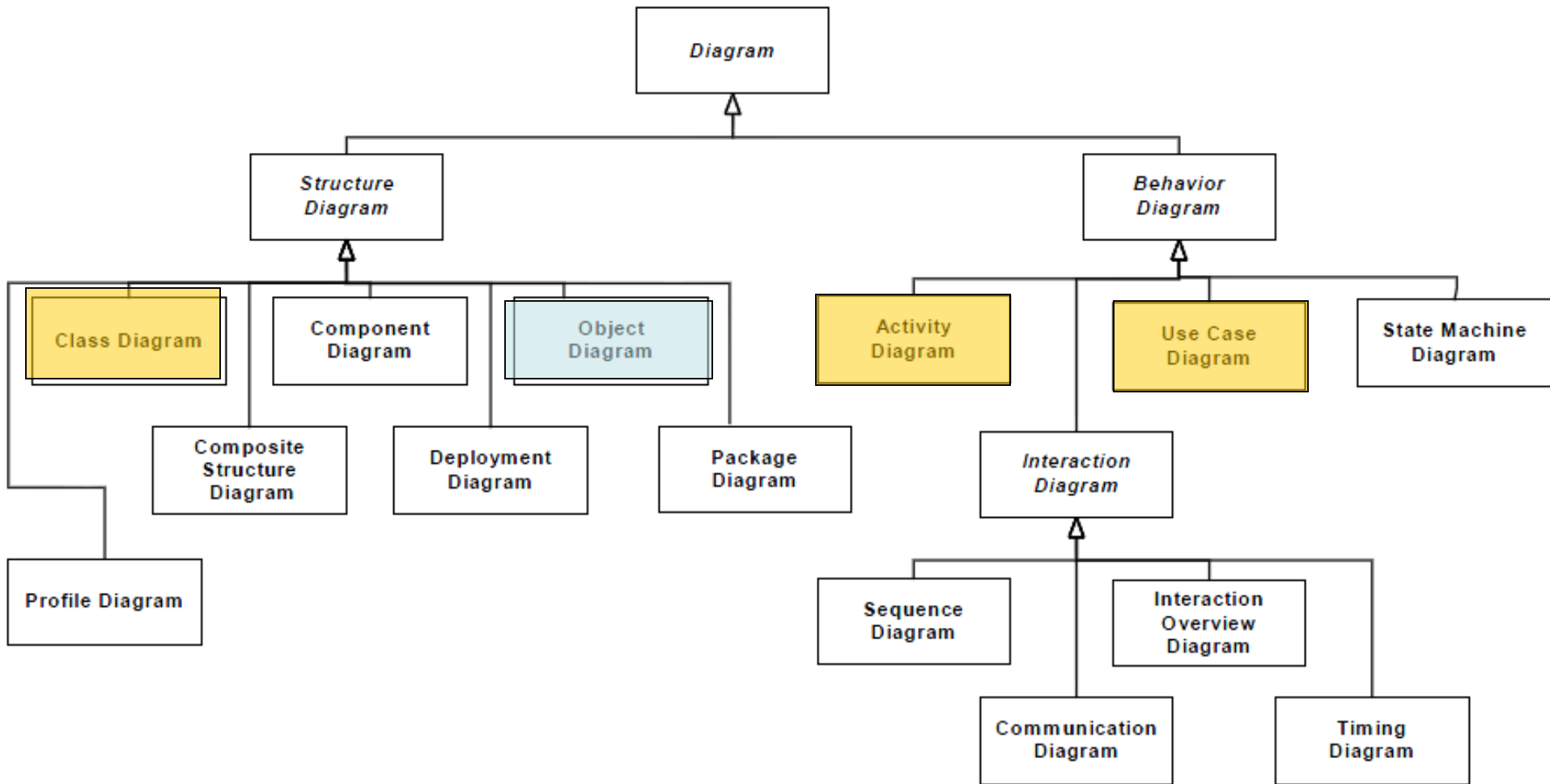


Object-Oriented Technology and UML Object Diagram

Object Diagram



Topics

- Basic concepts
- Constituent elements
- Representation
- Reading method
- Modeling

Instances and objects

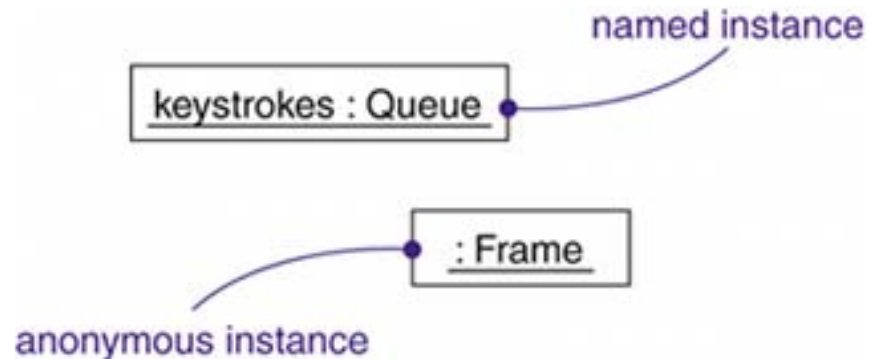
- The terms "instance" and "object" are largely synonymous, so, for the most part, they may be used interchangeably
 - An instance is a concrete manifestation of an abstraction to which a set of operations may be applied and which may have a state that stores the effects of the operation

Instances and objects

- You use instances to model concrete things that live in the real world
- An abstraction denotes the ideal essence of a thing
- An instance denotes a concrete manifestation

Instances and objects

- The UML provides a graphical representation for instances
 - This notation permits you to visualize named instances, as well as anonymous ones
 - Graphically, an instance specification is rendered by underlining its name



Object

- Instances don't stand alone, they are almost always tied to an abstraction
- Most instances you'll model with the UML will be instances of classes, and these things are called objects
- In a general sense, an object is something that takes up space in the real or conceptual world, and you can do things to it

Object

- An object is an entity with identity, state and behavior

Object

- State
- Operations
- Names

State

- An object also has state, which in this sense encompasses all the **properties** of the object plus the **current values** of each of these properties

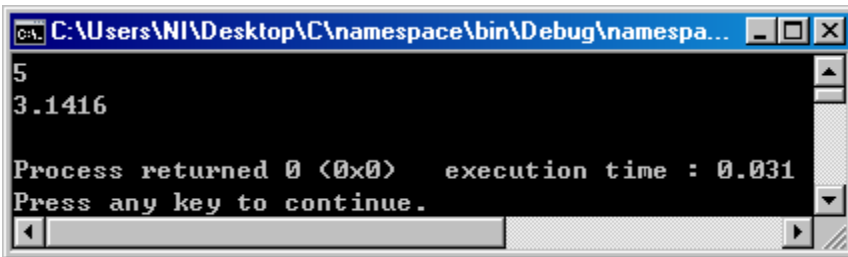
Operations

- Not only is an object something that usually takes up space in the real world, it is also something you can do things to
- The operations you can perform on an object are **declared in the object's abstraction**

Names

- An instance may have a name that distinguishes it from other instances within its context
- Namespace
 - C++ namespace
 - Java package

```
#include <iostream>
using namespace std;
namespace first
{
    int var = 5;
}
namespace second
{
    double var = 3.1416;
}
int main ()
{
    cout <<first::var<<endl;
    cout <<second::var<<endl;
    return 0;}
```



```
C:\Users\NI\Desktop\C\namespace\bin\Debug\namespa...
5
3.1416
Process returned 0 (0x0) execution time : 0.031
Press any key to continue.
```

Example of Object

- Person: A class
- Individual: an object (each has its own individual characteristics)
 - State: height, weight, education, job, income
 - Operations: read, exercise, eat
 - Name

Example of Object

主人公情報

趙雲 子龍
チウウン



主人公経験

農業	商業	技術	改修	治安	酒盛	仙術
529	520	746	410	983	205	60

歩兵	騎兵	弓兵	水軍	計略	謀報
374	891	760	68	747	710

戦歴	戦闘	一騎	舌戦
回数	56	80	65
勝利回数	53	80	65
連勝回数	5	80	65

特技

農業	商業	技術	改修	治安	微兵	訓練
突撃	火矢	一斉	鎮静	鼓舞	奇襲	隠密
同討	足止	誘引	混乱	天文	地理	石陣
回復	気合	気炎	受返	弾返	三段	螺旋
威圧	抗弁	反論	揚足	論破	挑発	面罵
軍師	名士	提督	間諜	酒豪	医者	仙人

説明

【農業】で実行期間の延長可
【条件】 農業経験200

能力 経験

統率	98(+ 5)	63
武力	99(+ 2)	5
知力	87(+10)	85
政治	74(+ 7)	67
魅力	83	32
傷病	健康	

中止 アイテム 状態 経験



Example of Object

主人公情報

趙雲 子龍
チヨウウン



主人公経験

農業	商業	技術	改修	治安	酒盛	仙術
529	520	746	410	983	205	60

歩兵	騎兵	弓兵	水軍	計略	謀報
374	891	760	68	747	710

戦歴	戦闘	一騎	舌戦
回数	56	80	65
勝利回数	53	80	65
連勝回数	5	80	65

年齢 47 性別 男 特技

能力 経験

統率	98(+ 5)	63
武力	99(+ 2)	5
知力	87(+10)	85
政治	74(+ 7)	67
魅力	83	32
傷病	健康	

特技

農業	商業	技術	改修	治安	徴兵	訓練
突撃	火矢	一斉	鎮静	鼓舞	奇襲	隠密
同討	足止	誘引	混乱	天文	地理	石陣
回復	気合	気炎	受返	弾返	三段	螺旋
威圧	抗弁	反論	揚足	論破	挑発	面罵
軍師	名士	提督	間諜	酒豪	医者	仙人

説明

【農業】で実行期間の延長可
【条件】 農業経験200

中止 アイテム 状態 経験

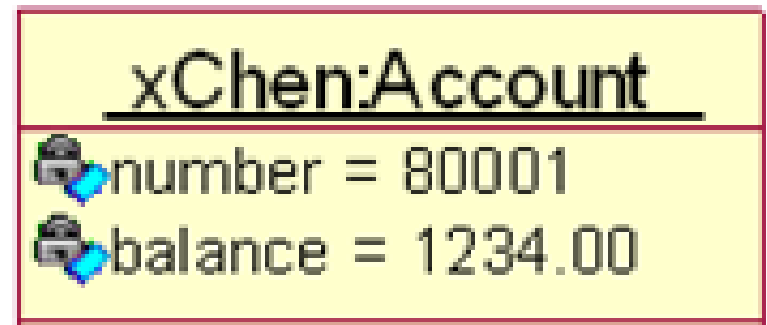
State

Name

Operations

Representation of Objects

- A name is a textual string
 - You can also **simply name an object** (such as myCustomer) and elide its abstraction if it's obvious in the given context. In many cases, however, the real name of an object is known only to the computer on which that object lives. In such cases, you can render **an anonymous object** (such as :AudioStream).
- You can use the UML to **render the value of an object's attributes**



Relationship between Objects

- A link is a semantic connection among objects
- In general, a link is an instance of an association
- Wherever a class has an **association** to another class, there may be a **link** between the instances of the two classes
- Wherever there is a **link** between two objects, one object can send a **message** to the other object.

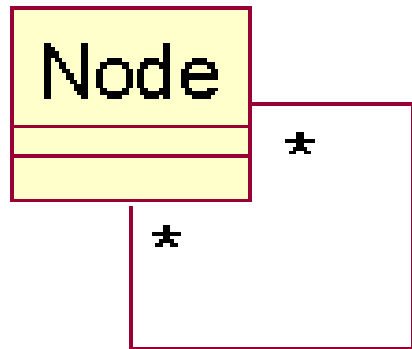
Object Diagram

- An object diagram is a diagram that shows a set of **objects** and their **relationships** at a point in time
 - Graphically, an object diagram is a collection of vertices and arcs
- Object diagrams commonly contain
 - Objects
 - Links

Note of Object Diagram

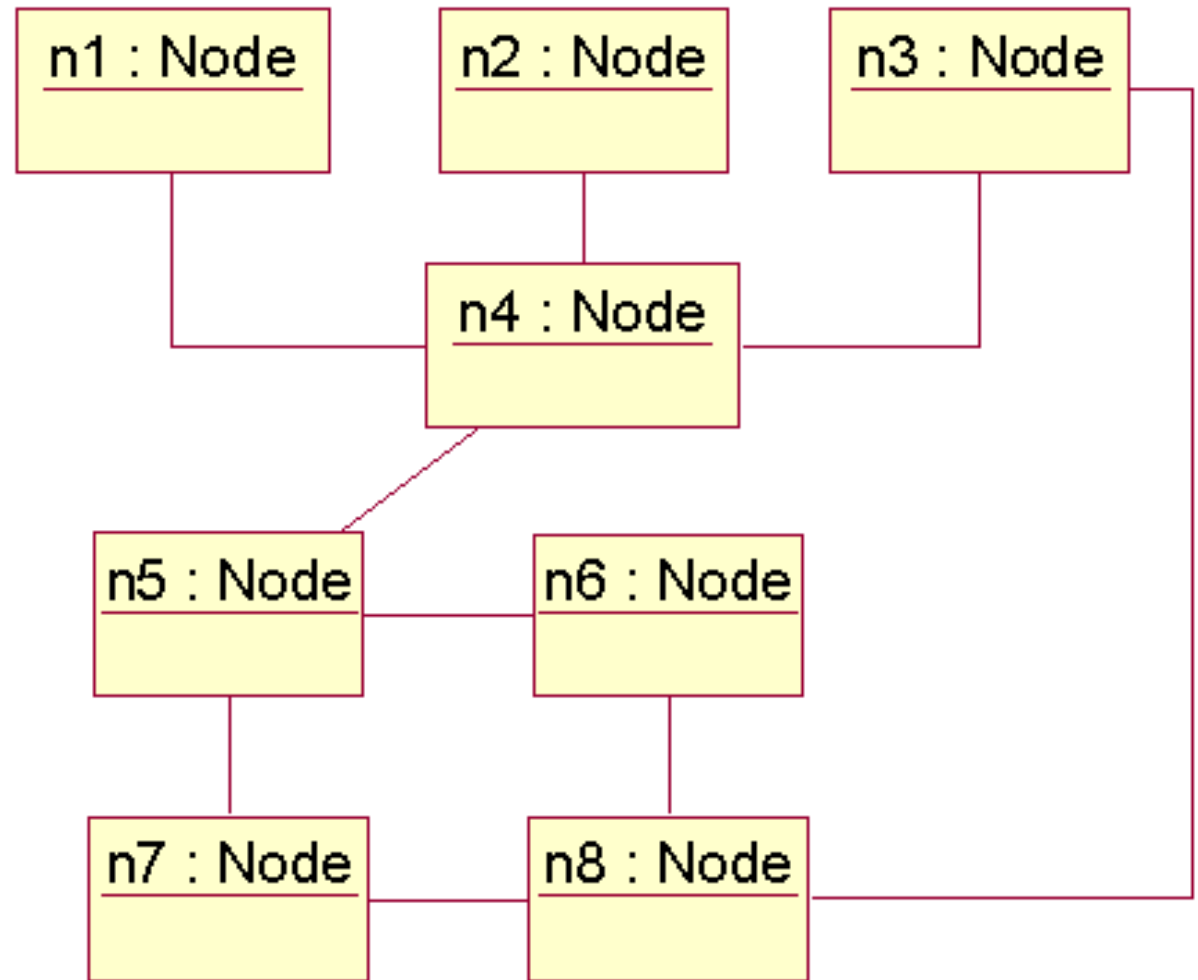
- An object diagram correlates with a class diagram
 - The class diagram describes the general situation, and the instance diagram describes specific instances derived from the class diagram
- You use object diagrams to model the **static design view** or **static process view** of a system just as you do with class diagrams, but from the perspective of real or prototypical instances

Example of Object Diagram

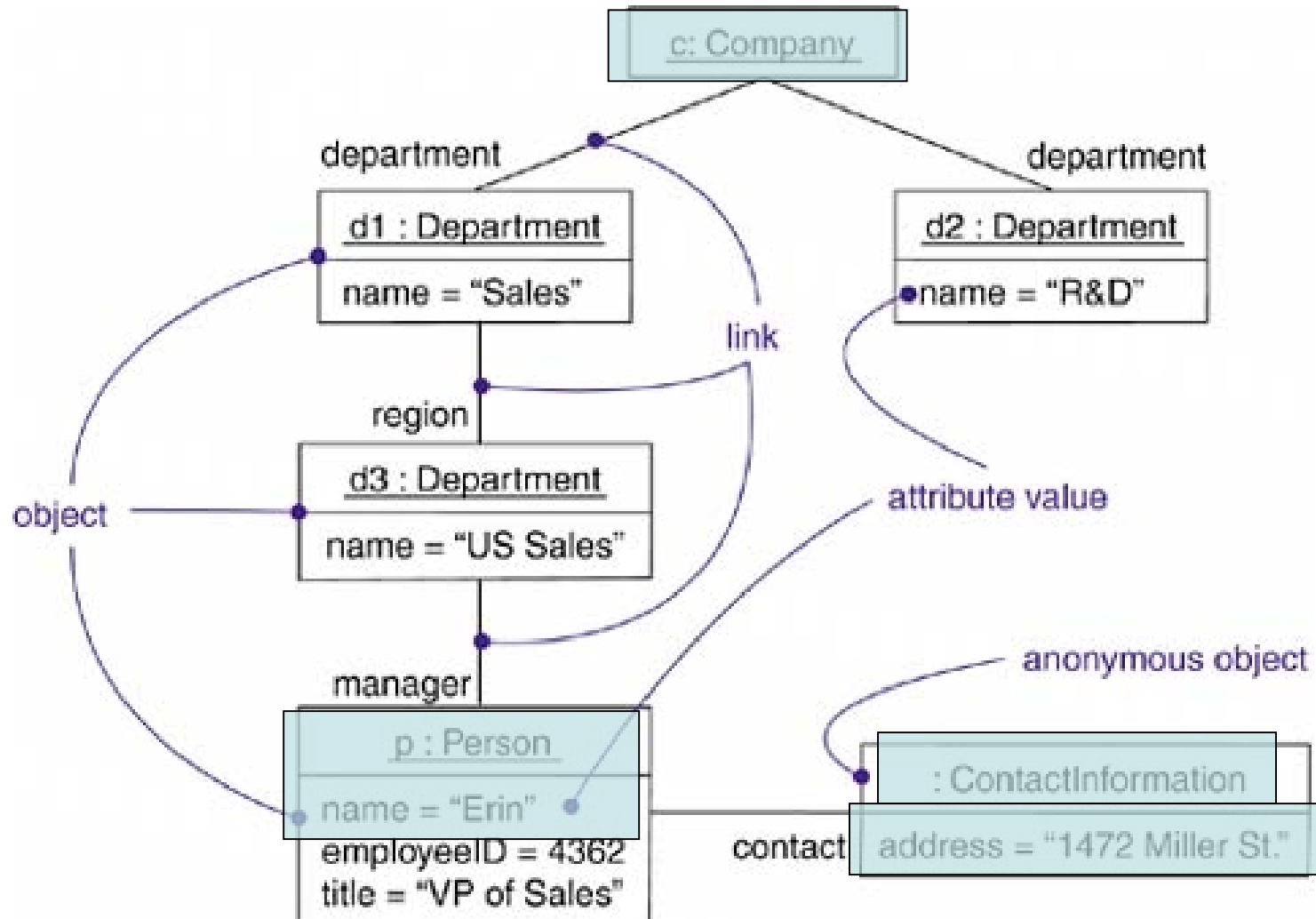


Connects

Class Diagram



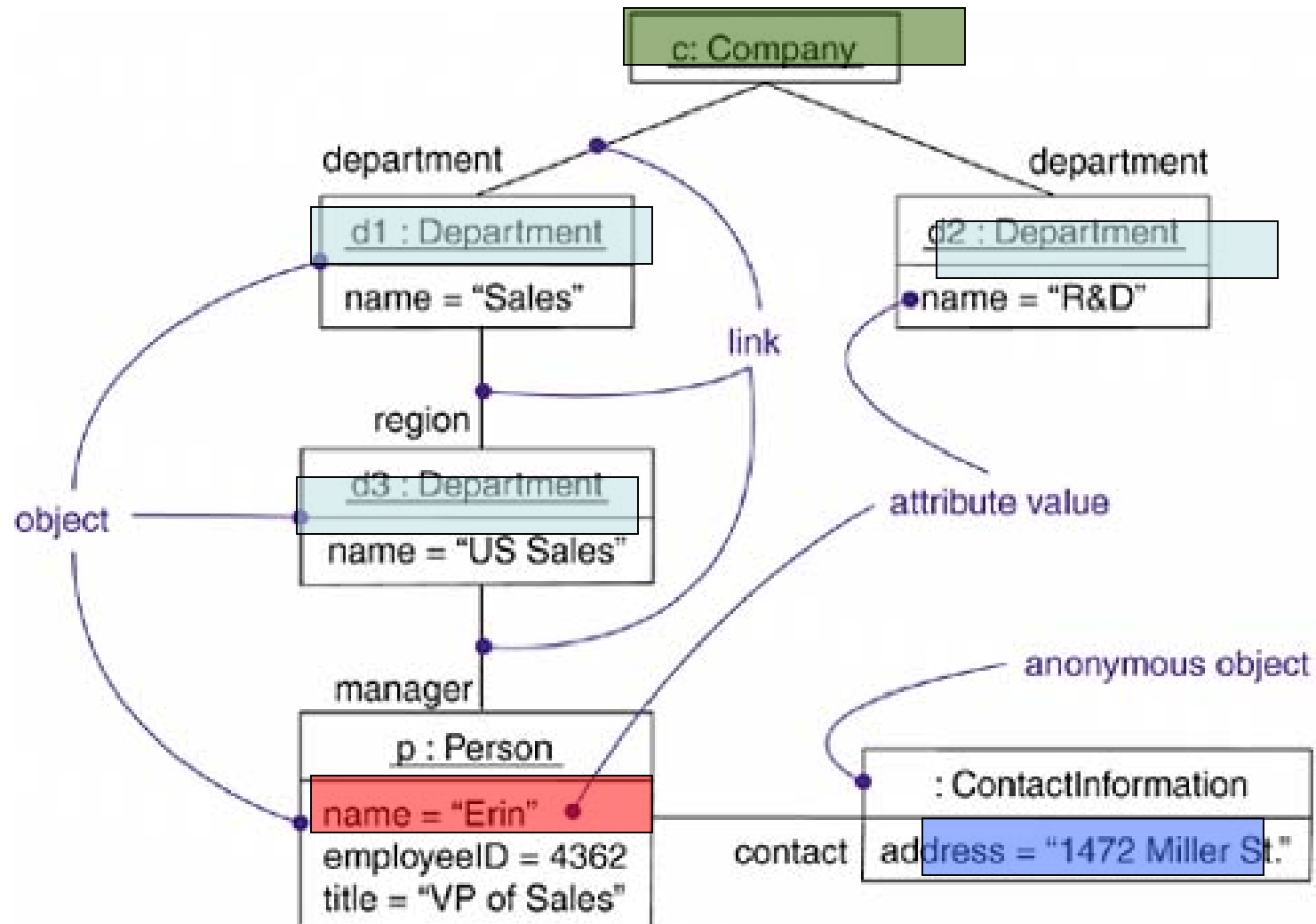
Example of Object Diagram



How To Read an Object Diagram

- Identify all of the classes in the object diagram
- Understand the semantics of each object
- Understand the connections between the objects

Read an Object Diagram



Uses of Object Diagram

● Modeling Object Structures

- Object diagrams are especially useful for modeling complex data structures
- Verification of class model
- Analysis and description of the source code

Style of UML Object Diagram

- Indicate Attribute Values to Clarify Instances
- Prefer Object Names over Attribute Values
- Indicate Roles to Distinguish Different Relationships

Style of UML Object Diagram

