CSSE 220

Object-Oriented Design
Files & Exceptions

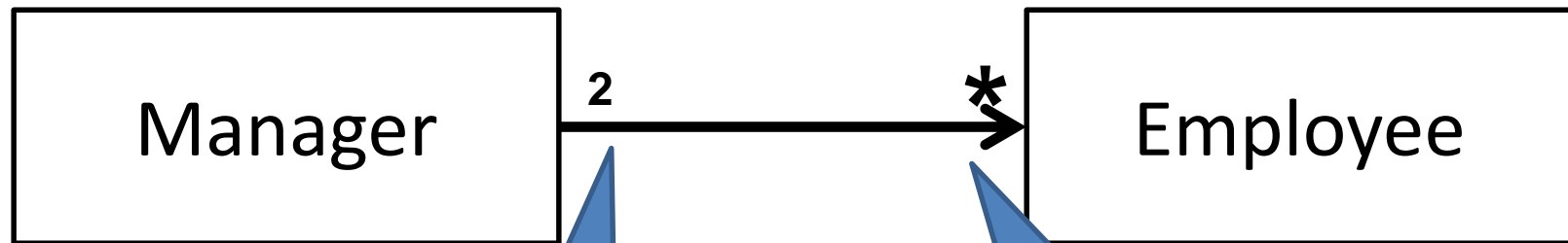# Announcements

- Take Moodle survey today to voice your preferences for project partners.

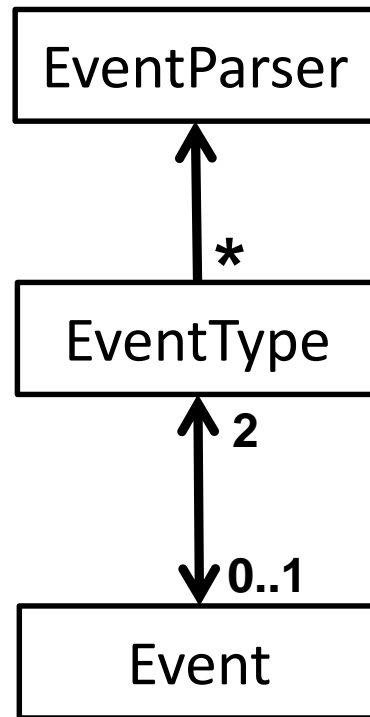# New UML Notation: Cardinality
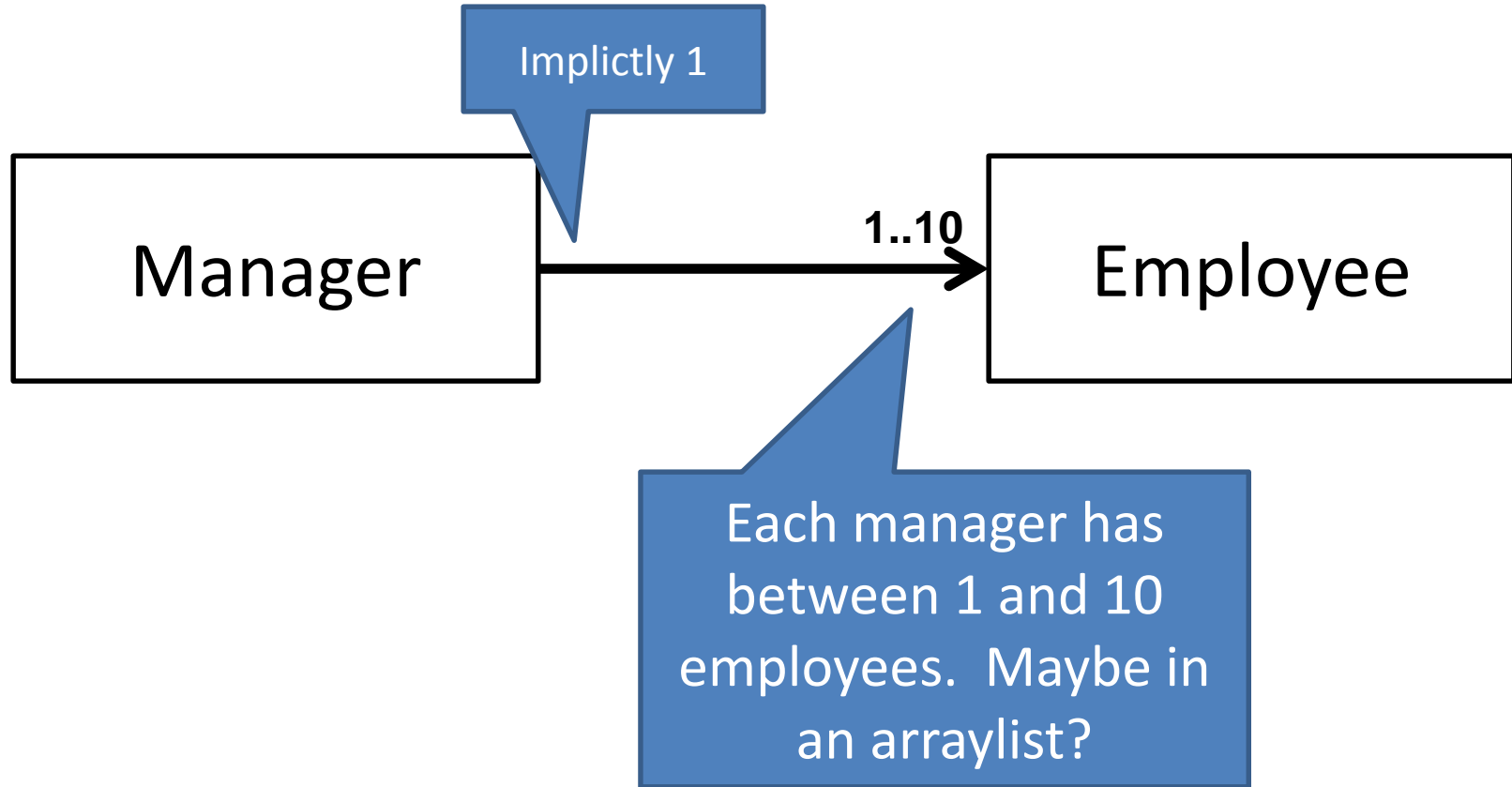
More Cardinality

Manager —2———*→ Employee

Every employee has exactly 2 managers. Note that this can be used even if there is no reference from Employee to Manager
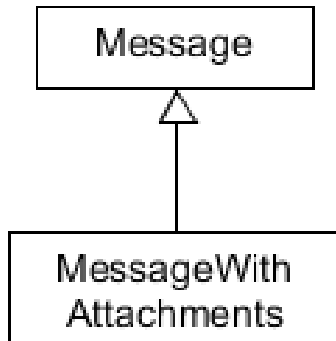
Managers have any number of employees.

The * means "zero to infinity" – any arbitrary number. You can also occasionally see something like 4..* to mean 4 or more.
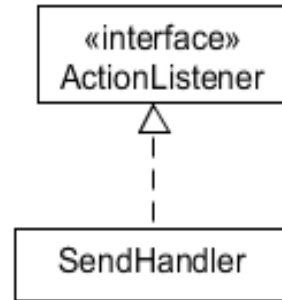
What does this diagram mean?

Summary of
UML Class Diagram Arrows

Inheritance (is-a)

```
Message
  △
  |
MessageWith
Attachments
```

Interface Implementation (is-a)

```
«interface»
ActionListener
  △
  ¦
SendHandler
```

Association (has-a-field)

```
MailPanel
  ↑
  |
MailFrame
```

Dependency (depends-on)

```
Dimension
  ↑
  ¦
MailFrame
```

```
User  ←——→  MailBox
```
Two-way Association
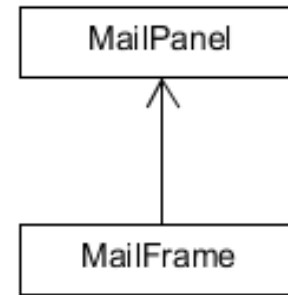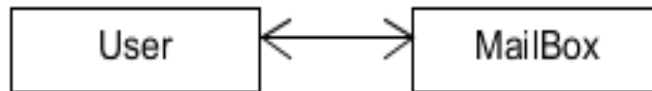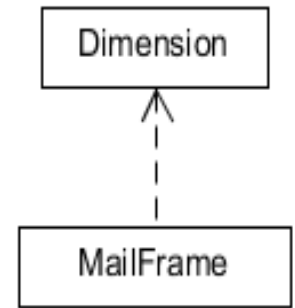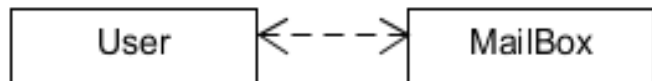
```
User  ←- - -→  MailBox
```
Two-Way Dependency
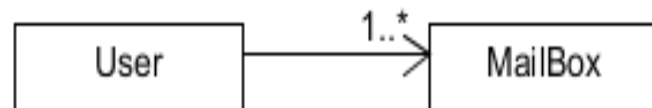
```
User  ——1..*—→  MailBox
```
Cardinality
(one-to-one, one-to-many)
One-to-many is shown on left

Q1

Reading & writing files
When the unexpected happens

FILES AND EXCEPTIONS

File I/O: Key Pieces

- Input: **File** and **Scanner**

- Output: **PrintWriter** and **println**

- ☺ Be kind to your OS: **close()** all files

- Letting users choose: **JFileChooser** and

  File

- Expect the unexpected: **Exception** handling

- Refer to examples when you need to...

Live code a level loader

Exception – What, When, Why, How?

- What:
  - Used to signal that something in the code has gone wrong
- When:
  - An error has occurred that cannot be handled in the current code
- Why:
  - Breaks the execution flow and passes exception up the stack

Exception – How?

- Throwing an exception:

throw new EOFException("Missing column");

- Handling (catching) an exception:

```
try {
        //code that could throw an exception
}
catch (ExceptionType ex) {
        //code to handle exception
}
```

- When caught you can:
  - Recover from the error OR exit gracefully

Q5

What happens when no exception is thrown?

```
Scanner inScanner;
try {
        inScanner =
                new Scanner(new File("test.txt");
        //code for reading lines
} catch (IOException ex) {
        JOptionPane.
                showMessageDialog("File not found.");
} finally {
        inScanner.close();
}
```

If this line is successful

Code continues on

The catch never executes

This runs after code in try completes

What happens when exception is thrown?

```
Scanner inScanner;
try {
        inScanner =
                new Scanner(new File("test.txt");
        //code for reading lines
} catch (IOException ex) {
        JOptionPane.
                showMessageDialog("File not found.");
} finally {
        inScanner.close();
}
```

If this line throws exception

Code after exception never executes

This is the next line executed

After catch is executed, this runs

When exception is not handled?

```java
public String readData(String filename)
                        throws IOException {
    Scanner inScanner =
            new Scanner(new File(filename));
    //code for reading lines
    inScanner.close();
}
```
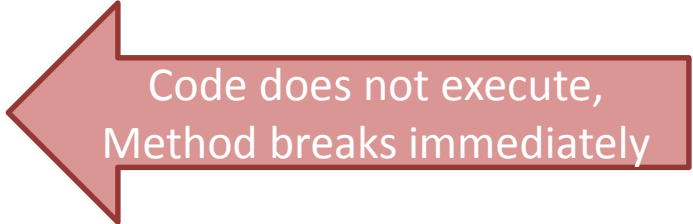
If this line throws exception

Code does not execute,
Method breaks immediately

main -> readAllFiles -> readData

If unhandled, exception bounces to
method that called it, then up the chain.

A Checkered Past

- Java has two sorts of exceptions

1. **Checked exceptions**: compiler **checks** that calling code isn't ignoring the problem
   - Used for **expected** problems

2. **Unchecked exceptions**: compiler lets us ignore these if we want
   - Used for fatal or avoidable problems
   - Are subclasses of RunTimeException or Error

A Tale of Two Choices

Dealing with checked exceptions

1. Can **propagate** the exception
   - Just declare that our method will pass any exceptions along…
   -   `public void loadGameState() throws IOException`

   - Used when our code isn't able to rectify the problem

2. Can handle the exception
   - Used when our code can rectify the problem

Q6

Handling Exceptions

- Use try-catch statement:

```
try {
    // potentially "exceptional" code
} catch (ExceptionType var) {
    // handle exception
}
```

Can repeat this part for as many different exception types as you need.

Related, try-finally for clean up:

```
try {
    // code that requires "clean up"
} // then maybe some catches
finally {
    // runs even if exception occurred
}
```

Q7

Exception Activity

- Look at the code in FileAverage, focusing on the use of exceptions
- Solve the problems in FileBestScore

Exam 2

- Paper part (~50 pts) includes:
- Questions about UML (~9 points)
- Questions about coupling, cohesion (~4 points)
- 1-2 Design Problem (~12 points)
- Question about exceptions (~4 points)
- Compile/runtime/printing question (~11 points)
- Tracing a recursive function (~10 points)
- **You can bring 1 sheet of notes + OO Principles for 220 + UML Cheatsheet**

Exam 2

- Computer part includes:
- Recursion
- Problem where you must use inheritance or interfaces to remove code duplication
- Problem where you have to layout a GUI and handle updates using listeners

[BallWorlds](#) DUE TONIGHT AT

Don't forget!

11:55pm

Bring review questions for Wednesday