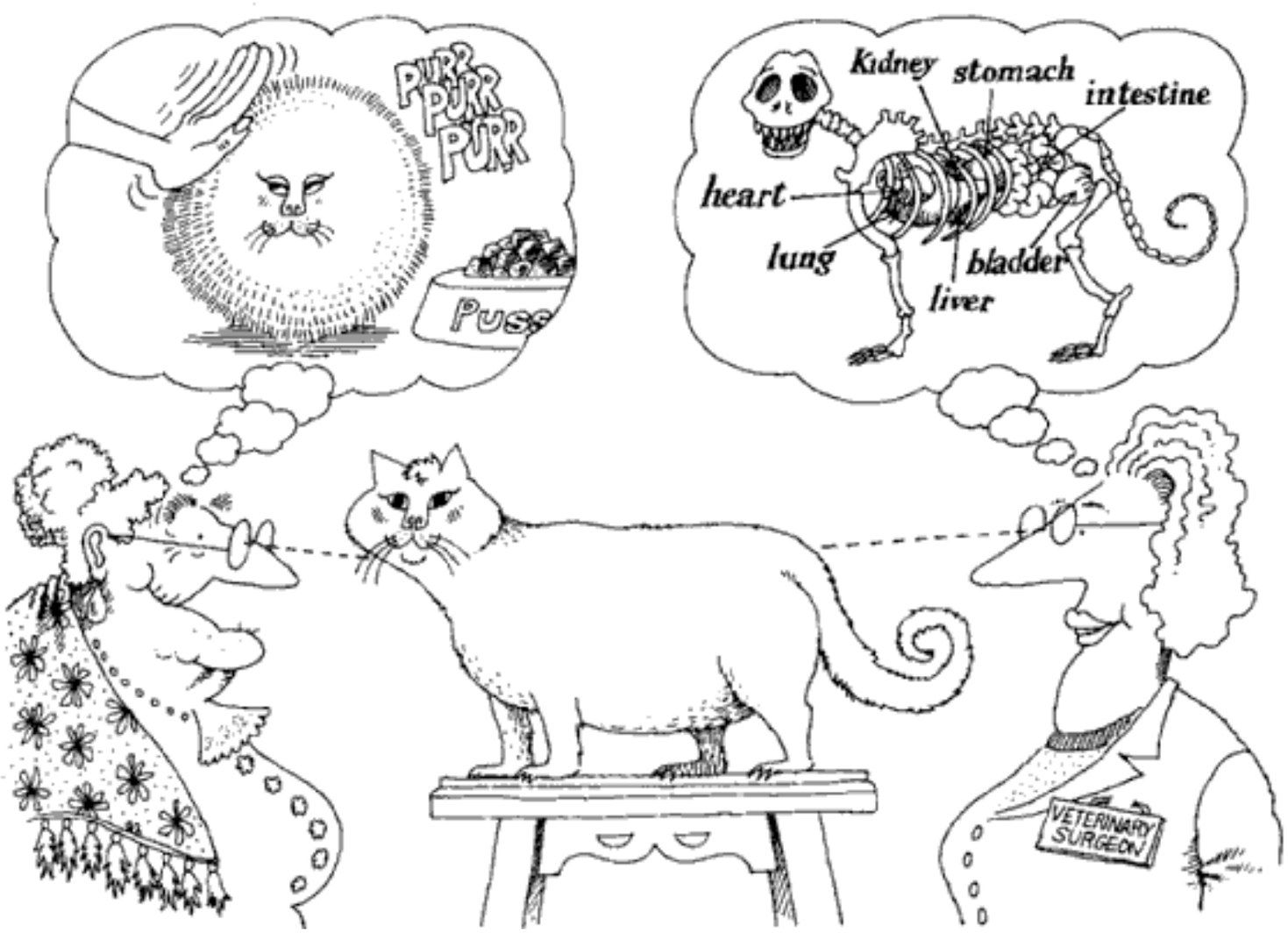
Types of UML models

Behavioral

e.g. Sequence diagram (SD)

Structural

e.g. Class diagram (CD)



Booch - OO Analysis and Design

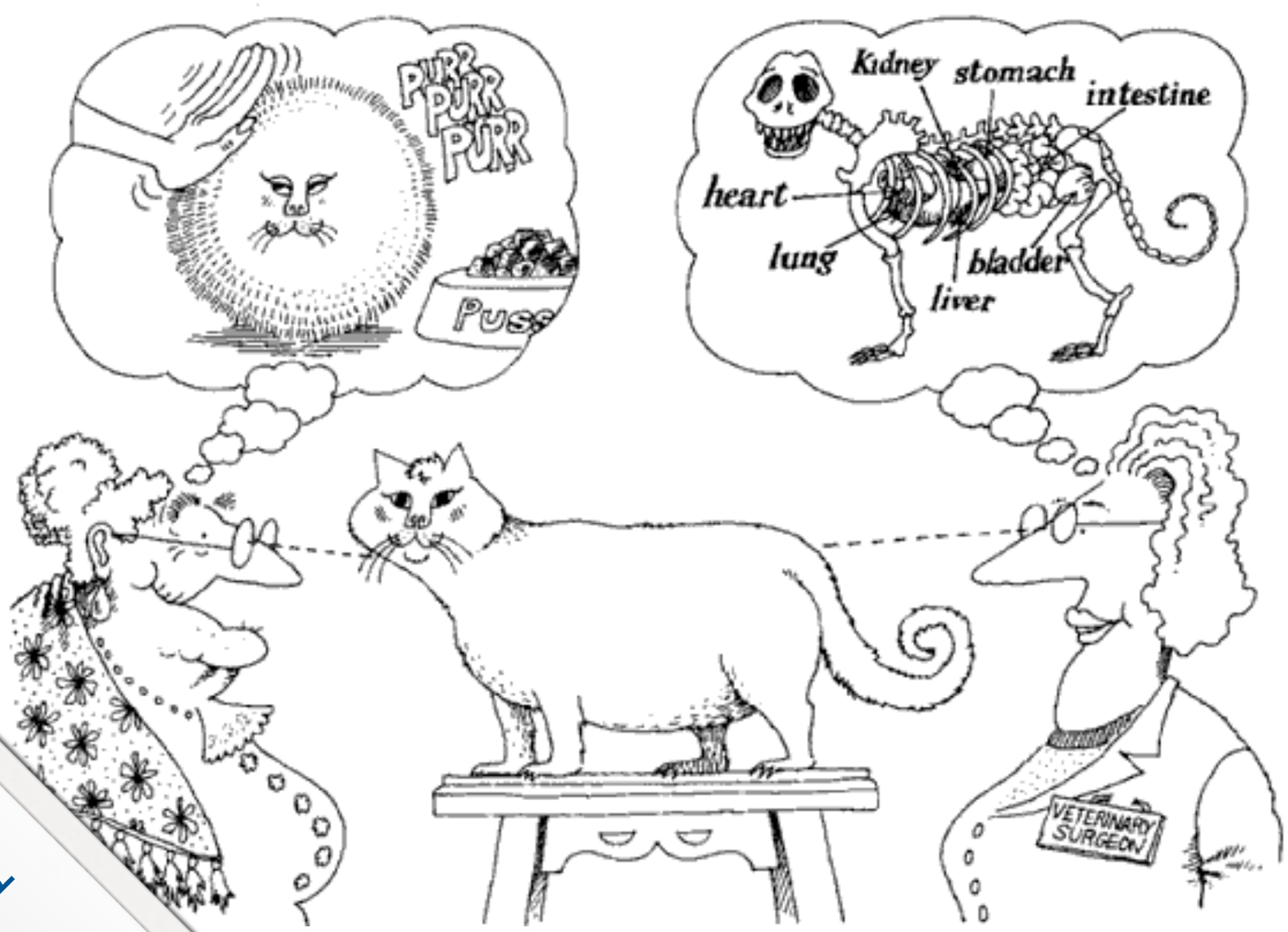
Types of UML models

Behavioral

e.g. Sequence diagram (SD)

Structural

e.g. Class diagram (CD)



Booch - OO Analysis and Design

Goal Refined

Learn the notations for CD and SD

UML usage perspectives

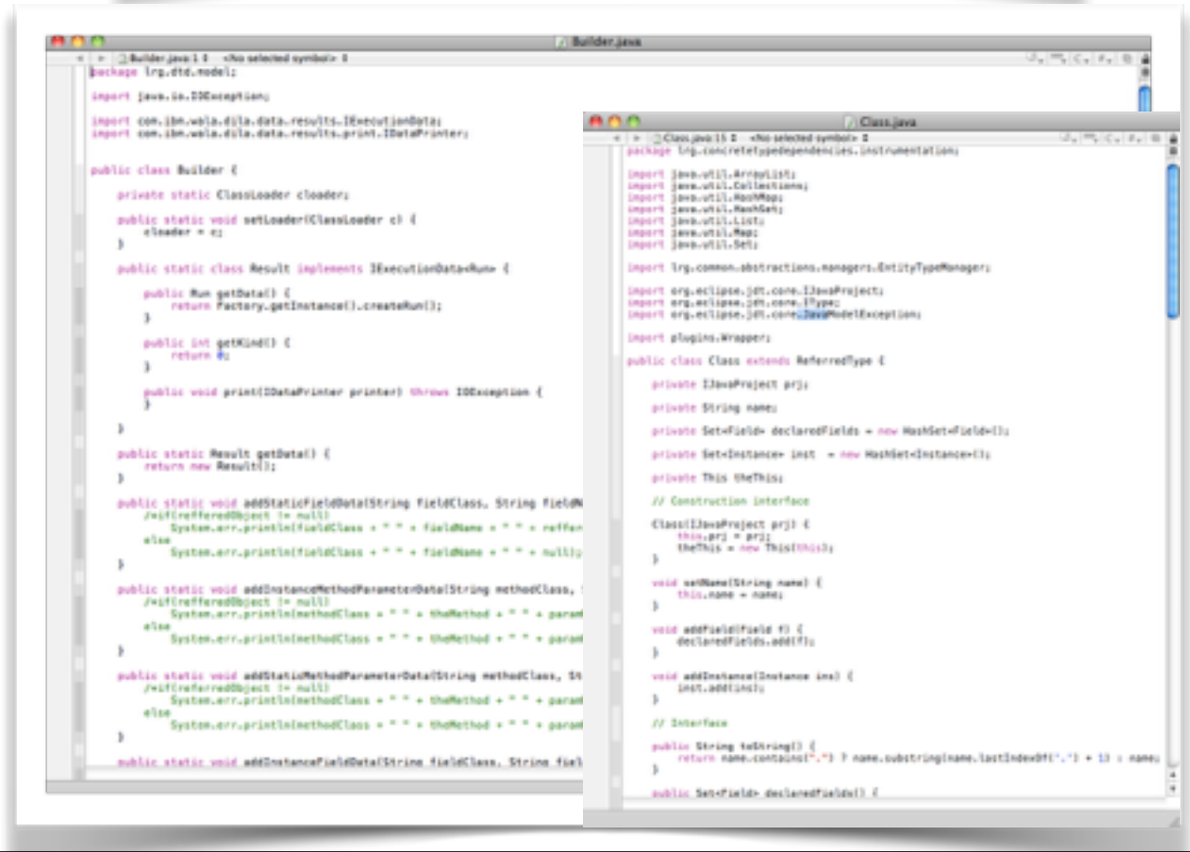
UML usage perspectives

UML usage perspectives

Conceptual

i.e. model a domain

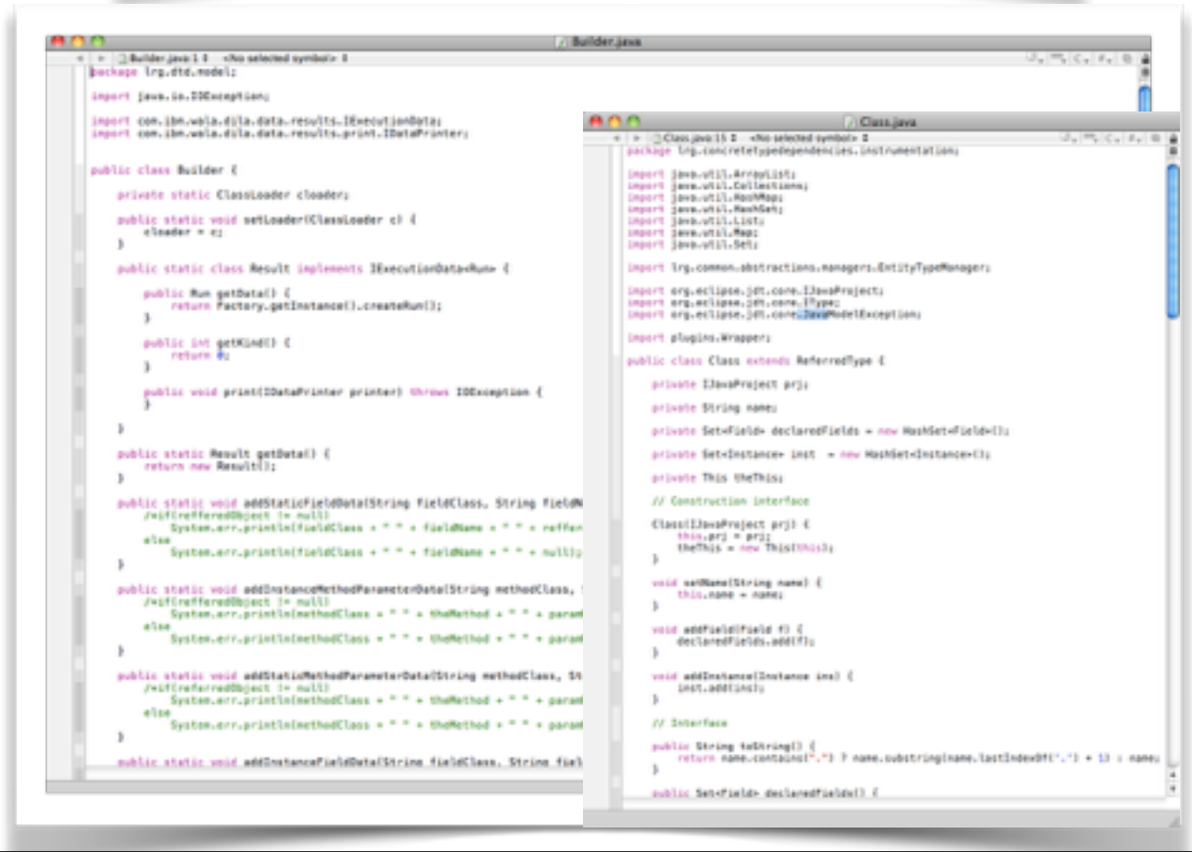
UML usage perspectives



Conceptual

i.e. model a domain

UML usage perspectives



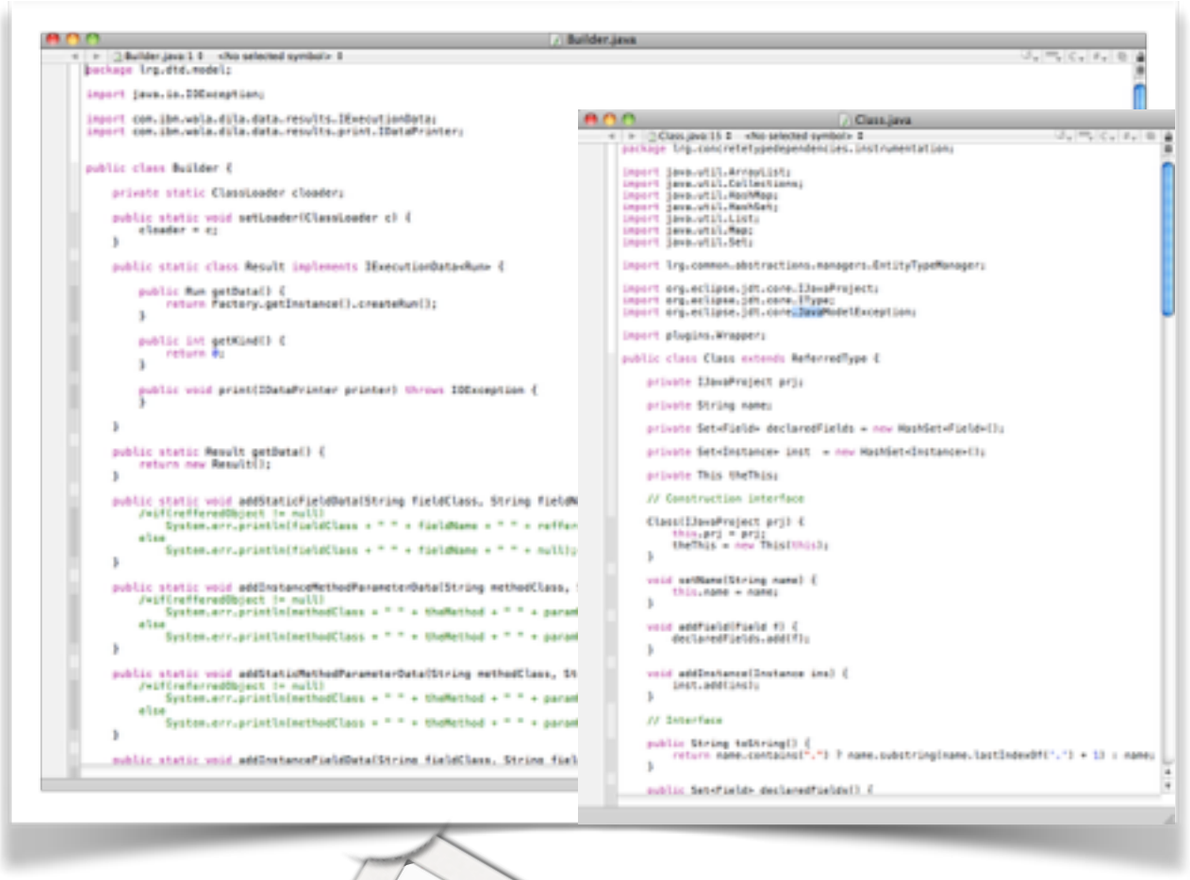
Conceptual

i.e. model a domain

Software

i.e. model a program

UML usage perspectives



Goal Refined

Learn notations for **CD** and **SD** and their usual meaning in Java source code

Software

i.e. model a program

1

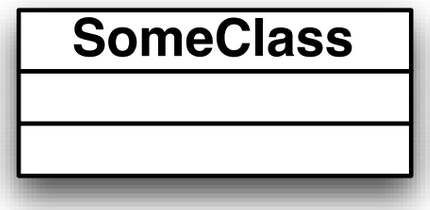
Class diagram

Structural model

1

Class diagram

Structural model

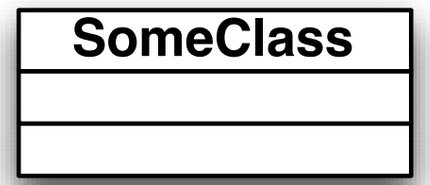


Classes

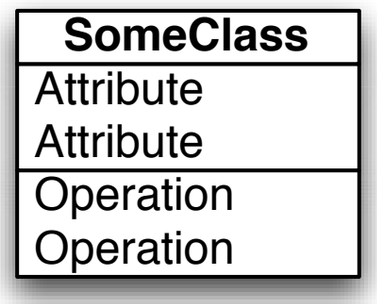
1

Class diagram

Structural model



Classes



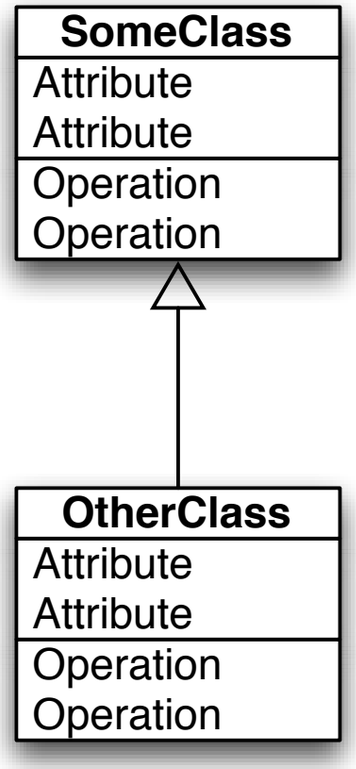
Features

Class diagram

Structural model

Classes

Features



Relations

1

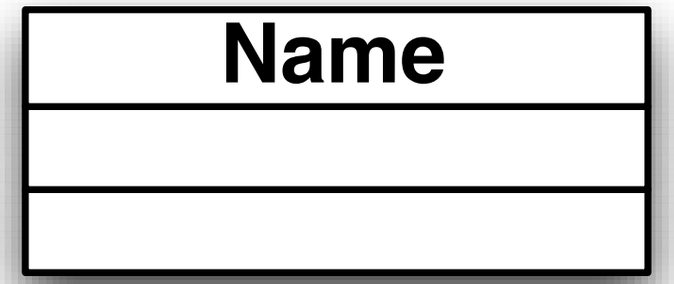
Class diagram

UML sketch

Java sketch

1

Class diagram

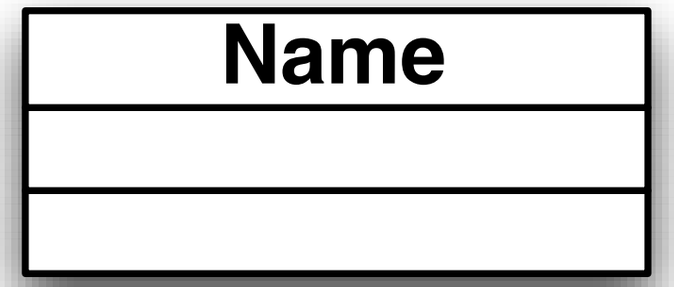


UML sketch

Java sketch

1

Class diagram

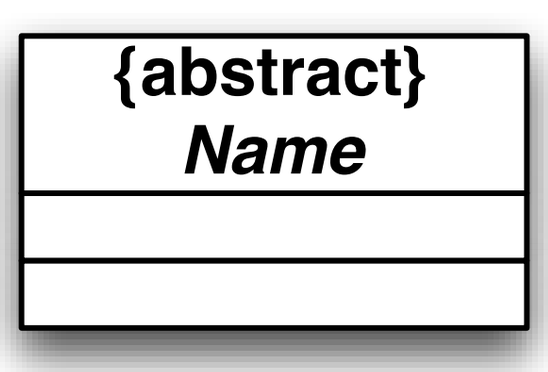


UML sketch

```
class Name {


}
```

Java sketch

Class diagram

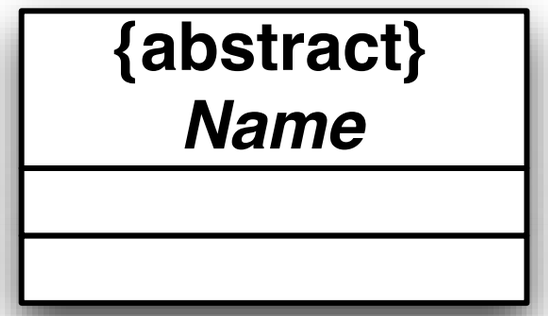


UML sketch

Java sketch

1

Class diagram



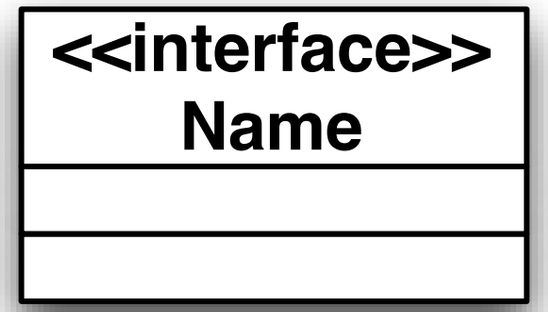

```
abstract class Name {

}
```

UML sketch

Java sketch

1

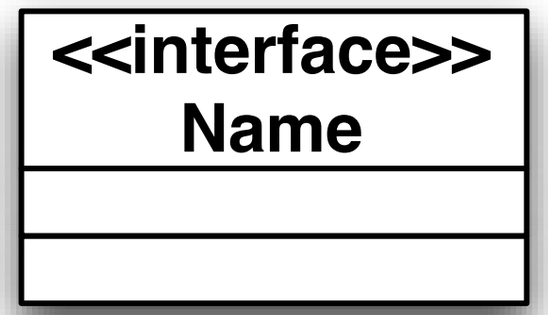
Class diagram



UML sketch

Java sketch

Class diagram



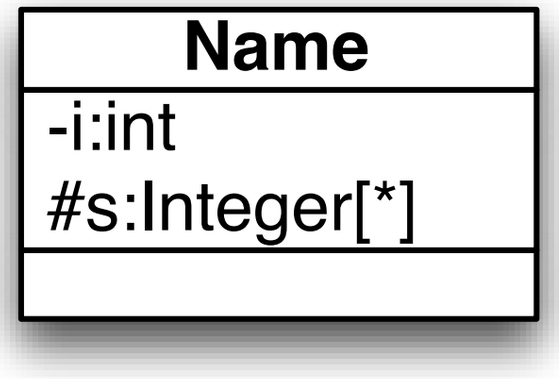

```
interface Name {

}
```

UML sketch

Java sketch

Class diagram

Attributes



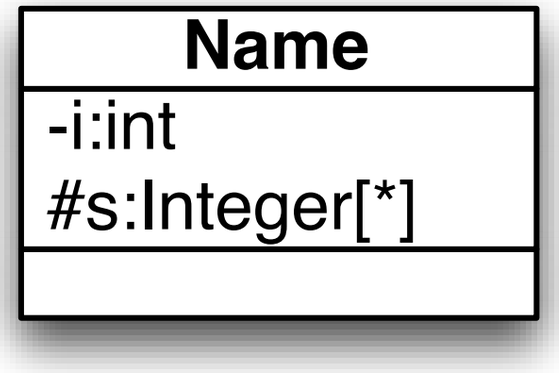
visibility name : type multiplicity
= implicitValue

UML sketch

Java sketch

Class diagram

Attributes



visibility name : type multiplicity
= implicitValue

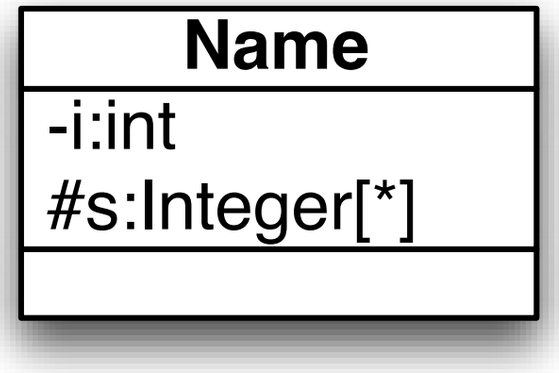
+ public
- private
protected
~ package

UML sketch

Java sketch

Class diagram

Attributes



visibility name : type multiplicity
= implicitValue

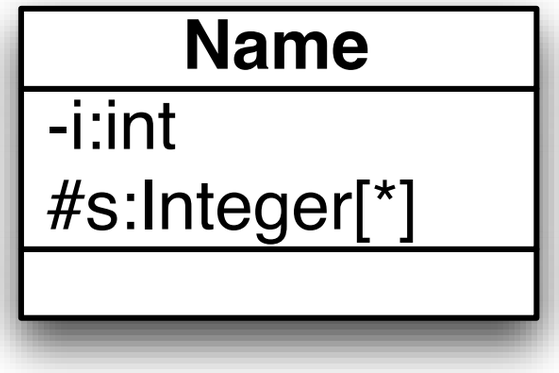
1 - exactly one
0..1 - zero or at most one
0..* or * - zero or more but
NO upper limit

UML sketch

Java sketch

Class diagram

Attributes




```
class Name {
    private int i;
    protected List<Integer> s;
    //s must be somehow initialized / created
}
```

visibility name : type multiplicity
= implicitValue

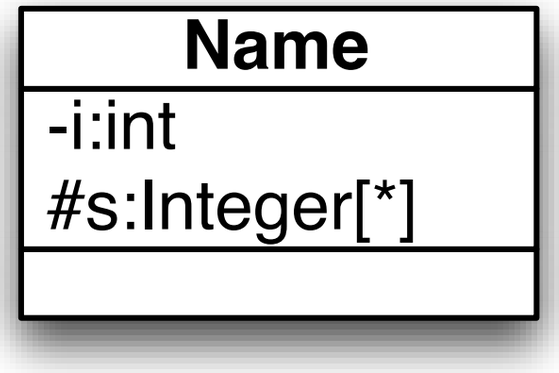
1 - exactly one
0..1 - zero or at most one
0..* or * - zero or more but
NO upper limit

UML sketch

Java sketch

Class diagram

Attributes



visibility name : type multiplicity
= implicitValue

1 - exactly one
0..1 - zero or at most one
0..* or * - zero or more but
NO upper limit

```
class Name {
   private int i;
   protected List<Integer> s;
   //s must be somehow initialized / created
}
```

```
class Name {
   private int i;
   protected Integer[] s;
   //s must be somehow initialized / created
   //an index may be required + you must
   //guarantee NO upper limit if necessary
   //(e.g. re-create & copy the array)
}
```

UML sketch

Java sketch

Class diagram

Operations

Name
+set(n:int):int

visibility name(param_list) : ret_type

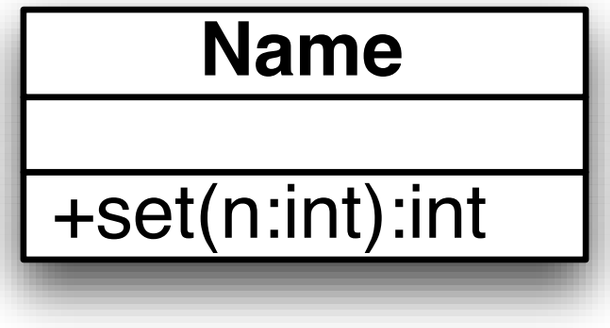
direction name : type = default

UML sketch

Java sketch

Class diagram

Operations



visibility name(param_list) : ret_type

direction name : type = default

```
class Name {
   public int set(int n) {
      ...
   }
}
```

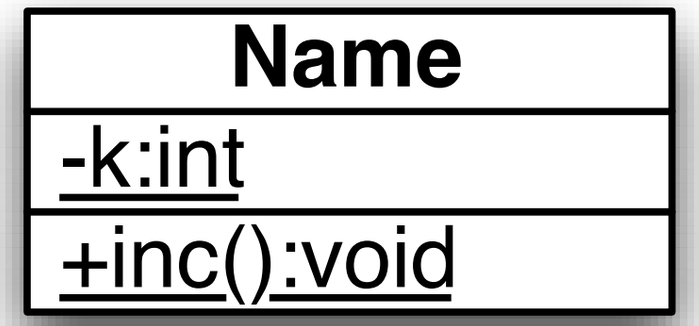
UML sketch

Java sketch

1

Class diagram

Scope

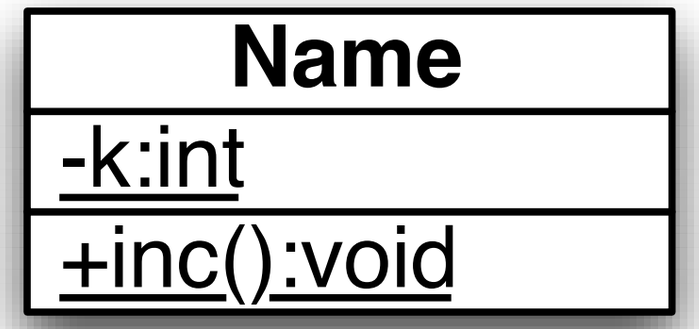


UML sketch

Java sketch

Class diagram

Scope




```
class Name {
   private static int k;
   public static void inc() {
      ...
   }
}
```

UML sketch

Java sketch

1

Class diagram

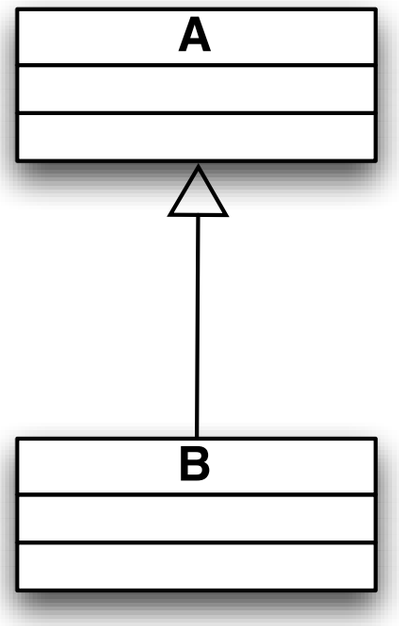
Generalization & Realization

UML sketch

Java sketch

Class diagram

Generalization & Realization

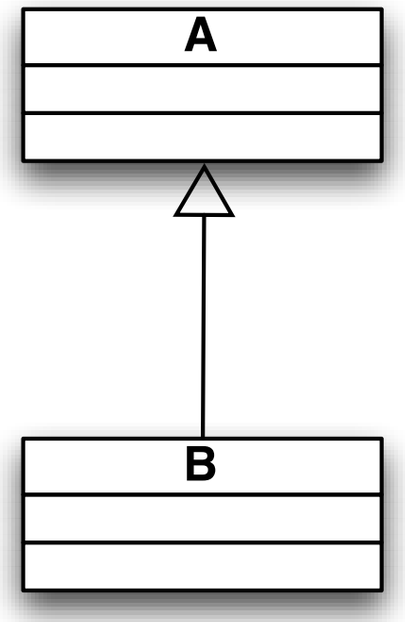


UML sketch

Java sketch

Class diagram

Generalization & Realization



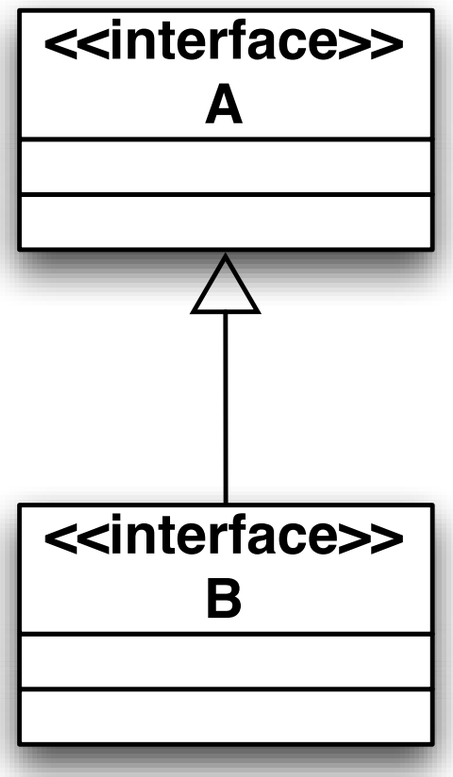
UML sketch

```
class A {
}

class B extends A {
}
```

Java sketch

Class diagram

Generalization & Realization

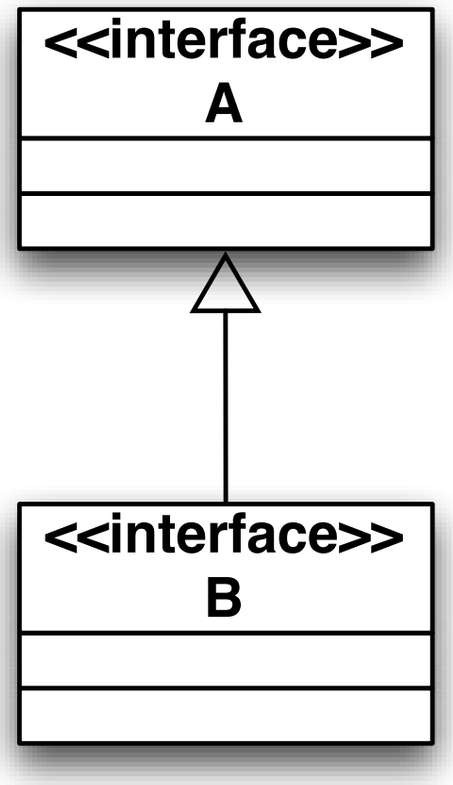


UML sketch

Java sketch

Class diagram

Generalization & Realization



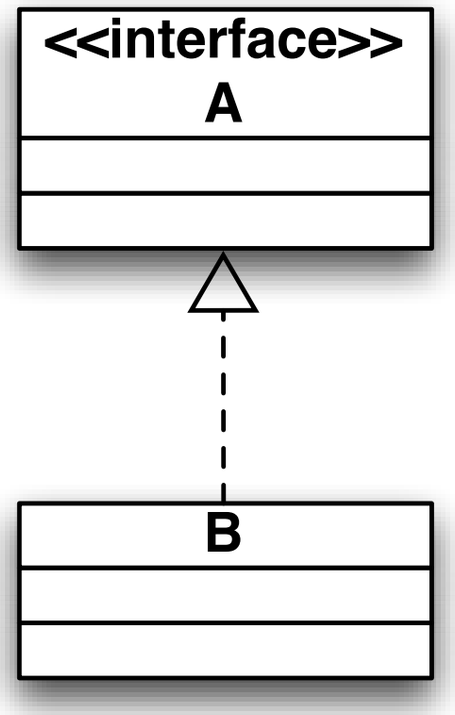
UML sketch

```
interface A {
}

interface B extends A {
}
```

Java sketch

Class diagram

Generalization & Realization

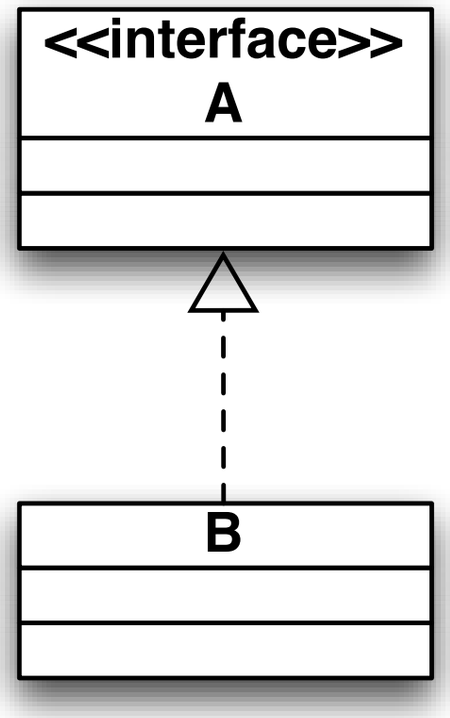


UML sketch

Java sketch

Class diagram

Generalization & Realization



UML sketch

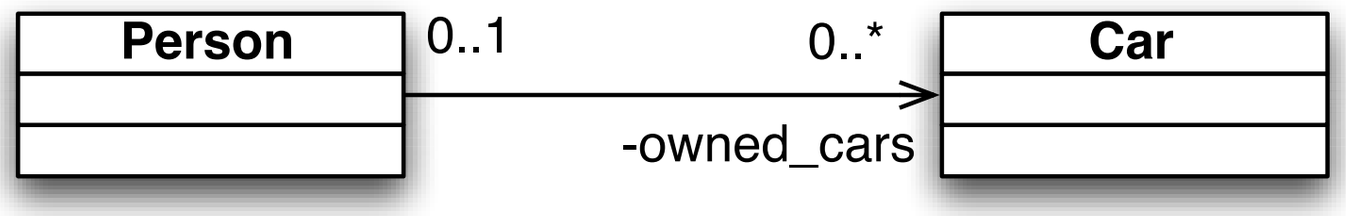
```
interface A {
}

class B implements A {
}
```

Java sketch

1

Class diagram

Association

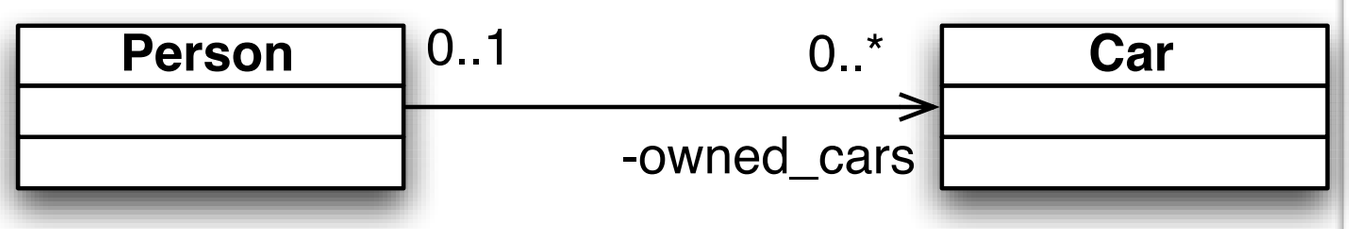


UML sketch

Java sketch

Class diagram

Association




```
class Person {

    //list must be somehow initialized / created
    private List<Car> owned_car;

    //add, remove methods usually exist

}
```

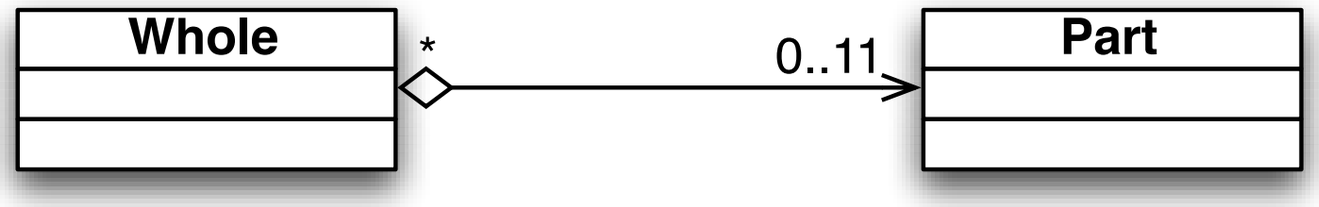
UML sketch

Java sketch

1

Class diagram

Aggregation

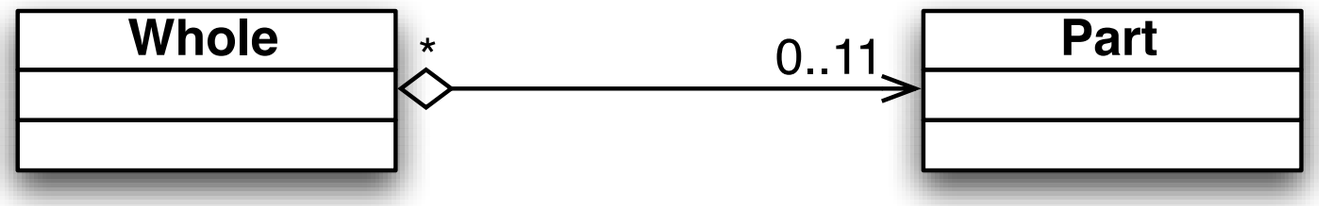


UML sketch

Java sketch

Class diagram

Aggregation



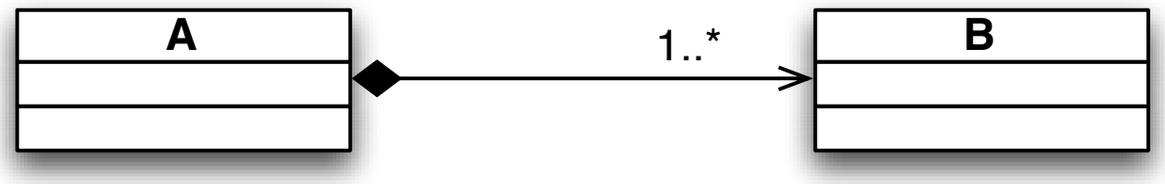
Similar to association

UML sketch

Java sketch

Class diagram

Composition



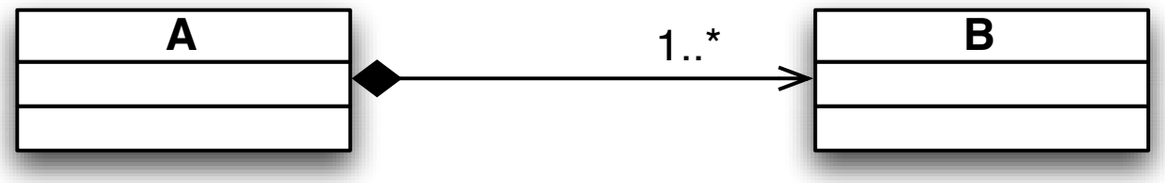
I. No-sharing
II. B objects cannot exists without their A object

UML sketch

Java sketch

Class diagram

Composition



I. No-sharing
II. B objects cannot exists without their A object

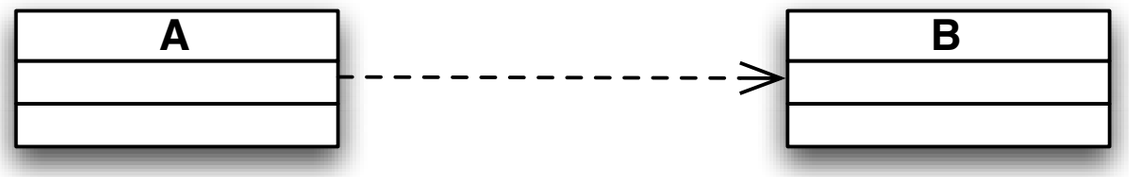
```
class  A {

    private List<B> my_list =
        new ...

    public A(...) {
        my_list.add(new B(...));
    }
    public void add(...) {
        my_list.add(new B(...));
    }
```

UML sketch

Java sketch

Class diagram

Dependency

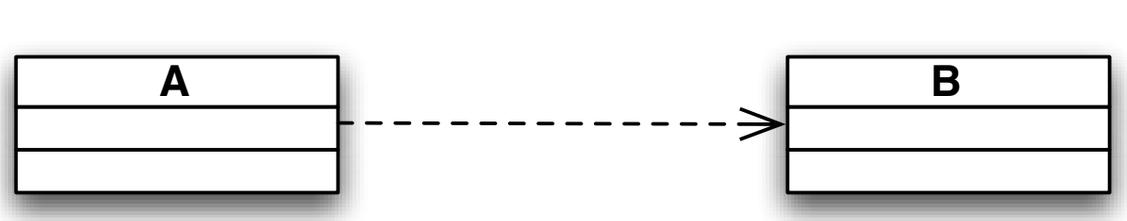


UML sketch

Java sketch

Class diagram

Dependency



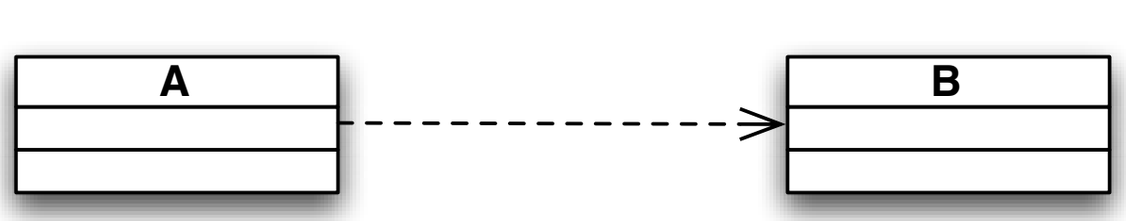
UML sketch

```
class A {

  public void m(B x) {
    x.doS();
  }

}
```

Java sketch

Class diagram

Dependency



UML sketch

```
class A {

  public void m(B x) {
    x.doS();
  }

}
```

Java sketch

And many other cases ...

2

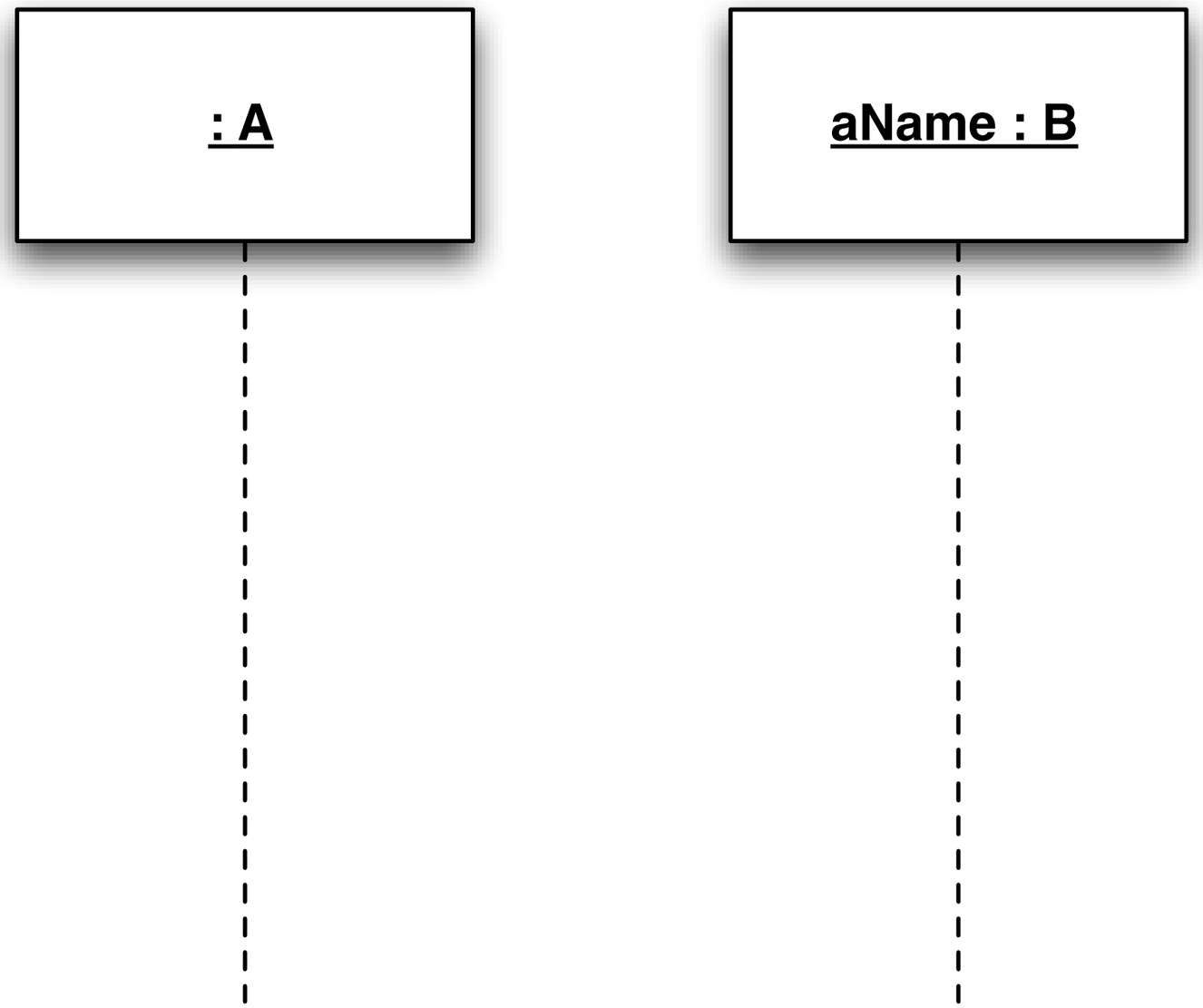
Sequence diagram

Behavioral & Interaction model

2

Sequence diagram

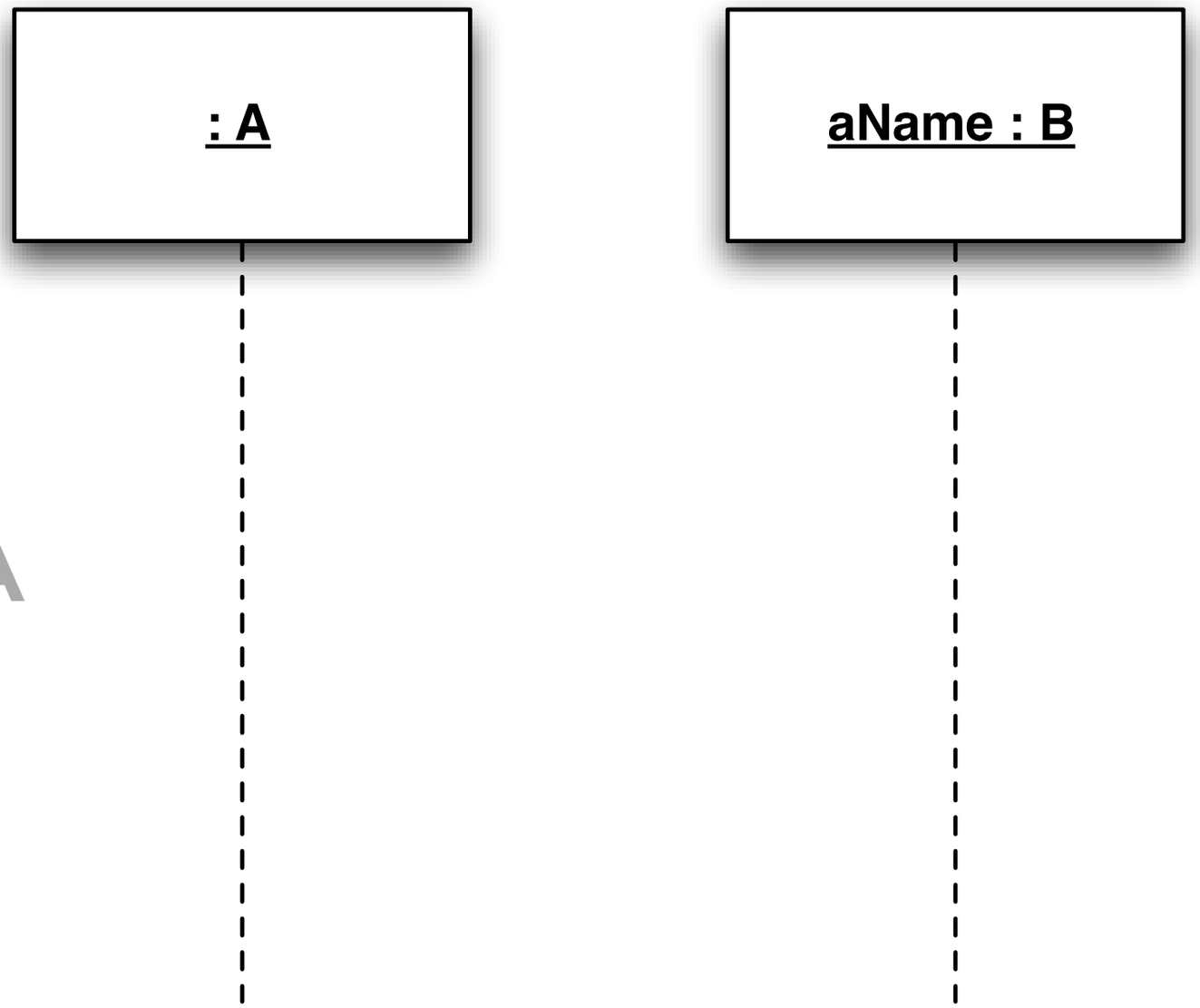
Behavioral & Interaction model



2

Sequence diagram

Behavioral & Interaction model

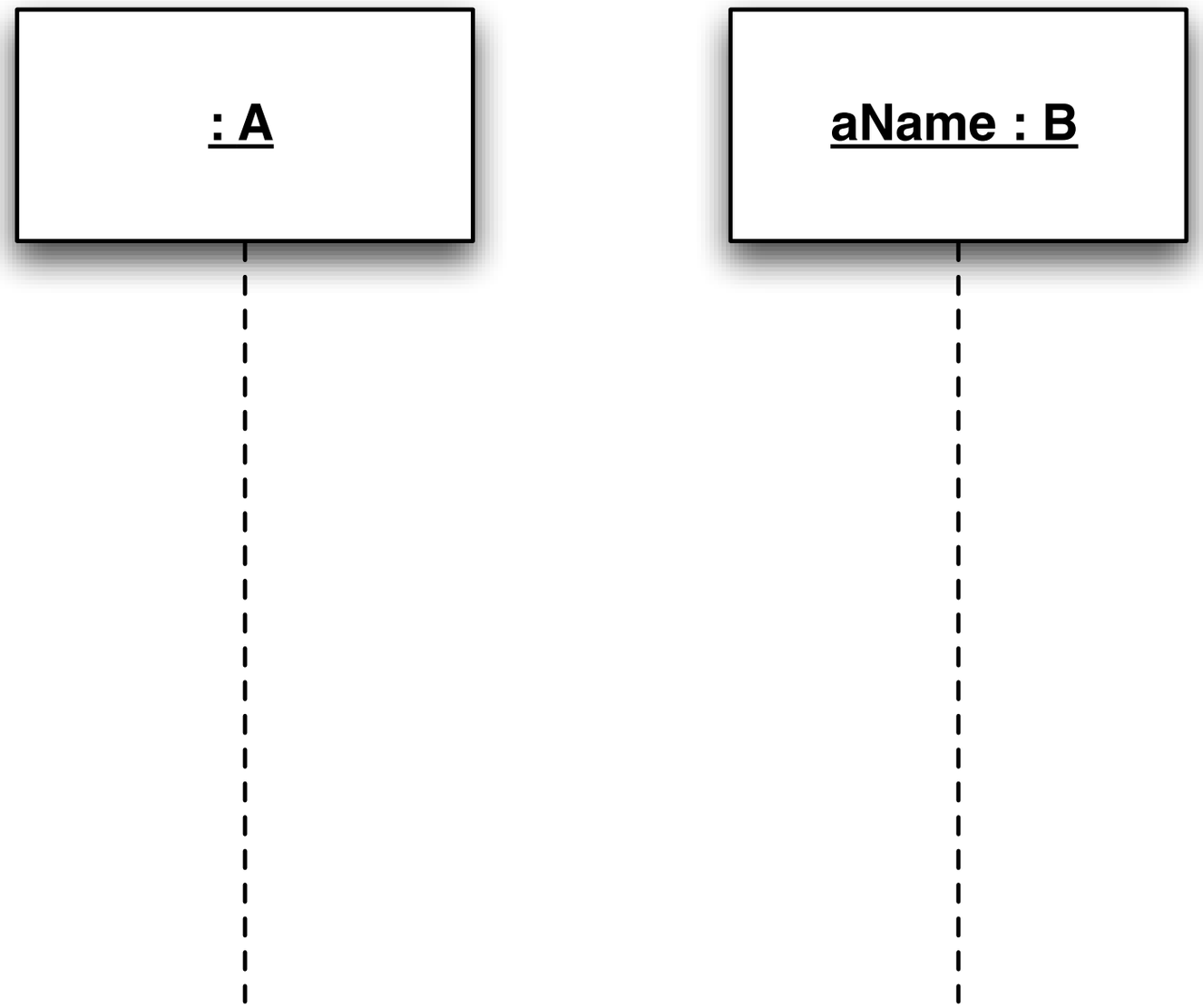


Object of A class

2

Sequence diagram

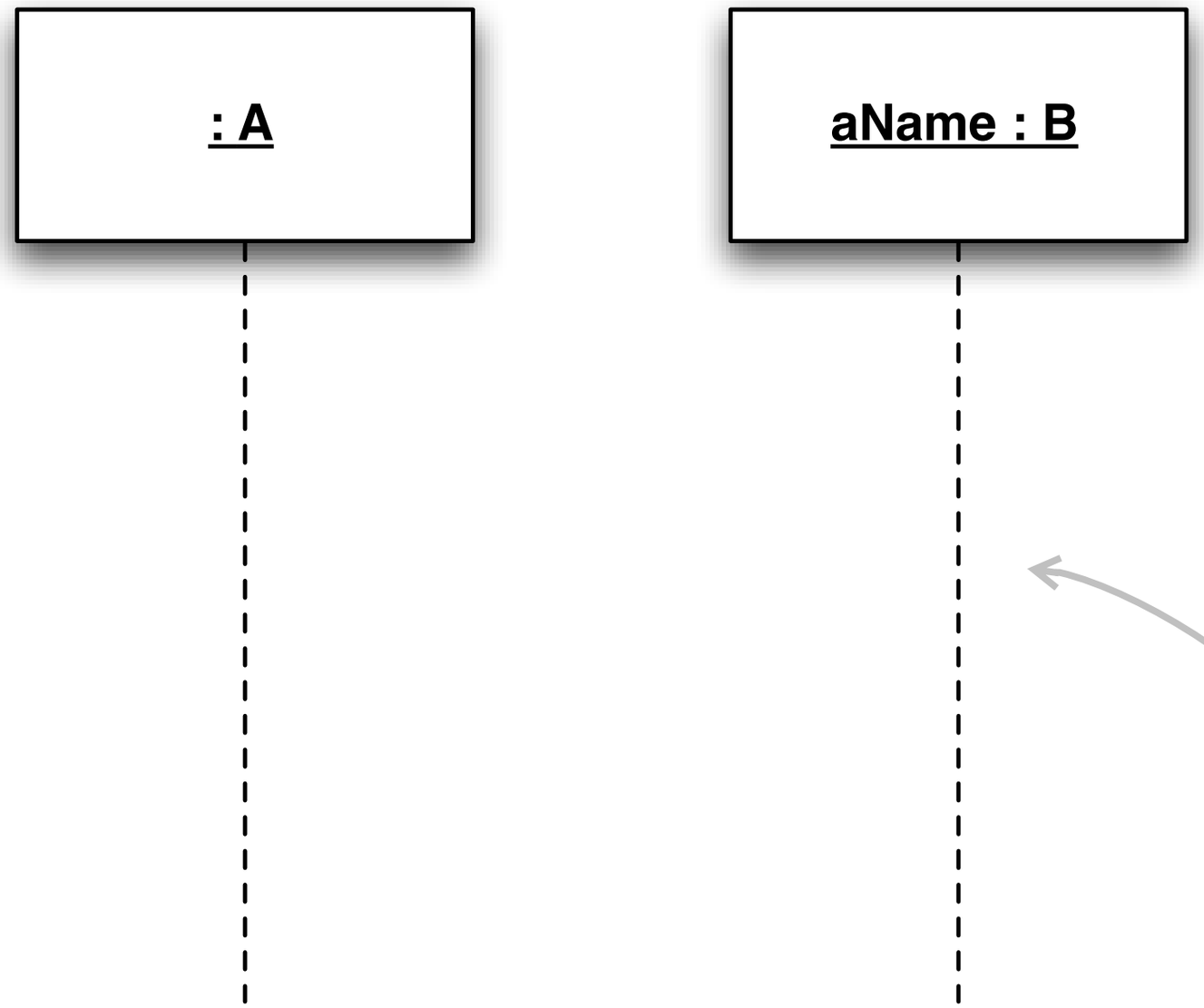
Behavioral & Interaction model



2

Sequence diagram

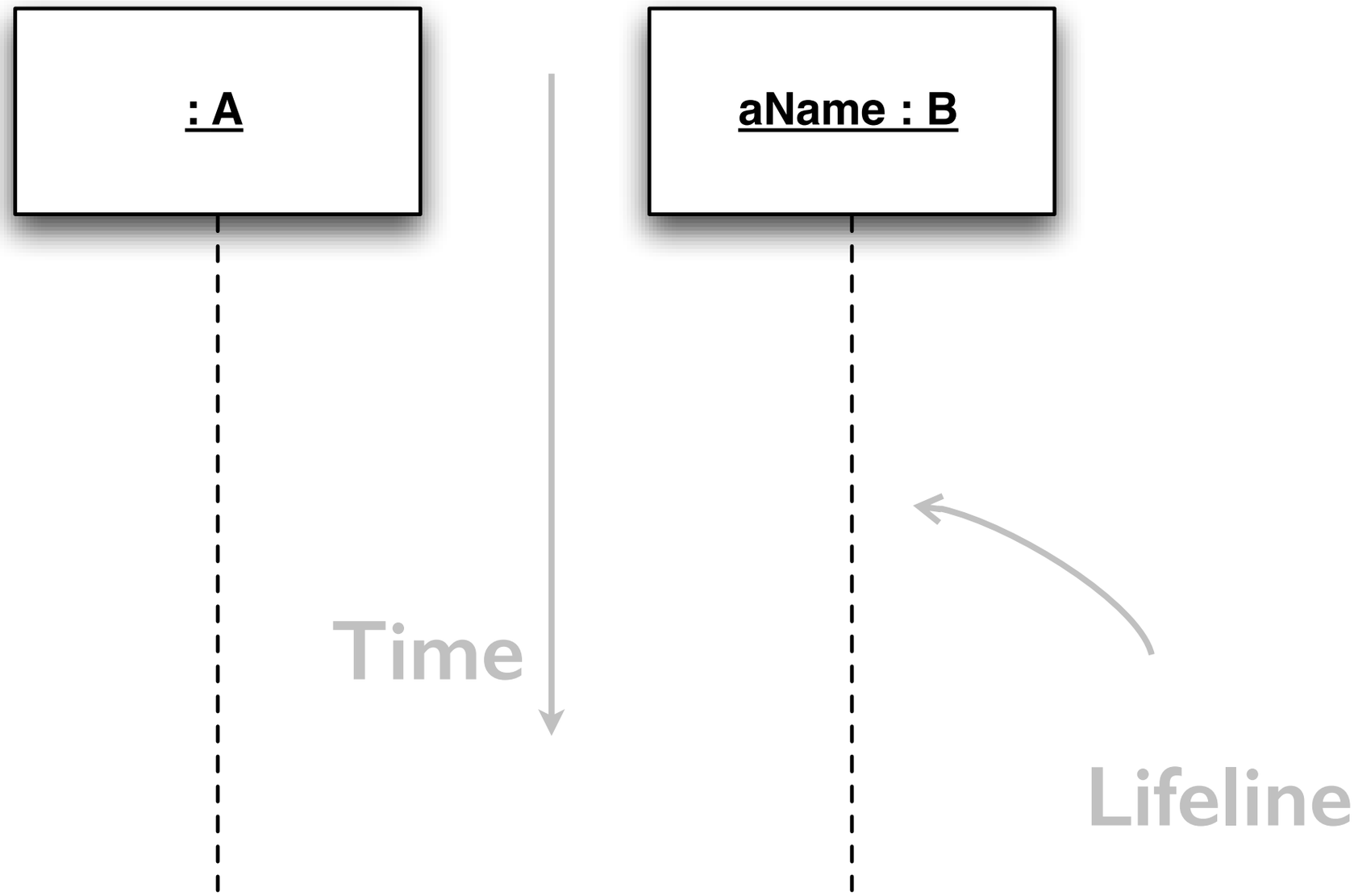
Behavioral & Interaction model



2

Sequence diagram

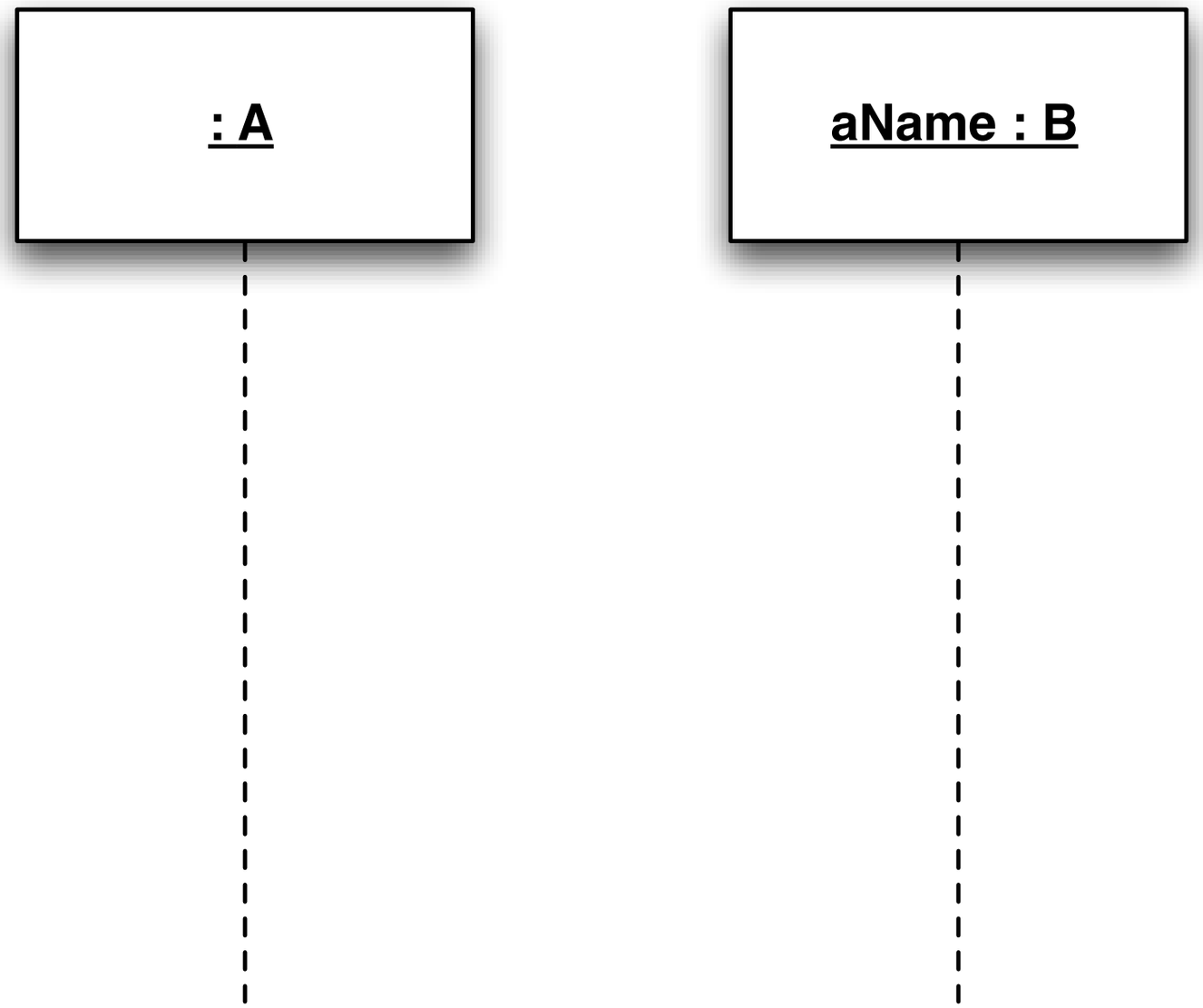
Behavioral & Interaction model



2

Sequence diagram

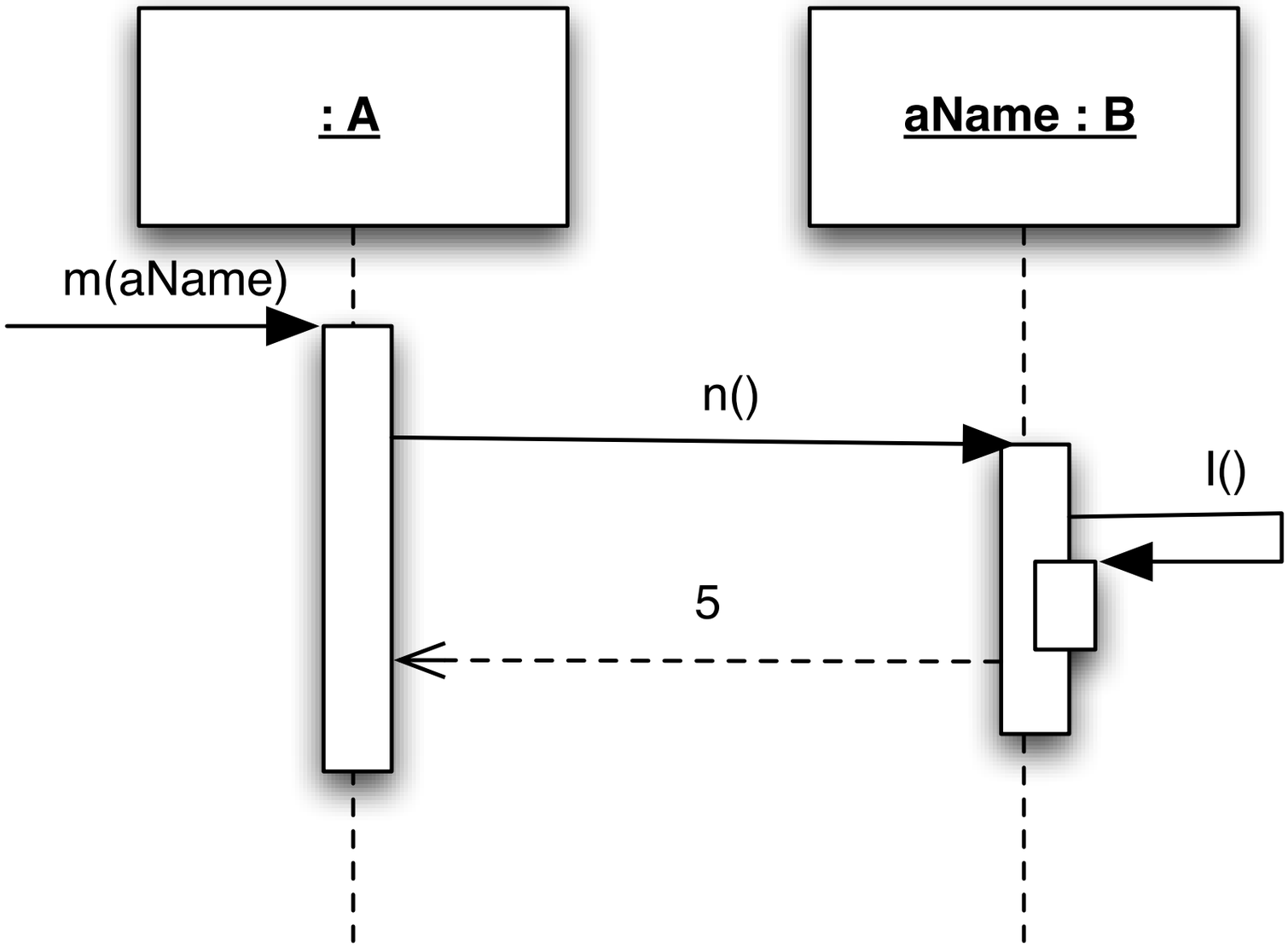
Behavioral & Interaction model



2

Sequence diagram

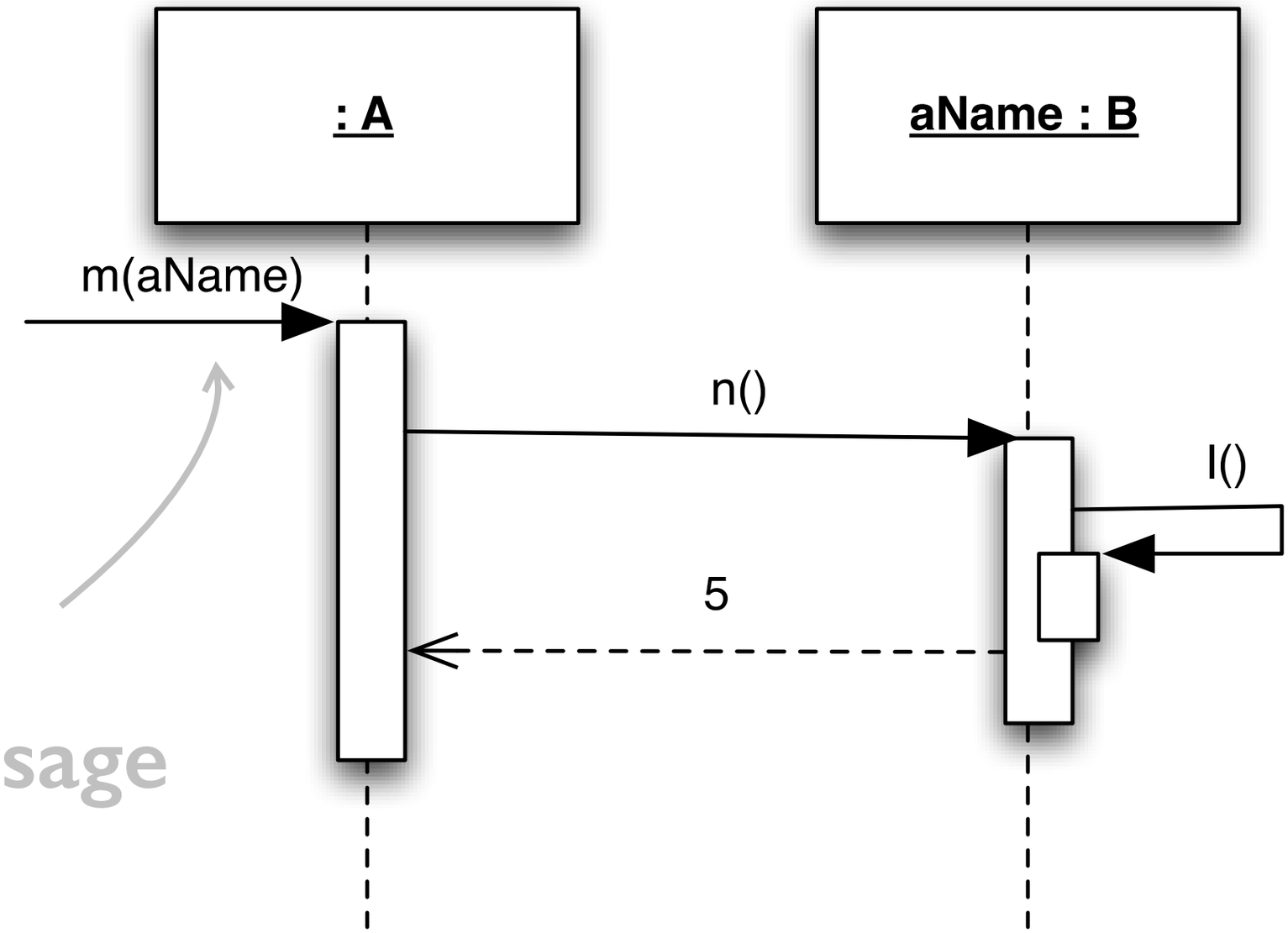
Behavioral & Interaction model



2

Sequence diagram

Behavioral & Interaction model

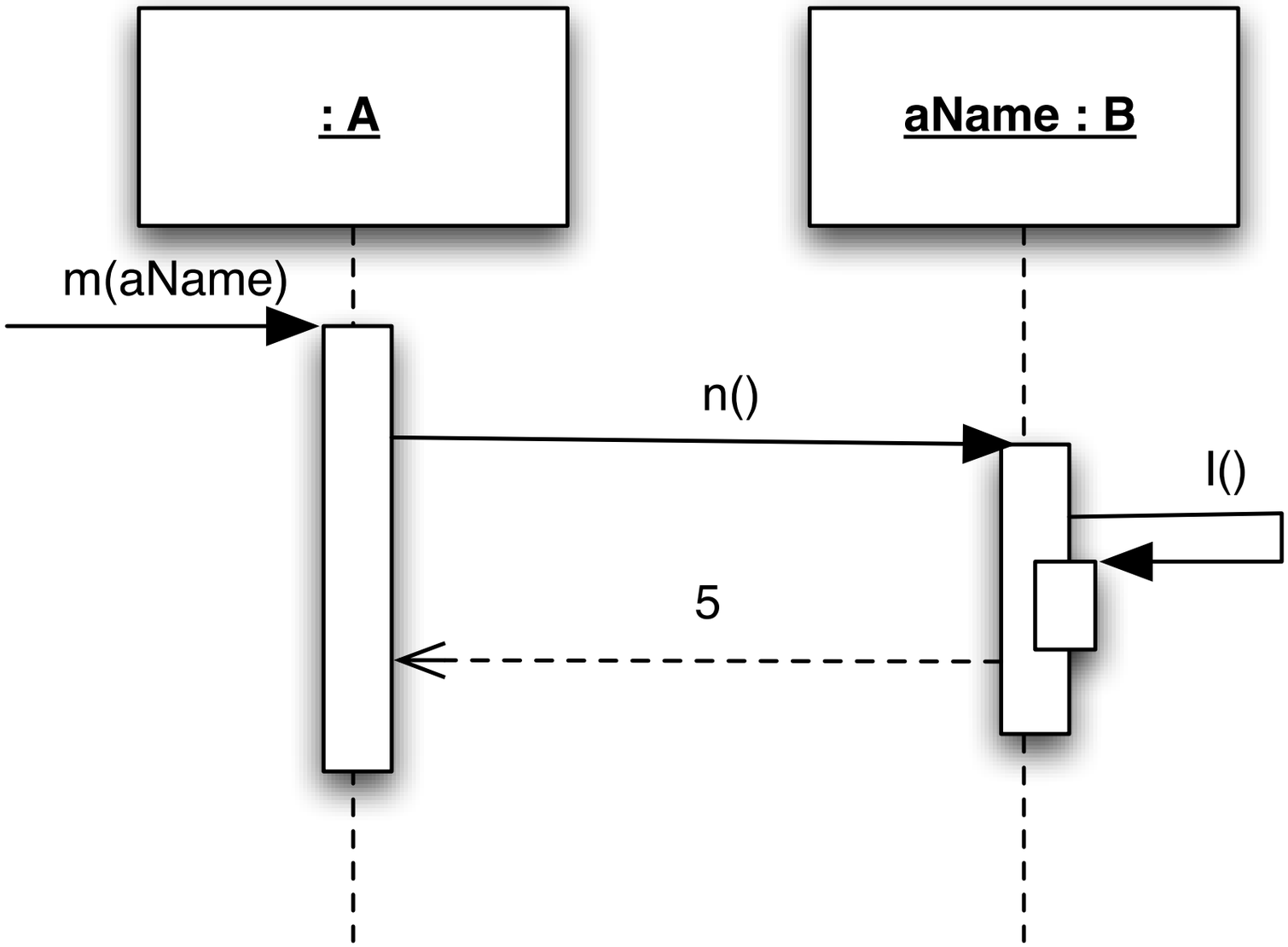


Message

2

Sequence diagram

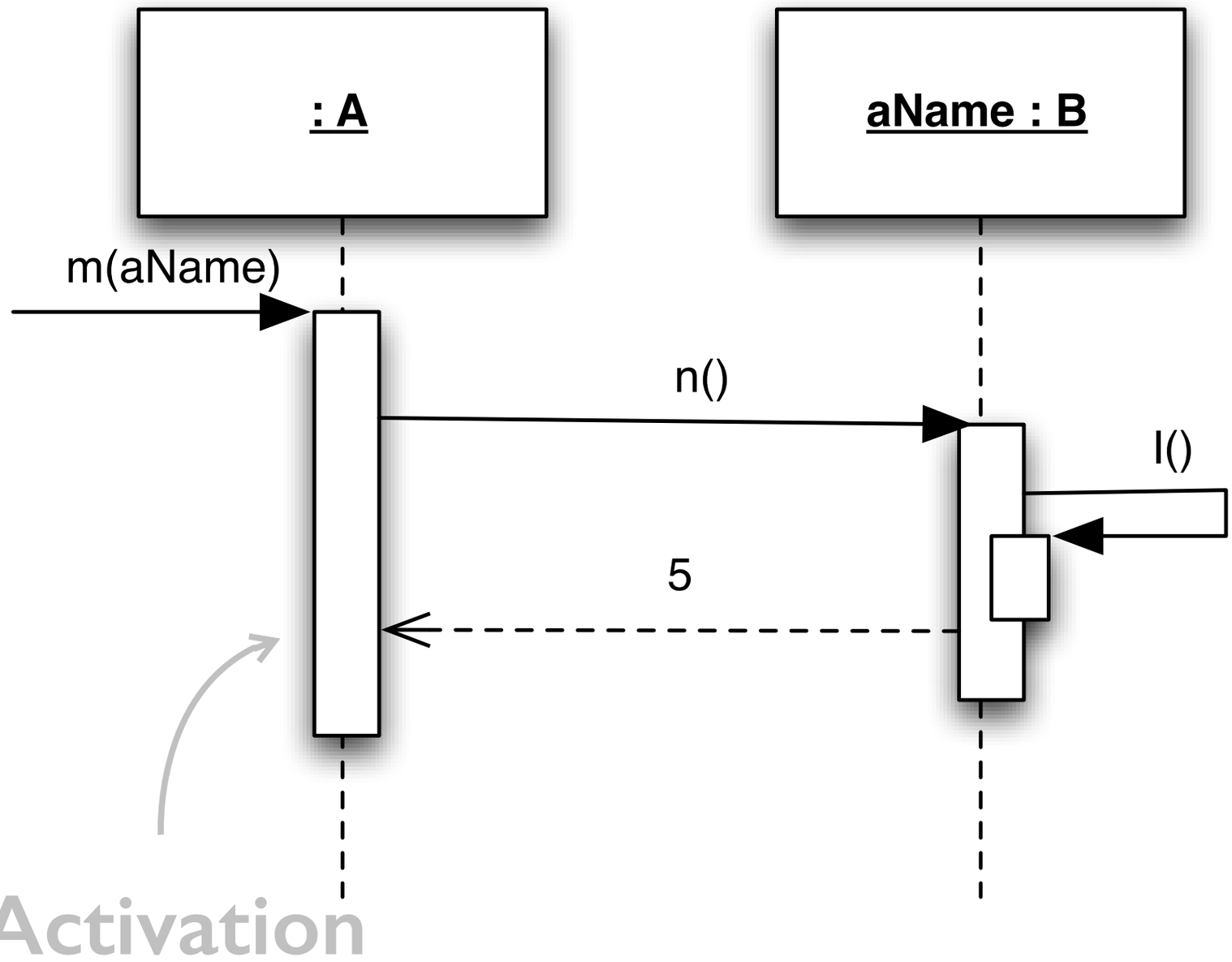
Behavioral & Interaction model



2

Sequence diagram

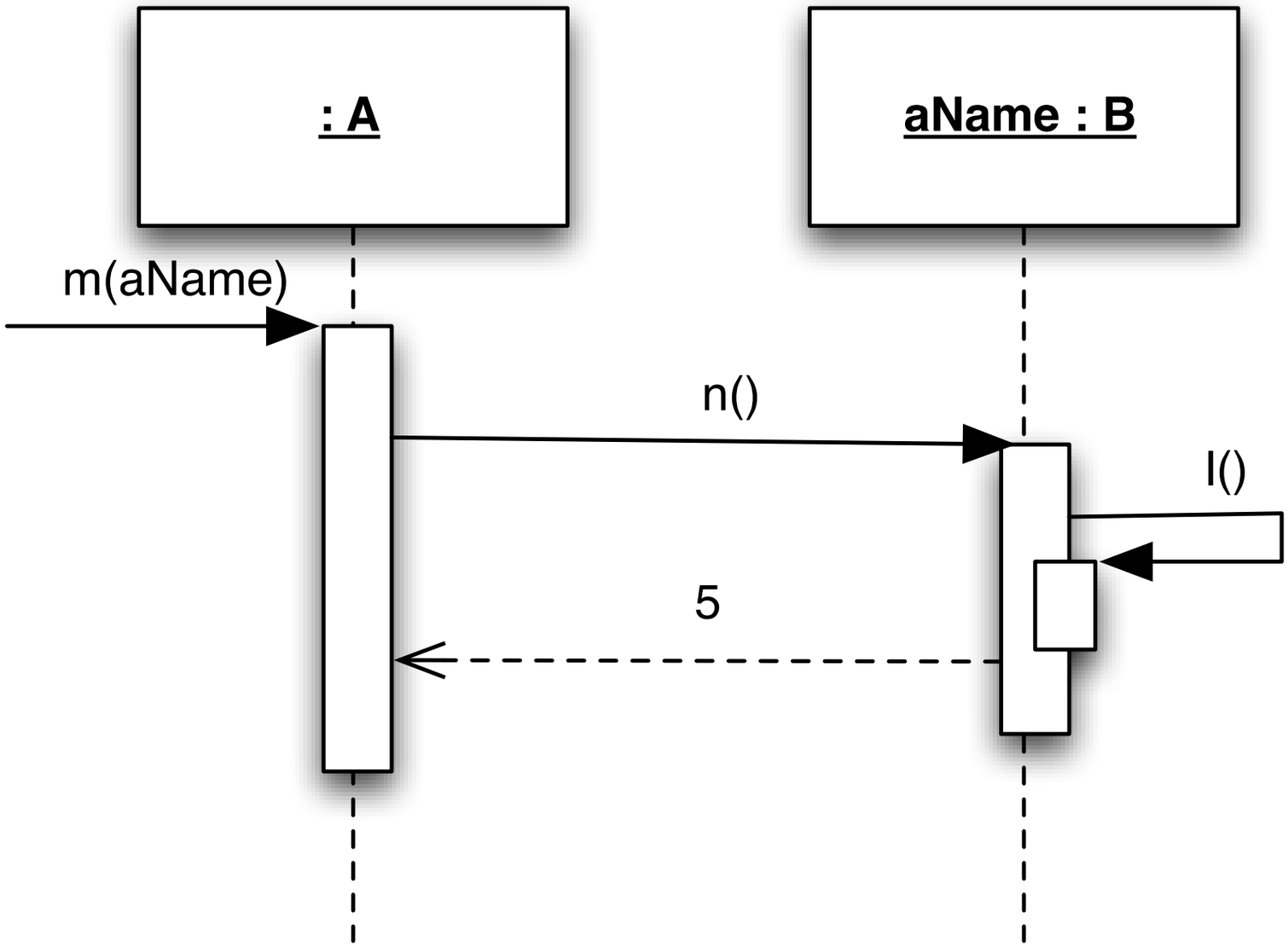
Behavioral & Interaction model



2

Sequence diagram

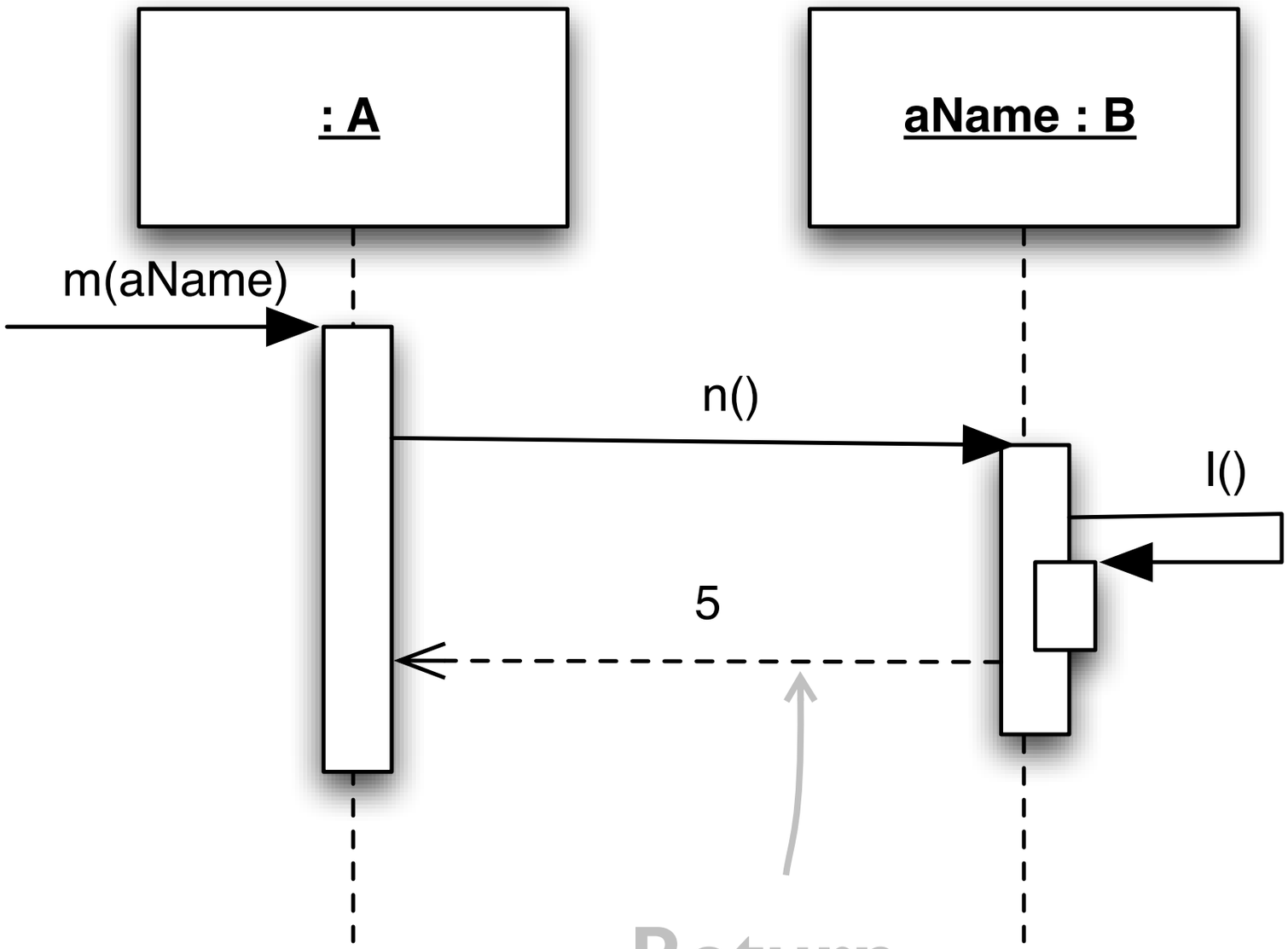
Behavioral & Interaction model



2

Sequence diagram

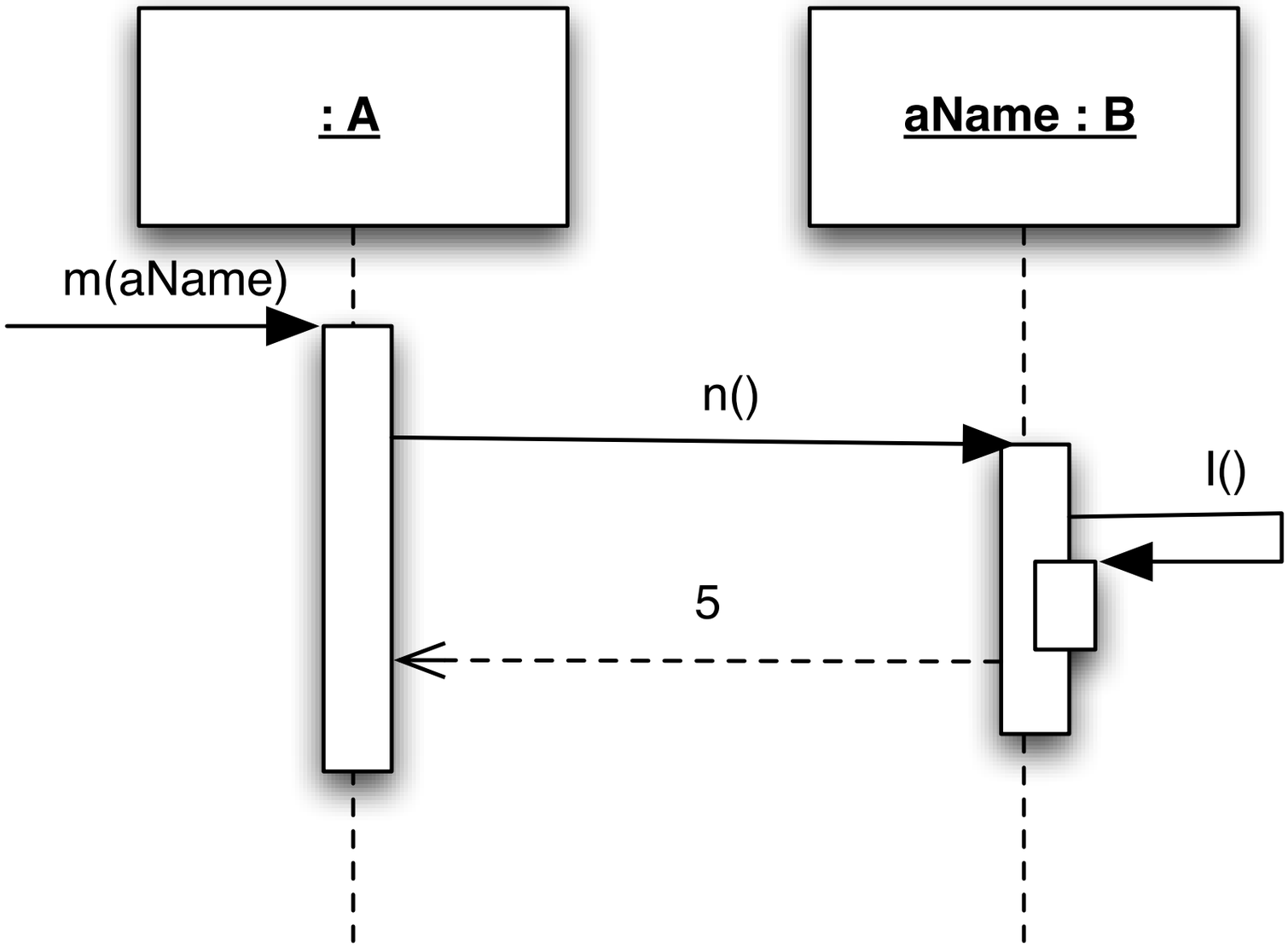
Behavioral & Interaction model



2

Sequence diagram

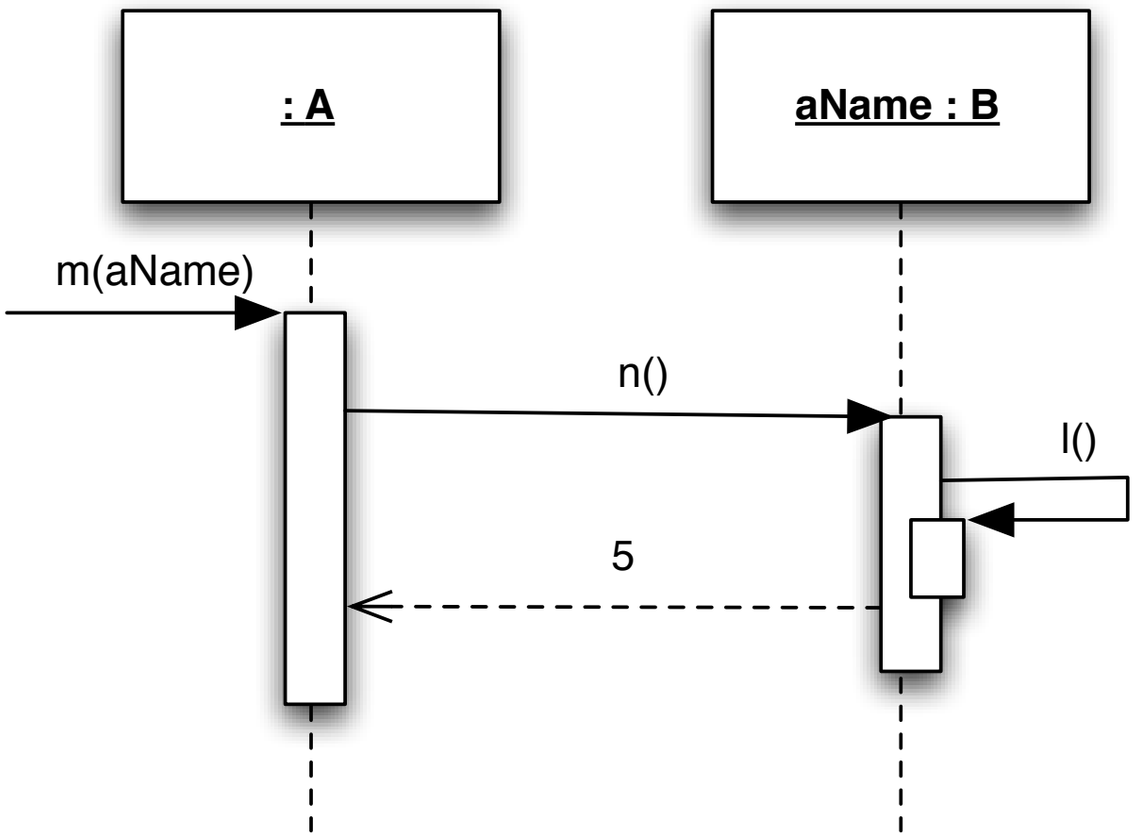
Behavioral & Interaction model



2

Sequence diagram

Behavioral & Interaction model



UML sketch

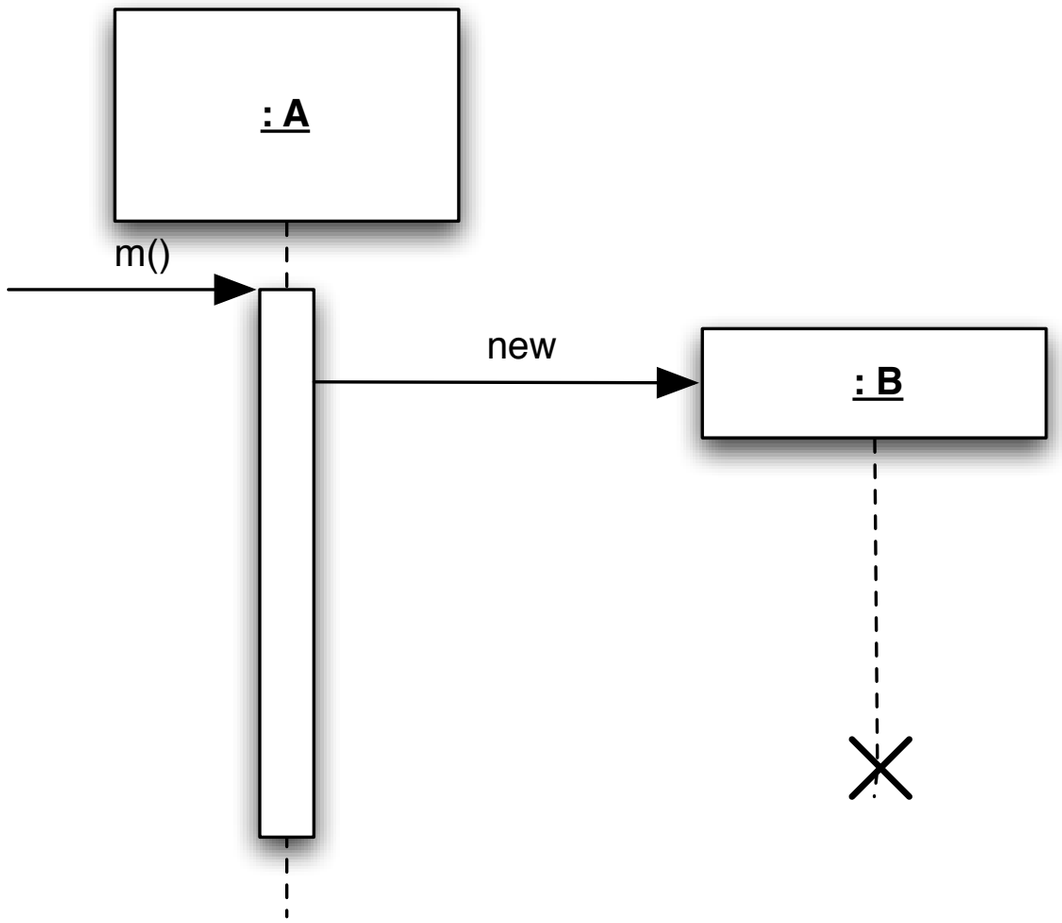
```
class A {
   public void m(B x) {
      x.n();
   }
}
class B {
   public int n() {
      this.l();
      return 5;
   }
   public void l() {}
```

Java sketch

2

Sequence diagram

Object creation & deletion



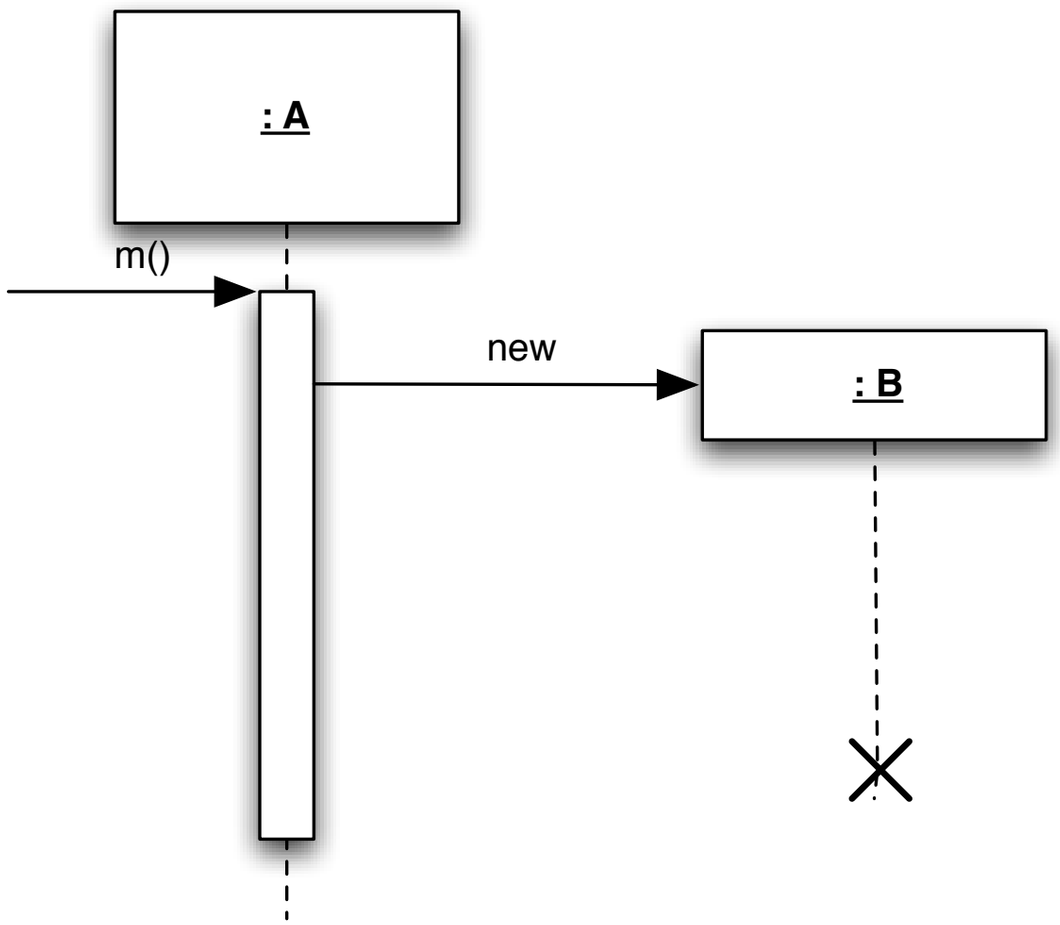
UML sketch

Java sketch

2

Sequence diagram

Object creation & deletion



UML sketch

```
class A {
    public void m() {
        ...
        new B();
        ...
        //the object is
        //no more accessible
    }
}
```

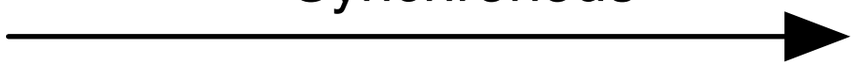
Java sketch

2

Sequence diagram

Other notations

Synchronous



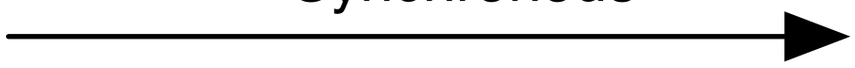
e.g. method invocation

2

Sequence diagram

Other notations

Synchronous



e.g. method invocation

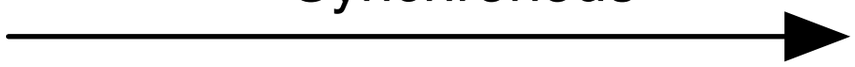
Async Message

e.g. start an execution thread

Sequence diagram

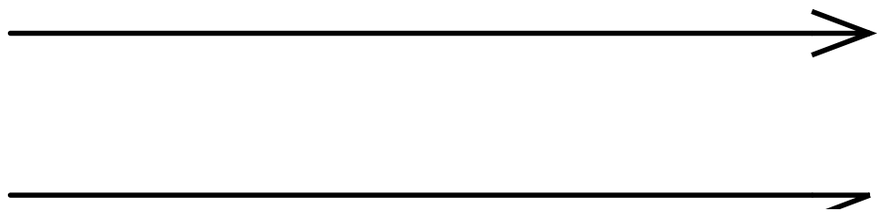
Other notations

Synchronous

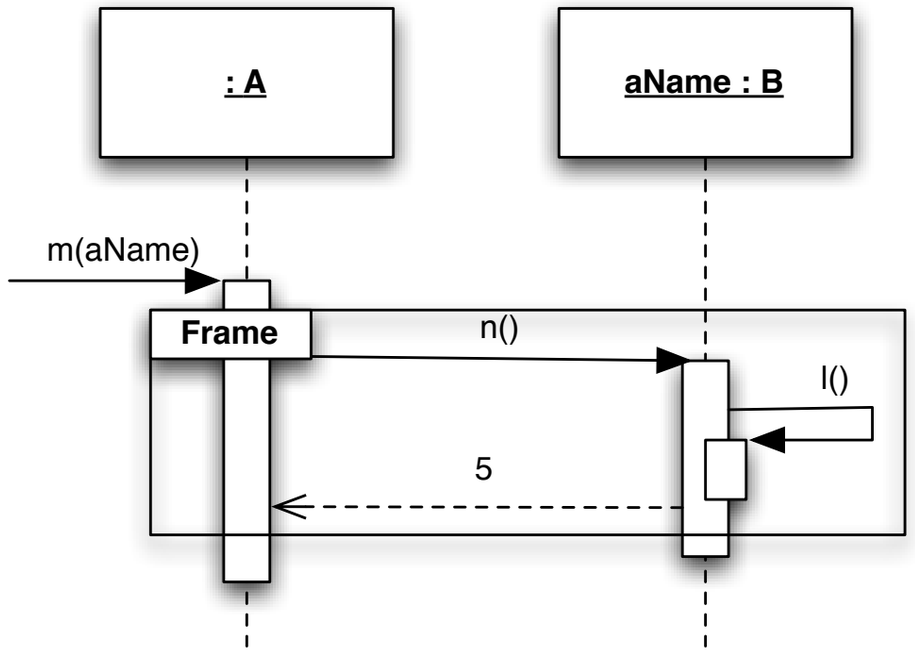


e.g. method invocation

Async Message



e.g. start an execution thread



e.g. loops, conditions

Sequence diagram

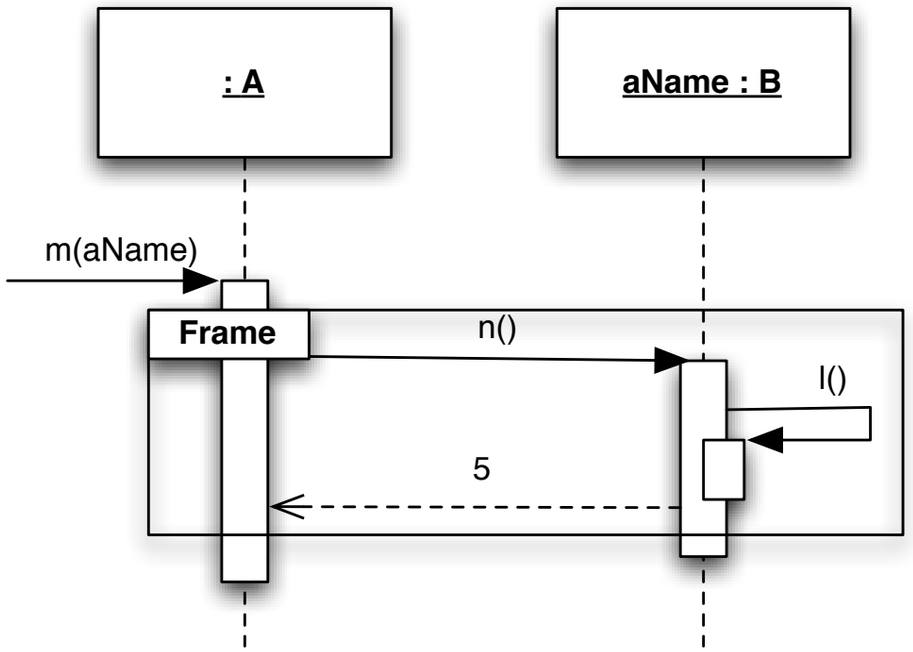
Other notations

Synchronous

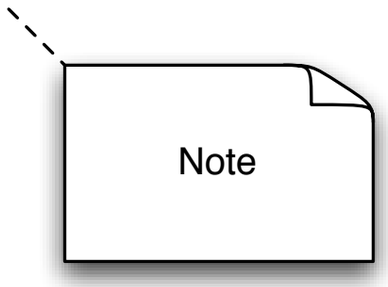
e.g. method invocation

Async Message

e.g. start an execution thread



e.g. loops, conditions



Comments

2

Sequence diagram

Let us see an example ...