Quality of Design (cont.)

Comp-304 : Quality of Design (cont.)
Lecture 17

Alexandre Denault
Original notes by Hans Vangheluwe
Computer Science
McGill University
Fall 2007

Tutorial

Two tutorials will be held today.

16h30-17h30
18h30-19h30

McConell 103

CSGames

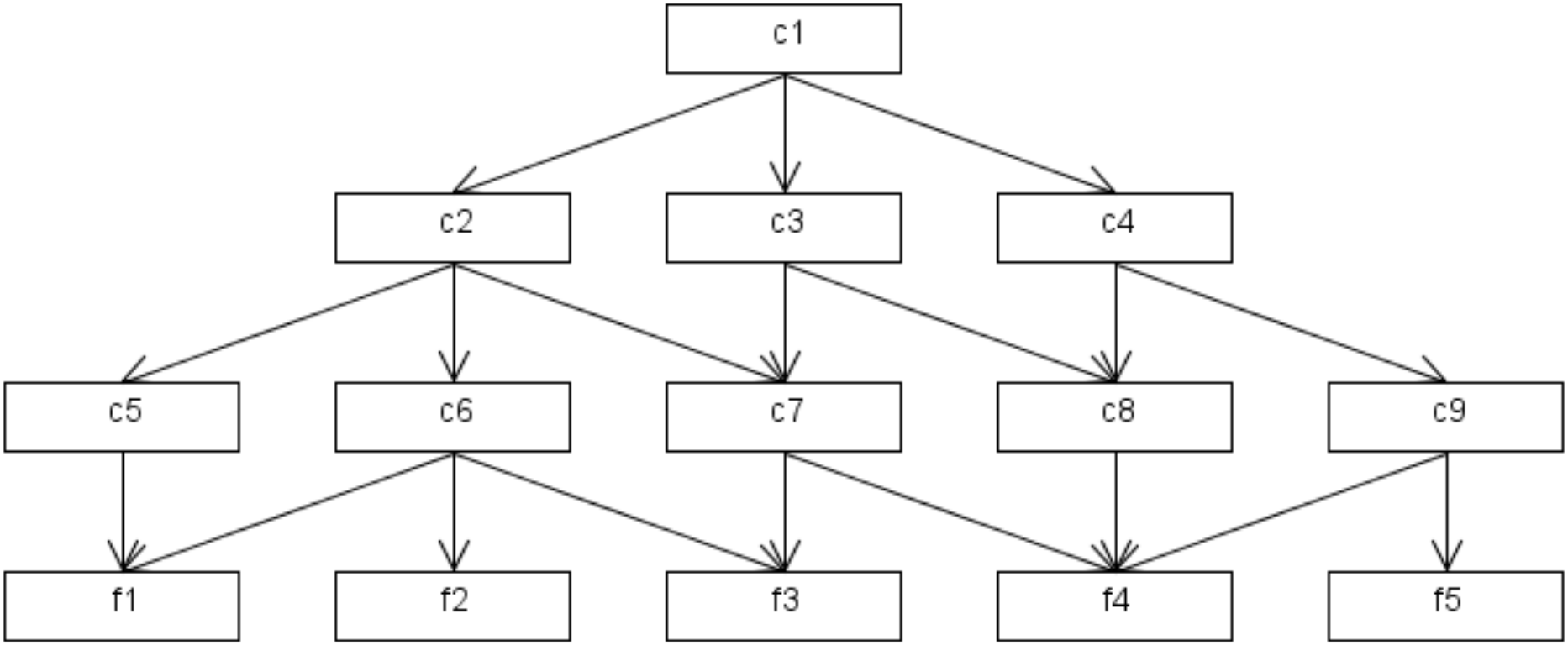
The 2007 CS Games have returned home to McGill University and we are looking for volunteers. (www.csgames.org)

The CS Games are an annual competition between North American universities with computer science oriented events such as debugging and scripting, including other competitions like LAN gaming and a scavenger hunt.

We will need help putting together all of the events and converting Trottier into our competition hall for the weekend. A background in computer science is not necessary. The games will be held on the weekend of March 9-11. If interested, please contact the volunteer coordinator Chris at volunteer@csgames.org. Please include your availability and major.

If you've got friends who are not in computer science, or even at McGill, they are more than welcome to help out as well!

Not-So-Simple Example



Low/High Encumbrance

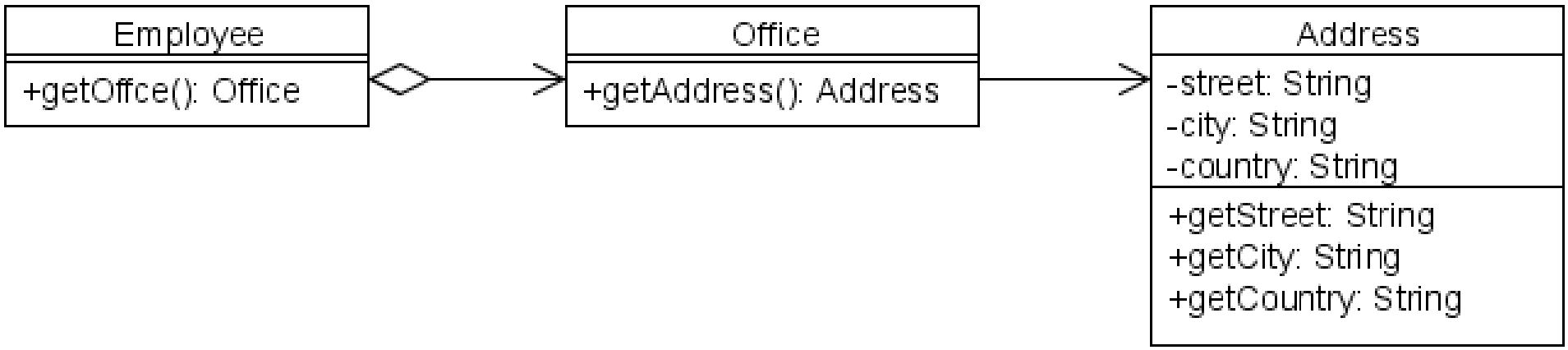
- A foundation class should have low encumbrance.
- An application class should have high encumbrance.
- A good indication of a problem in the design is:
  - High indirect encumbrance in the Foundation domain.
  - Low indirect encumbrance in an Application domain.

Law of Demeter

- The Law of Demeter limits the size of direct class reference sets.
- It states that if an object o1 refers to a object o2 through some method m of o1, then o2 must be:
  - the object itself (so o2 is actually o1)
  - an object referred to by the arguments of m
  - an object referred to by a variable of of o1
  - an object created by m
  - an object referred to by a global variable
- In summary, an object should only send messages to objects it can directly reference.

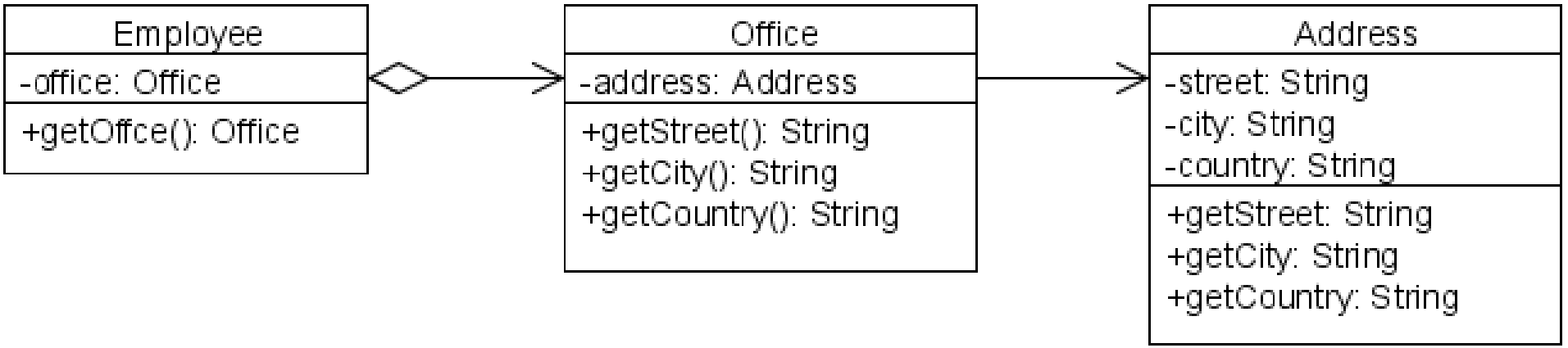
  Only talk to your immediate friends!

Example




```
String employeeStreet = this.office.getAddress().getStreet();
```

Example Fixed



String employeeStreet = this.office.getStreet();

Cohesion

- Measure of interrelatedness of features (attributes and methods) in an external interface of a class.

- Low (bad) cohesion
  - set of features that don't belong together
- High (good) cohesion
  - set of features that all contribute to the implementation

Three types of Cohesion

- Mixed-Instance Cohesion
  - Really really bad!
- Mixed-Domain Cohesion
  - Really bad!
- Mixed-Role Cohesion
  - Bad!

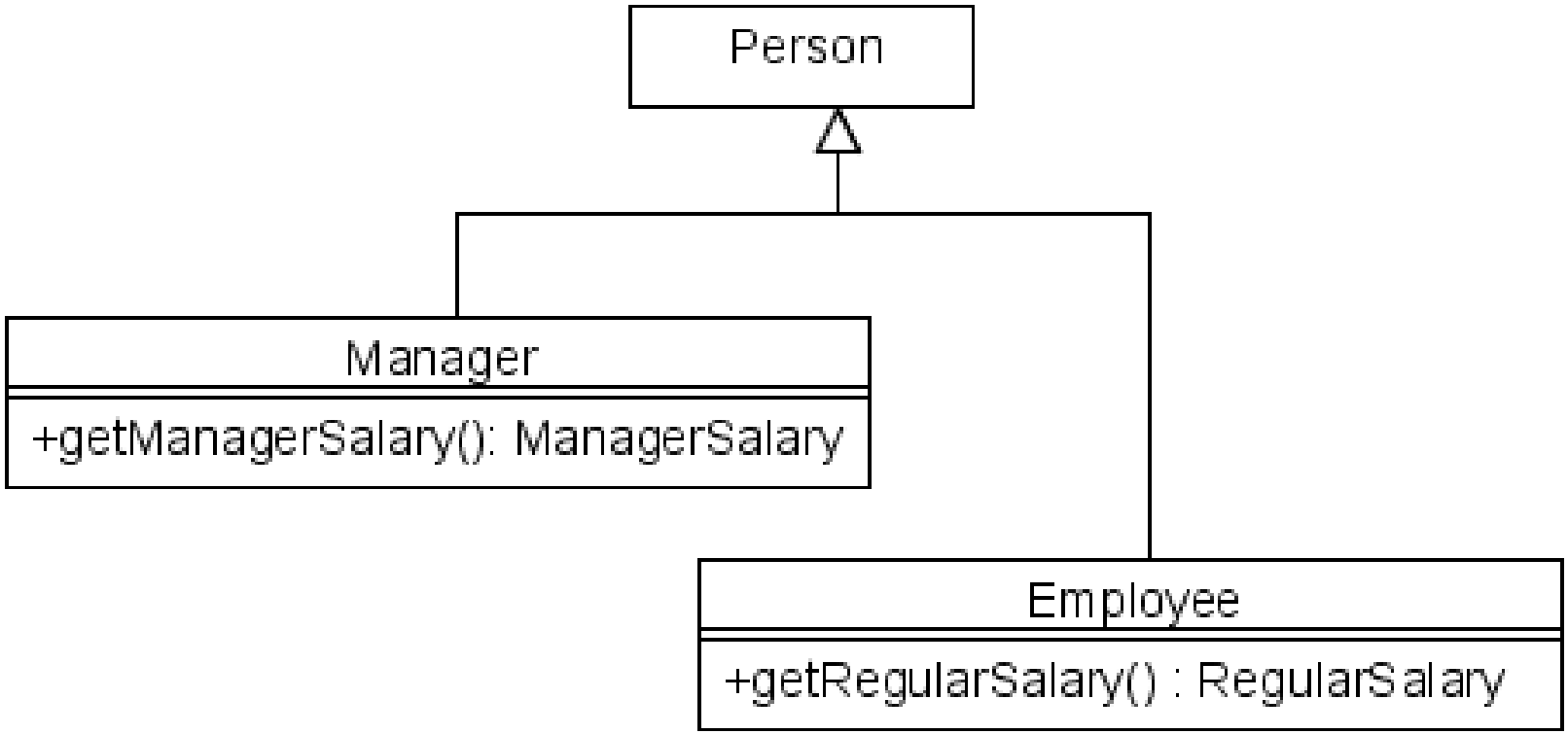
Mixed-Instance Cohesion

A class with mixed-instance cohesion has some features that are undefined for some objects of the class.

- Suppose you work at a company that has managers and non-managers.
- Managers receive a ManagerSalary and other employees receive a RegularSalary.
- Imagine employees are implemented using a Person class.
  - That class has a getManagerSalary() and a getRegularSalary() method which returns both types of salary.
- For each Person instance, we have features that won't be used.
  - Thus, Person is too broad

How to solve this?

- Usually means that there is a class hierarchy missing.
  - in our case, we should have classes Manager and Employee that inherit from a superclass Person.
- Now we won't have any unused features.



Extrinsic vs Intrinsic

- The class B is extrinsic to A if A can be fully defined with no notion of B.
  - For example, Dog is extrinsic to Person, because in no sense does “Dog” capture some characteristic of Person.
- B is intrinsic to A if B captures some characteristic inherent to A.
  - For example, Dog is intrinsic to DogOwner, because “Dog” captures some characteristic of DogOwner.

Mixed-Domain Cohesion

A class with mixed-domain cohesion contains an element that directly encumbers the class in an extrinsic class of a different domain.

- In other words, a class should only encumber classes in other domains if they are intrinsic.
- For example, Invoice and Currency are two classes that exist in different domains.
- Since Currency is intrinsic to Invoice, there is no mixed-domain cohesion.
- However, Invoice and Printer, which also exist in different domains, would present mixed-domain cohesion since Printer is extrinsic to Invoice.

Mixed-Role Cohesion

A class C with mixed-role cohesion contains an element that directly encumbers the class with an extrinsic class that lies in the same domain as C.

- In other words, a class should only encumber classes if they are intrinsic.
- Lets go back to our example with Dog and Person.
- Although both classes exist in the same domain, they are not intrinsic
- As such, Dog should not encumber Person.
- Although this is the less serious cohesion problem, you must take it into account when designing for reusability.