Softwaretechnologie, Übung 5

Henry Haustein

Aufgabe 1

(a) Mit Mengen

```
public class Buch implements Comparable<Buch> {
  private String isbn;
  private String autor;
  private String title;

  public Buch(String isbn, String autor, String title) {
    this.isbn = isbn;
    this.autor = autor;
    this.title = title;
  }

  public Buch(String isbn) {
    this.isbn = isbn;
  }

  public String toString() {
    return isbn + " - " + title;
  }

  public String getISBN() {
    return isbn;
  }

  public String getAuthor() {
    return autor;
  }

  public String getTitle() {
    return title;
  }

  public int compareTo(Buch buch){
    return this.isbn.compareTo(buch.getISBN());
  }

  @Override
  public boolean equals(Object obj){
```

```java
    if (obj instanceOf Buch) {
      Buch buch = (Buch) obj;
      return isbn.equals(buch.isbn);
    }
    return false;
  }

  @Override
  public int hashCode(){
    return isbn.hashCode();
  }
}

public class Bibliothek {

  private Set<Buch> bestand = new TreeSet<>();

  public boolean sortiertesEinfuegen(Buch neuesBuch){
    //sortiert nach ISBN, automatisch sortiert
    return bestand.add(neuesBuch);
  }

  public Buch sucheNachBuch(String isbn){
    for (Buch buch : bestand) {
      if (buch.getIsbn().equals(isbn)){
        return buch;
      }
    }
  }

  // Alterntive mit Iterator
  public Buch sucheNachISBN(String isbn) {
    Iterator<Buch> iter = bestand.iterator();
    while (iter.hasNext()) {
      Buch buch = iter.next();
      if (buch.getIsbn().equals(isbn)){
        return buch;
      }
    }
  }

  public Collection<Buch> sucheNachAutor(String autor){
    Collection<Buch> authorBooks = new TreeSet<Buch>();
    for (Book b : bestand) {
      if (b.getAuthor().equals(autor)) {
        authorBooks.add(b);
      }
    }
    return authorBooks;
  }
}
```

(b) Quelltext

```java
public Collection<String> getAllAuthors() {
  Set<String> result = new HashSet<String>();
  for (Buch b : bestand) {
    result.add(b.getAuthor());
  }
  return result;
}

public Map<String, Set<Buch>> bestandNachAutor() {
  Map<String, Set<Buch>> map = new TreeMap<String, Set<Buch>>();
  Collection<String> autorenliste = getAllAuthors();
  for (String autor : autorenliste) {
    Set<Buch> buecher = sucheNachAutor(autor);
    map.put(autor, buecher);
  }
  return map;
}

// Alternative mit nur einer benoetigten Funktion
public Map<String, Set<Buch>> bestandNachAutor() {
  Map<String, Set<Buch>> map = new TreeMap<String, Set<Buch>>();
  for (Buch buch : bestand) {
    Set<Buch> buecher = map.get(buch.getAutor());
    if (buecher == null) {
      buecher = new TreeSet<Buch>();
      map.put(buch.getAutor(), buecher);
    }
  // Call-by-reference
  buecher.add(buch);
  }
}
```

Aufgabe 2

Datei Book.java

```java
package collections3;

public class Book implements Comparable<Book>{
  private String isbn;
  private String author;
  private String title;

  public Book(String isbn, String author, String title){
    if (isbn == null | author == null | title == null) {
      throw new IllegalArgumentException("Fill the fucking arguments.");
    } else {
      this.isbn = isbn;
      this.author = author;
      this.title = title;
    }
  }

  public Book(String isbn){
    if (isbn == null) {
      throw new IllegalArgumentException("Fill the fucking arguments.");
    } else {
      this.isbn = isbn;
      this.author = "";
      this.title = "";
    }
  }

  public String getIsbn() {
    return this.isbn;
  }

  public void setIsbn(String isbn) {
    this.isbn = isbn;
  }

  public String getAuthor() {
    return this.author;
  }

  public void setAuthor(String author) {
    this.author = author;
  }

  public String getTitle() {
    return this.title;
  }

  public void setTitle(String title) {
    this.title = title;
  }

  public String toString() {
    return this.getTitle() + " by " + this.getAuthor() + " (ISBN " + this.getIsbn() + ")";
  }

  public int hashCode(){
    return isbn.hashCode();
  }

  @Override
  public boolean equals(Object o) {
    if (o instanceof Book) {
      Book book = (Book) o;
      return this.isbn.equals(book.getIsbn());
    }
    return false;
  }

  @Override
  public int compareTo(Book book) {
    return this.isbn.compareTo(book.getIsbn());
  }
}
```

Datei `Library.java`

```
package collections3;

import java.util.*;

public class Library {
  private Map<String, Set<Book>> stock;

  public Library(){
    stock = new TreeMap<String, Set<Book>>();
  }

  public Collection<Book> getStock(){
    Set<Book> output = new TreeSet<Book>();;
    for (Map.Entry<String, Set<Book>> entry : stock.entrySet()) {
      output.addAll(entry.getValue());
    }
    return output;
  }

  // alternativ, mit stock.values()
  public Collection<Book> getStock() {
    Collection<Book> books = new TreeSet<Book>();
    for (Collection<Book> b : this.stock.values()) {
      books.addAll(b);
    }
    return books;
  }

  public boolean insertBook(Book toInsertBook){
    Set<Book> tempSet = new TreeSet<Book>();
    tempSet.add(toInsertBook);
    if(stock.containsKey(toInsertBook.getAuthor())){
      Set<Book> savedSet = stock.get(toInsertBook.getAuthor());
      if(savedSet.contains(toInsertBook)){
        return false;
      } else {
        savedSet.add(toInsertBook);
        stock.put(toInsertBook.getAuthor(), savedSet);
        //replace - stellt sicher, dass Autor schon da ist
        stock.replace(toInsertBook.getAuthor(), savedSet);
      }
    } else {
      stock.put(toInsertBook.getAuthor(), tempSet);
    }
    return true;
  }
  //inloop approved, aber sehr umstaendlich

  // alterantiv
  public boolean insertBook_2(Book book){
    if (!stock.containsKey(book.getAuthor())){
      stock.put(book.getAuthor(), new TreeMap<Buch>());
    }
    Set<Buch> booksOfAuthor = stock.get(book.getAuthor());
    return booksOfAuthor.add(book);
  }

  // alternativ
  public boolean insertBook(Book newBook) {
    if (stock.containsKey(newBook.getAuthor())) {
      // Autor gibt es bereits
      Set<Book> authorBooks = stock.get(newBook.getAuthor());
      boolean result = authorBooks.add(newBook);
      stock.put(newBook.getAuthor(), authorBooks);
      return result;
    }
    else {
      // Autor muss noch hinzugefuegt werden
      Set<Book> authorBooks = new TreeSet<Book>();
      authorBooks.add(newBook);
      stock.put(newBook.getAuthor(), authorBooks);
      return true;
    }
  }

  public Book searchForIsbn(String isbn){
    for (Map.Entry<String, Set<Book>> entry : stock.entrySet()) {
      for(Book book : entry.getValue()){
        if(book.getIsbn().equals(isbn)){
          return book;
        }
      }
    }
    return null;
  }

```

```java
  // alternativ
  public Book searchForIsbn_2(String isbn){
    for (Set<Buch> books : stock.values()){
      for (Book book : books ) {
        if (book.getIsbn().equals(isbn)){
          return book;
        }
      }
    }
    return null;
  }

  public Map<String,Set<Book>> listStockByAuthor(){
    return stock;
  }

  public Collection<Book> searchForAuthor(String author){
    Set<Book> output = new TreeSet<Book>();
    if(!stock.isEmpty()){
      if(stock.get(author) == null){
        return output;
      }
      return stock.get(author);
    } else {
      return output;
    }
  }

  // alternativ
  public Collection<Book> searchForAuthor(String author) {
    if (stock.get(author) == null) {
      //alternativ: return Collections.emptySet();
      return new TreeSet<Book>();
    }
    else {
      return stock.get(author);
    }
  }
}
```