CARL VON OSSIETZKY universität OLDENBURG

Sommersemester 2020

Objektorientierte Modellierung und Programmierung

Dr. Christian Schönberg

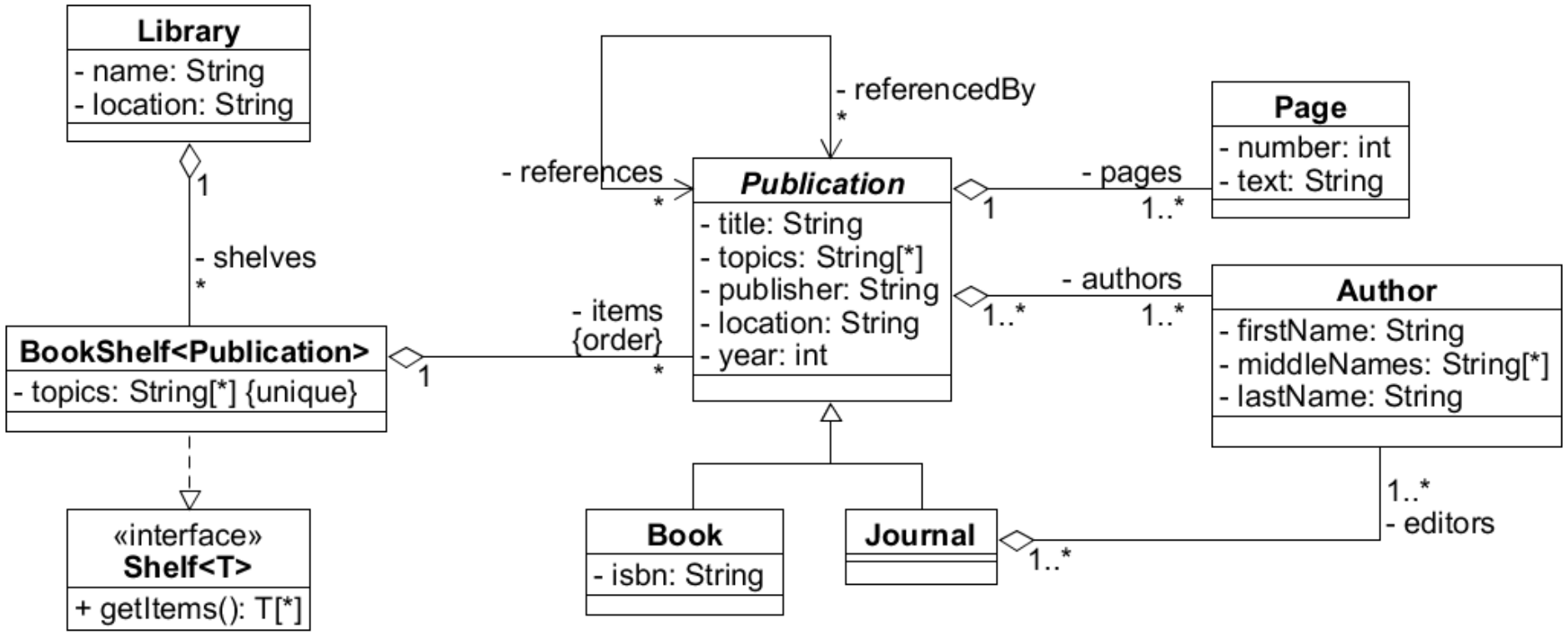
CARL VON OSSIETZKY universität OLDENBURG

Großübung

Modellierung in der UML

- Bibliothek
  - Regale, insb. Regale für Bücher und Zeitschriften
  - Bücher, Zeitschriften
  - Seiten
  - Autoren
  - Literaturverweise

Beispiellösung



CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
public interface Shelf<T> {

    Collection<T> getItems();

}
```

```
public class BookShelf implements Shelf<Publication> {

    private Set<String> topics = new HashSet<>();
    private List<Publication> items = new ArrayList<>();

    public Set<String> getTopics() {
        return topics;
    }

    public List<Publication> getItems() {
        return items;
    }

}
```

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
public abstract class Publication {

    private List<Publication> references = new ArrayList<>();
    private List<Publication> referencedBy = new ArrayList<>();

    public Collection<Publication> getReferences() {
        return Collections.unmodifiableList(references);
    }

    public Collection<Publication> getReferencedBy() {
        return Collections.unmodifiableList(referencedBy);
    }

    public void addReference(Publication to) {
        references.add(to);
        to.referencedBy.add(this);
    }

}
```

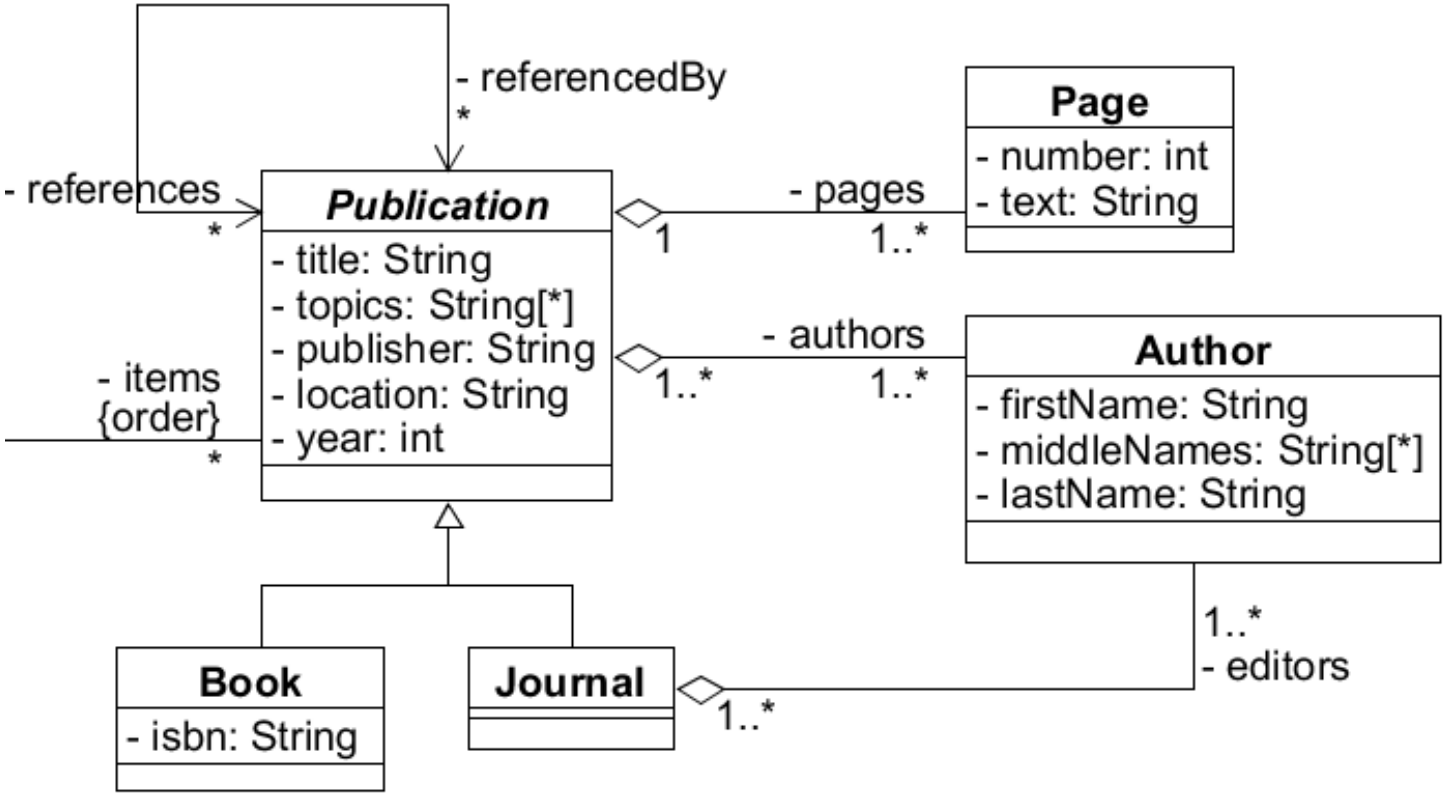
CARL VON OSSIETZKY universität OLDENBURG

Java Streams

- Gib die Titel aller Publikationen von Alan Turing aus, sortiert nach Datum
- Gib aus, wie viele Bücher mit mehr als 100 Seiten es gibt, die mindestens zwei Autoren haben
- Gib die Namen aller Autoren aus, die Publikationen geschrieben haben,
  die von mindestens drei anderen Publikationen referenziert werden
  - optional: diese Publikationen dürfen keinen Autor mit der referenzierten Publikation gemeinsam haben (keine Selbstreferenzen)

Java Streams

- Gib die Titel aller Publikationen von Alan Turing aus, sortiert nach Datum
- Gib aus, wie viele Bücher mit mehr als 100 Seiten es gibt, die mindestens zwei Autoren haben
- Gib die Namen aller Autoren aus, die Publikationen geschrieben haben, die von mindestens drei anderen Publikationen referenziert werden
  - optional: diese Publikationen dürfen keinen Autor mit der referenzierten Publikation gemeinsam haben (keine Selbstreferenzen)




```
shelf.getItems().stream()...
```

→ `Stream<Publication>`

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
// Gib die Titel aller Publikationen von Alan Turing aus,
// sortiert nach Datum
shelf.getItems().stream()
    .filter((p) -> p.getAuthors().contains(alanTuring))
    .sorted((p1, p2) -> Integer.compare(p1.getYear(), p2.getYear()))
    .forEach((p) -> System.out.println(p.getTitle()));
```

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
// Gib aus, wie viele Bücher mit mehr als 100 Seiten es gibt,
// die mindestens zwei Autoren haben
long count = shelf.getItems().stream()
    .filter((p) -> p instanceof Book)
    .filter((p) -> p.getPages().size() > 100)
    .filter((p) -> p.getAuthors().size() >= 2)
    .count();
System.out.println(count);
```

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
// Gib die Namen aller Autoren aus, die Publikationen geschrieben haben,
// die von mindestens drei anderen Publikationen referenziert werden
shelf.getItems().stream()
    .filter((p) -> p.getReferencedBy().size() >= 3)
    .map((p) -> p.getAuthors().stream())
    .reduce((s1, s2) -> Stream.concat(s1, s2))
    .orElse(Stream.empty())
    .distinct()
    .forEach((a) -> System.out.println(a));
```

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
// Gib die Namen aller Autoren aus, die Publikationen geschrieben haben,
// die von mindestens drei anderen Publikationen referenziert werden
// optional: diese Publikationen dürfen keinen Autor mit der referenzierten
// Publikation gemeinsam haben (keine Selbstreferenzen)
shelf.getItems().stream()
    .filter((p) -> p.getReferencedBy().stream().filter(
            (r) -> { return !intersection(p.getAuthors(), r.getAuthors()); }
        ).count() >= 3)
    .map((p) -> p.getAuthors().stream())
    .reduce((s1, s2) -> Stream.concat(s1, s2))
    .orElse(Stream.empty())
    .distinct()
    .forEach((a) -> System.out.println(a));
```

```
private static <T> boolean intersection(Collection<T> c1, Collection<T> c2) {
    for (T d1 : c1) {
        if (c2.contains(d1)) { return true; }
    }
    return false;
}
```

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
// Hole den Stream aus der Bibliothek, nicht direkt aus einem Regal
lib.getShelves().stream()
    .map((s) -> s.getItems().stream())
    .reduce((s1, s2) -> Stream.concat(s1, s2))
    .orElse(Stream.empty())
    .forEach((p) -> System.out.println(p.getTitle()));
```

CARL VON OSSIETZKY universität OLDENBURG

I/O

- Schreibe eine Publikation (mit Seiten, Autoren und ein- und ausgehende Referenzen) in eine Datei
- Lade die Publikation wieder aus der Datei und gebe Titel, Autoren und Anzahl der ein- und ausgehenden Referenzen aus

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
public abstract class Publication implements Serializable { … }
```

```
public class Page implements Serializable { … }
```

```
public class Author implements Serializable { … }
```

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
// Schreibe eine Publikation (mit Seiten, Autoren und ein- und
// ausgehenden Referenzen) in eine Datei
try (ObjectOutputStream out = new ObjectOutputStream(
        new BufferedOutputStream(new FileOutputStream("lib.ser")))) {
    out.writeObject(turing1);
} catch (FileNotFoundException e) {
    System.err.println("The file could not be opened!");
} catch (IOException e) {
    System.err.println("Error writing to the file!");
}
```

CARL VON OSSIETZKY universität OLDENBURG

Beispiellösung

```
// Lade die Publikation wieder aus der Datei und gebe Titel,
// Autoren und Anzahl der ein- und ausgehenden Referenzen aus
try (ObjectInputStream in = new ObjectInputStream(
        new BufferedInputStream(new FileInputStream("lib.ser")))) {
    Publication pub = (Publication) in.readObject();
    System.out.println(pub.getTitle());
    System.out.println(pub.getAuthors());
    System.out.println(pub.getReferences().size());
    System.out.println(pub.getReferencedBy().size());
} catch (FileNotFoundException e) {
    System.err.println("The file could not be opened!");
} catch (IOException e) {
    System.err.println("Error reading from the file!");
} catch (ClassNotFoundException e) {
    System.err.println("Java class not available!");
}
```