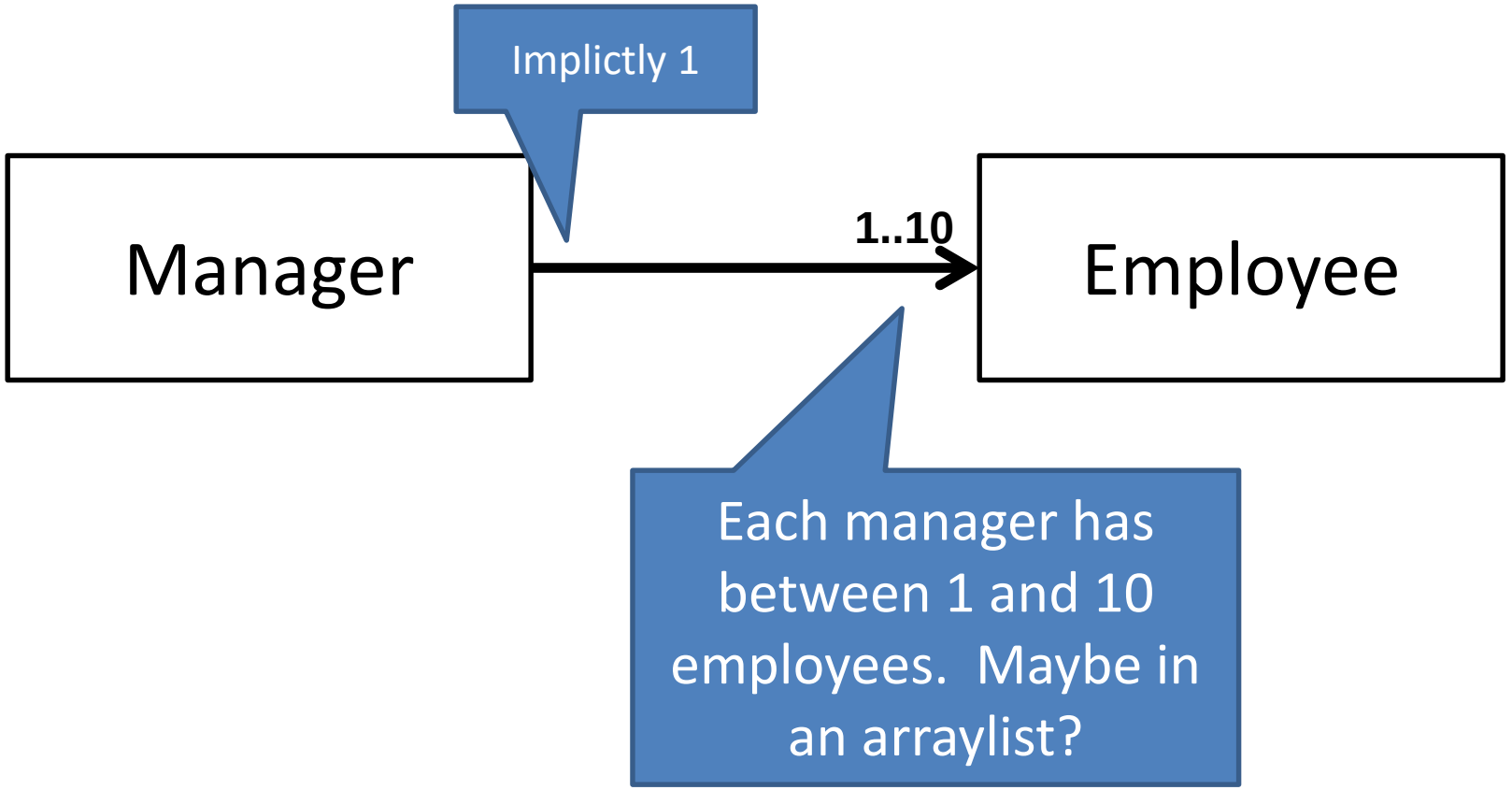
CSSE 220

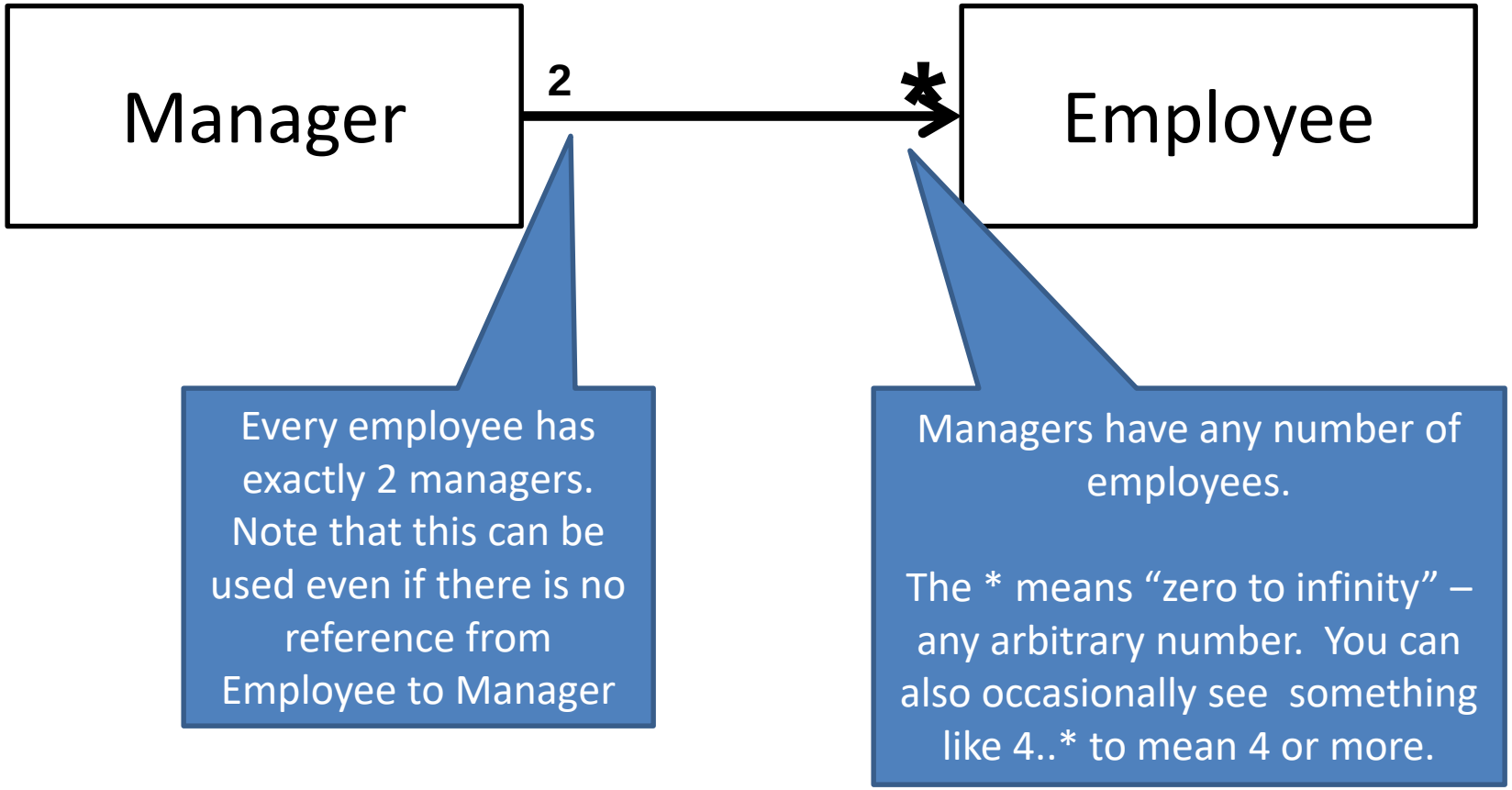
Object-Oriented Design
Files & Exceptions

Check out *FilesAndExceptions* from SVN

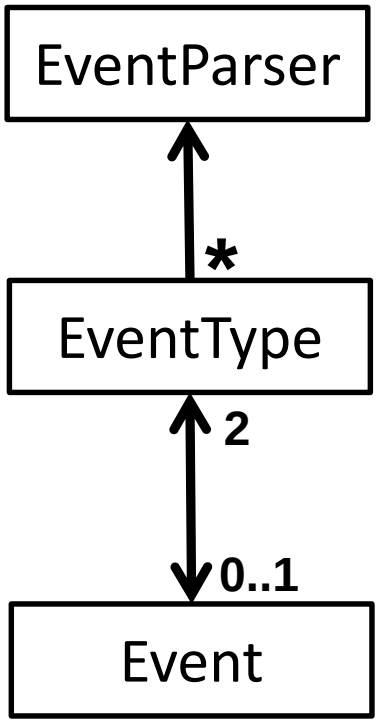
New UML Notation: Cardinality



More Cardinality

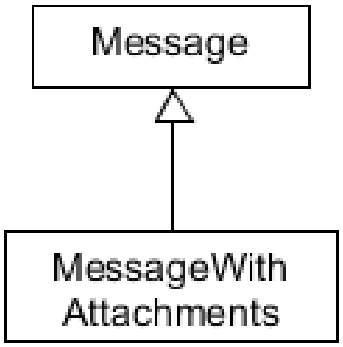


What does this diagram mean?

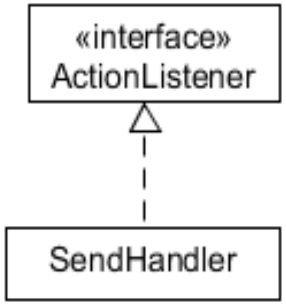


Summary of UML Class Diagram Arrows

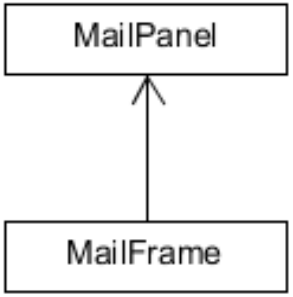
Inheritance (is-a)



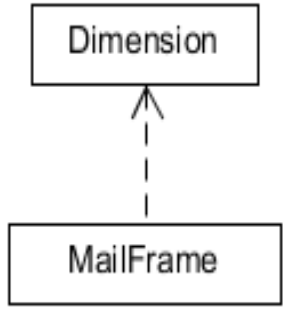
Interface Implementation (is-a)

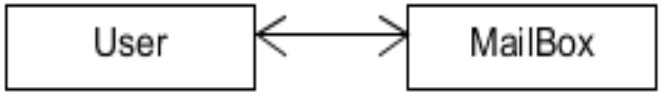


Association (has-a-field)

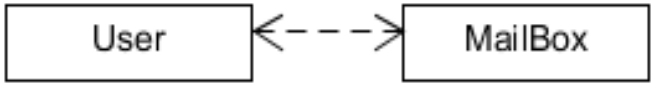


Dependency (depends-on)

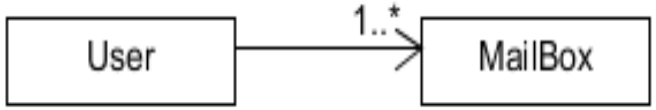




Two-way Association



Two-Way Dependency



Cardinality
(one-to-one, one-to-many)
One-to-many is shown on left

Q2

Reading & writing files

When the unexpected happens

FILES AND EXCEPTIONS

Review of Anonymous Classes

- Look at GameOfLifeWithIO
  - GameOfLife constructor has 2 listeners, two *local anonymous* class
  - ButtonPanel constructor has 3 listeners which are *local anonymous* classes

- Feel free to use as examples for your project

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…

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

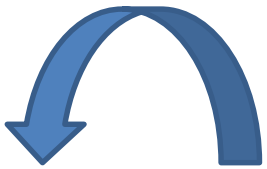
After catch is executed, this runs

When exception is not handled?

```
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



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

main -> readAllFiles -> readData

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

1. **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

1. 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"
} 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