

Szenario

Sie sollen für die örtliche Bücherei eine neue Softwareanwendung mit den folgenden zentralen Kernfunktionen entwickeln:

- Alle Bücher der Bibliothek in einer Übersicht anzeigen
- Details (Id, Name des Autors, Titel, Status (*verfügbar* oder *verliehen*)) zu einem Buch anzeigen
- Neue Bücher anlegen
- Bücher verleihen und zurücknehmen

Sie haben bereits das Design der Softwareanwendung erstellt (Klassendiagramm und Oberflächen-Entwürfe) und sollen sich nun an die Umsetzung des Designs machen.

Aufbau der Klausur

Der erste Teil der Klausur beinhaltet Theoriefragen zum gegebenen Szenario. Das zugehörige Klassendiagramm, sowie die Oberflächen-Entwürfe sind der Klausur beigelegt. Im zweiten Teil der Klausur sollen Sie Teile des Klassendiagramms und der Oberflächen-Entwürfe mit Hilfe der Programmiersprache Java umsetzen.

Hinweise zur Klausur

- Lesen Sie die Aufgabenbeschreibung sorgfältig durch und achten Sie darauf, Antworten möglichst kurz und knapp zu formulieren!
- Benötigte Klassen- oder Schnittstellen-Imports müssen von Ihnen nicht explizit angegeben werden!
- Schreiben Sie Ihre Antworten und Ihr Coding ausschließlich auf die Aufgabenblätter! Notfalls können Sie auch die Rückseite verwenden!
- Insgesamt werden in der Klausur 116 Punkte vergeben (44 Punkte im Theorieteil, 72 Punkte im Praxisteil), zur Berechnung der Note wird aber ein 100-Punkte-Schlüssel angewendet!

Benennen Sie die Komponenten der Klasse *Book*, auf die innerhalb der Klasse *PaperBook* zugegriffen werden kann!

[illegible]

Kreuzen Sie an, welche der folgenden Aussagen richtig sind und welche falsch!
Richtige Antworten geben 1 Pluspunkt, falsche Antworten 1 Minuspunkt.

Aussage	Richtig	Falsch
Die Klasse <i>Book</i> ist die Oberklasse der Klasse <i>EBook</i> .		
Die Klasse <i>Book</i> leitet die Klasse <i>PaperBook</i> ab.		
Die Klasse <i>Book</i> kann instanziiert werden.		
Die Klasse <i>PaperBook</i> könnte eine abstrakte Methode definieren.		
Die Typumwandlung von der Klasse <i>Book</i> zur Klasse <i>EBook</i> wird als Upcast bezeichnet.		
Die Ausnahme <i>WrongFileSizeException</i> muss behandelt werden.		
Mit der Klasse <i>FileInputStream</i> können byte-orientierte Daten in eine Datei geschrieben werden.		
Der Datentyp <i>T</i> der Schnittstelle <i>Loanable</i> wird auch als formaler Typparameter bezeichnet.		
Das Attribut <i>books</i> der Klasse <i>Library</i> kann Bücher mit identischen Attributen aufnehmen.		
Objekte der Klasse <i>JCheckBox</i> können mit Hilfe der Klasse <i>ButtonGroup</i> gruppiert werden.		

Aufgabe 1.3 (2 Punkte)

Erläutern Sie kurz, warum in einer Schnittstelle die Angabe von *abstract* bei Methodendefinitionen nicht notwendig ist!

[illegible]

Aufgabe 1.4 (2 Punkt)

Erläutern Sie kurz, wozu sich die Klasse *Library* durch die Implementierung der Schnittstelle *Loanable* verpflichtet!

[illegible]

Aufgabe 1.5 (4 Punkte)

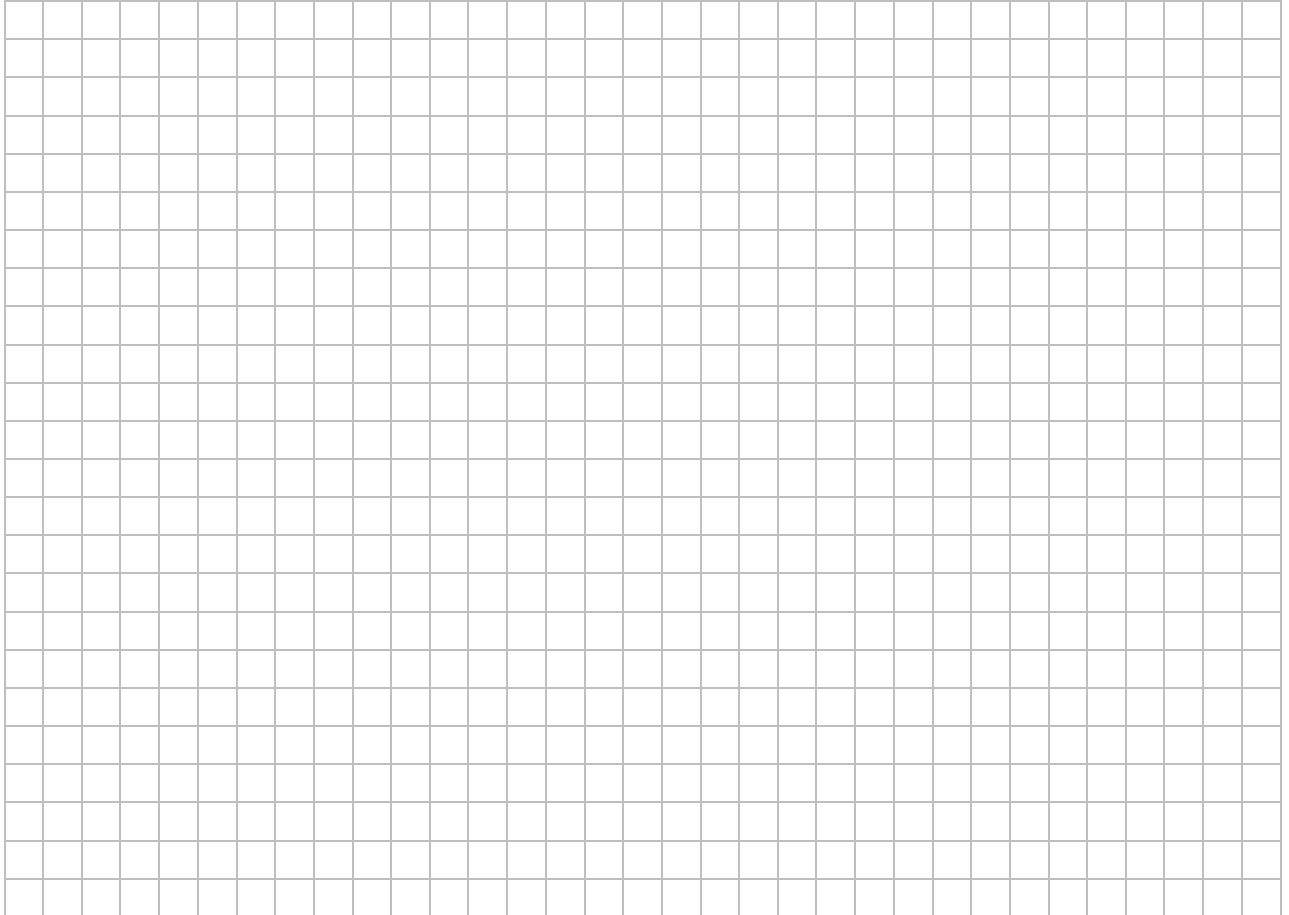
Geben Sie an, wie die Aufzählungskonstante *AZW* der Aufzählung *FileFormat* intern definiert ist und erläutern Sie kurz, warum diese so definiert ist!

Hinweis: Die Abkürzung *AZW* bezeichnet das *Amazon Kindle Format!*

A full-page sheet of graph paper featuring a uniform grid of thin, light gray lines on a white background. The grid consists of small squares covering the entire area.

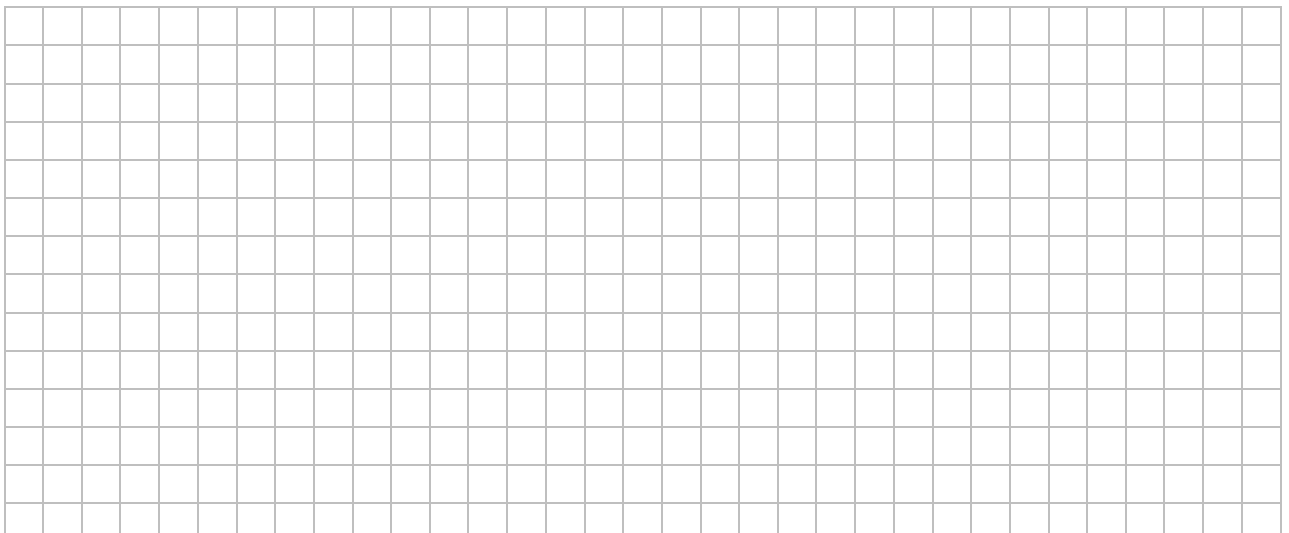
Aufgabe 1.6 (8 Punkte)

Benennen Sie die 4 Möglichkeiten, die Java zur Definition von inneren Klassen besitzt, erläutern Sie kurz, welche dieser Möglichkeiten für die Klasse *Author* in Frage kommen und erläutern Sie kurz, warum diese in Frage kommen!



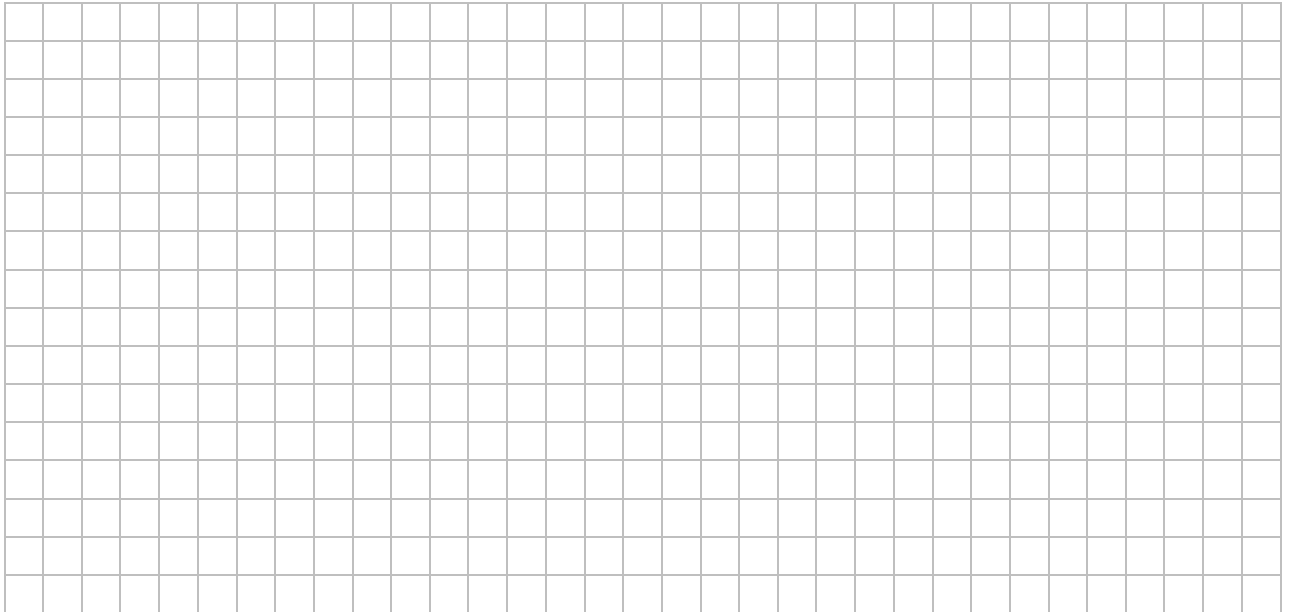
Aufgabe 1.7 (4 Punkte)

Skizzieren Sie den Ausnahmenbehandlungs-Prozess in Java und erläutern Sie kurz, was man unter der Catch-Or-Throw-Regel versteht!



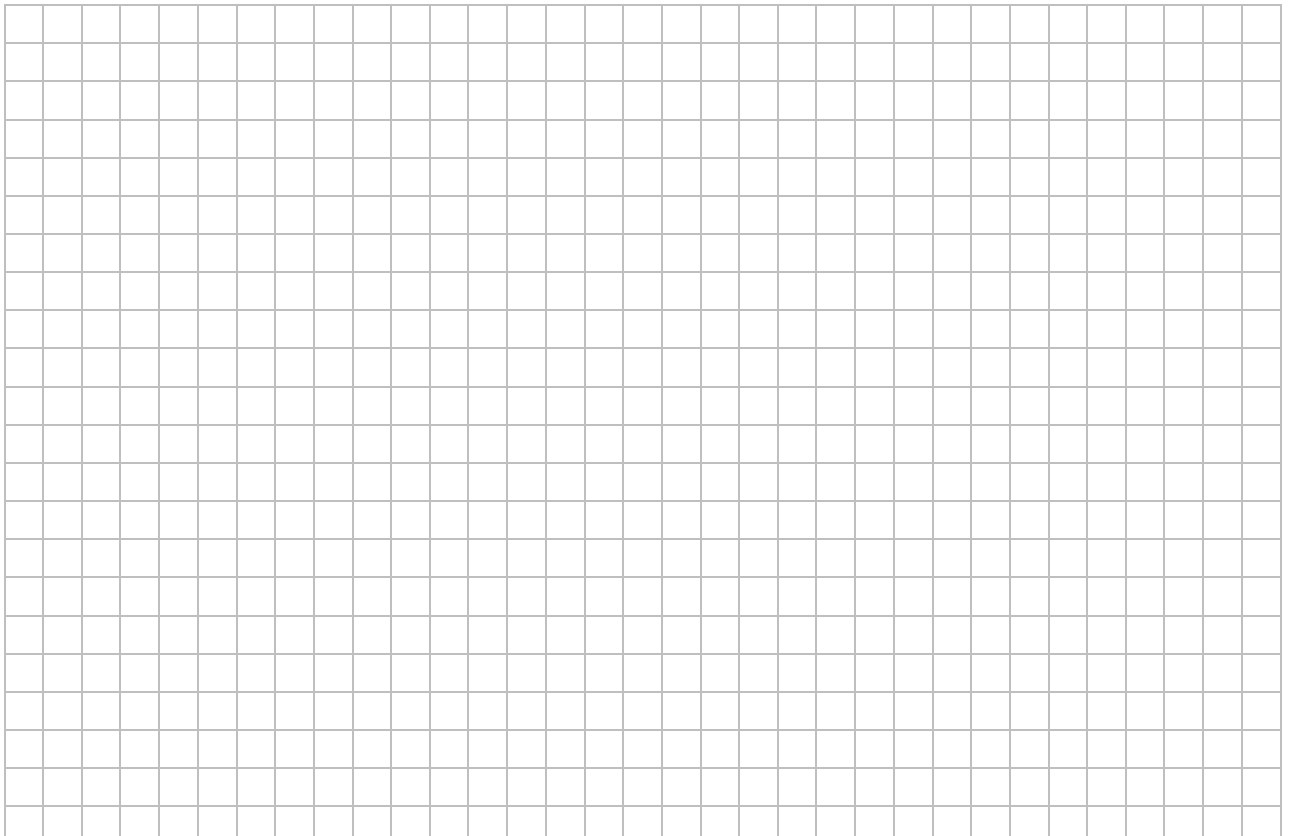
Aufgabe 1.8 (4 Punkt)

Erläutern Sie kurz, wozu die Klasse *Book* die Schnittstelle *Serializable* implementiert und erläutern Sie kurz, wozu dies im gegebenen Szenario verwendet werden könnte!



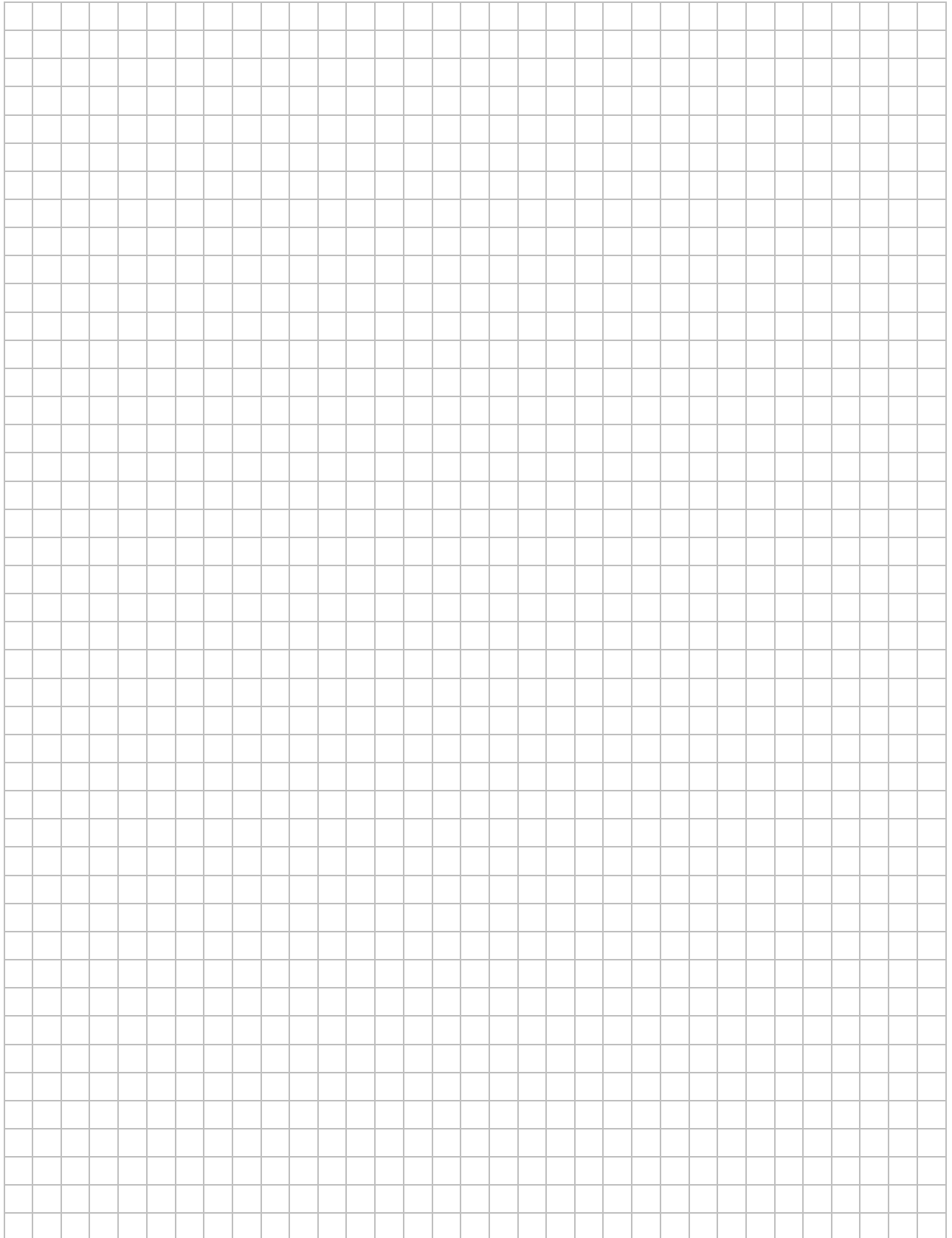
Aufgabe 1.9 (4 Punkt)

Benennen Sie den Varianztyp, der in der Klasse *Library* bei der Implementierung der Schnittstelle *Loanable* für den formalen Typparameter *T* verwendet wird und erläutern Sie kurz, welche Typeinschränkungen dadurch entstehen.



Aufgabe 1.10 (4 Punkte)

Erläutern Sie kurz das Quicksort-Verfahren.

A large grid of graph paper, consisting of 20 columns and 30 rows of small squares, intended for the student to write their answer to the task.

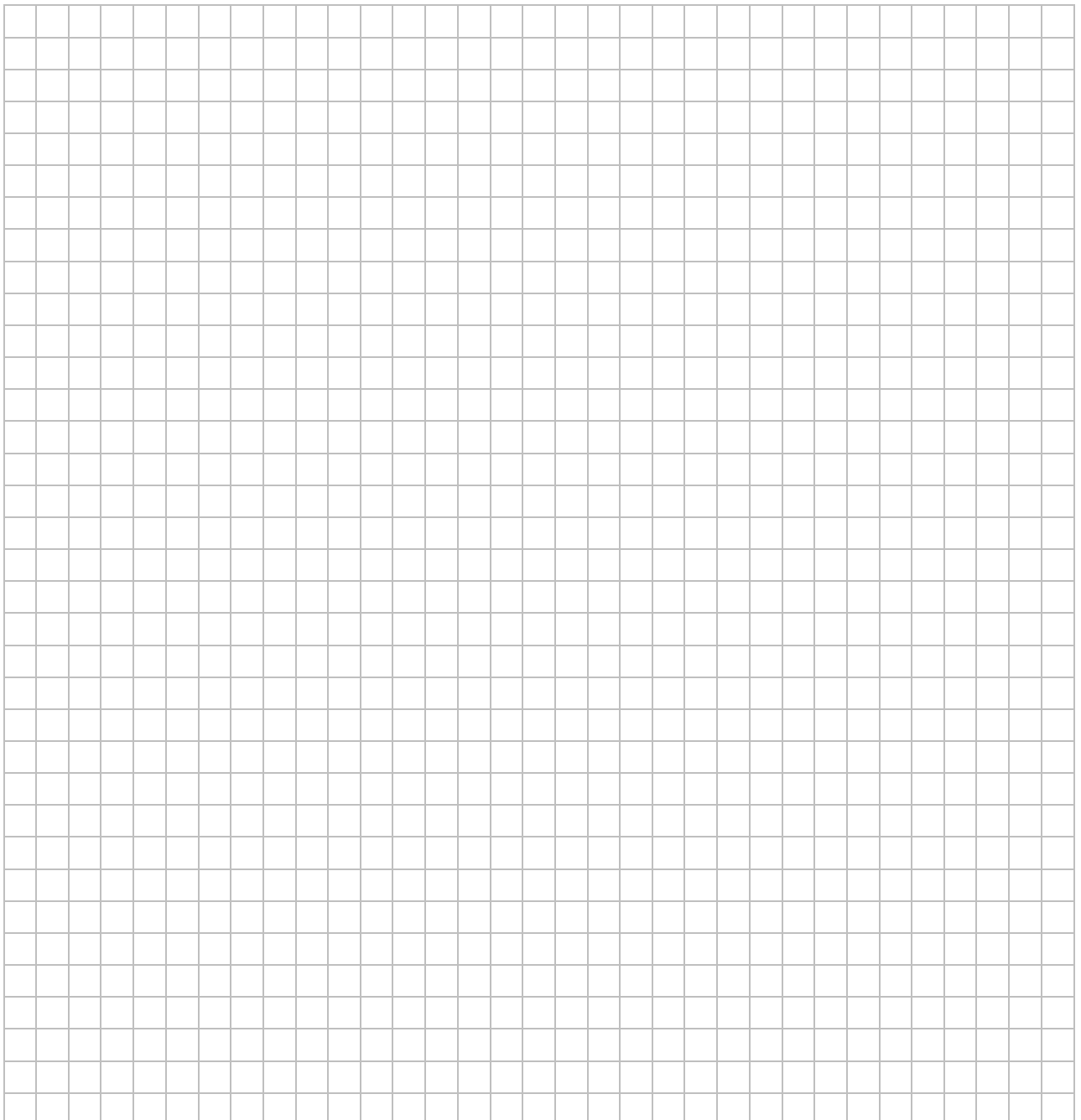
Teil 2: Praxis (72 Punkte)

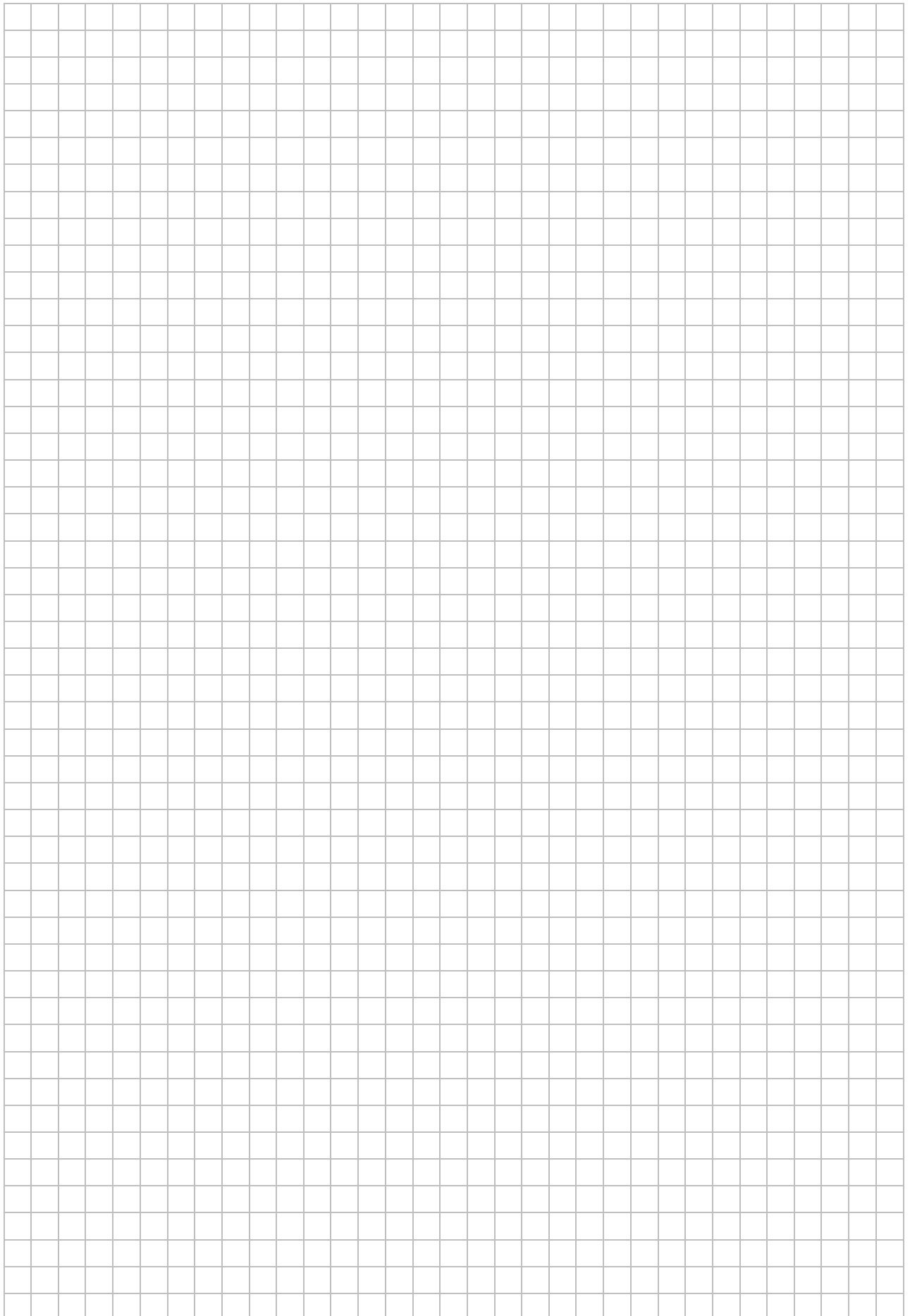
Aufgabe 2.1 (10 Punkte)

Erstellen Sie die Klasse *EBook* anhand des gegebenen Klassendiagramms.

Hinweise:

- Der Konstruktor soll alle Attribute initialisieren!
- Die Methode *getFileFormat()* soll das Dateiformat zurückgeben!
- Die Methode *getFileSize()* soll die Dateigröße zurückgeben!
- Bei einer Dateigröße kleiner gleich Null soll die Ausnahme *WrongFileSizeException* ausgelöst werden!





Aufgabe 2.2 (6 Punkte)

Erstellen Sie die Klasse *Author* als innere Klasse der Klasse *Book* anhand des gegebenen Klassendiagramms.

Hinweise:

- Der Konstruktor soll alle Attribute initialisieren!
- Die Methode *getName()* soll den Namen des Autoren zurückgeben!
- Die Methode *getNationality()* soll die Nationalität des Autoren zurückgeben!

```
public abstract class Book implements Serializable {  
    private String id;  
    private Author author;  
    private String title;  
    public Book(Author author, String title) {  
        id = UUID.randomUUID().toString();  
        this.author = author;  
        this.title = title;  
    }  
    public String getId() { return id; }  
    public Author getAuthor() { return author; }  
    public String getTitle() { return title; }  
}
```

}

Aufgabe 2.3 (8 Punkte)

Implementieren Sie die Methode *getBook(int)* der Klasse *Library*.

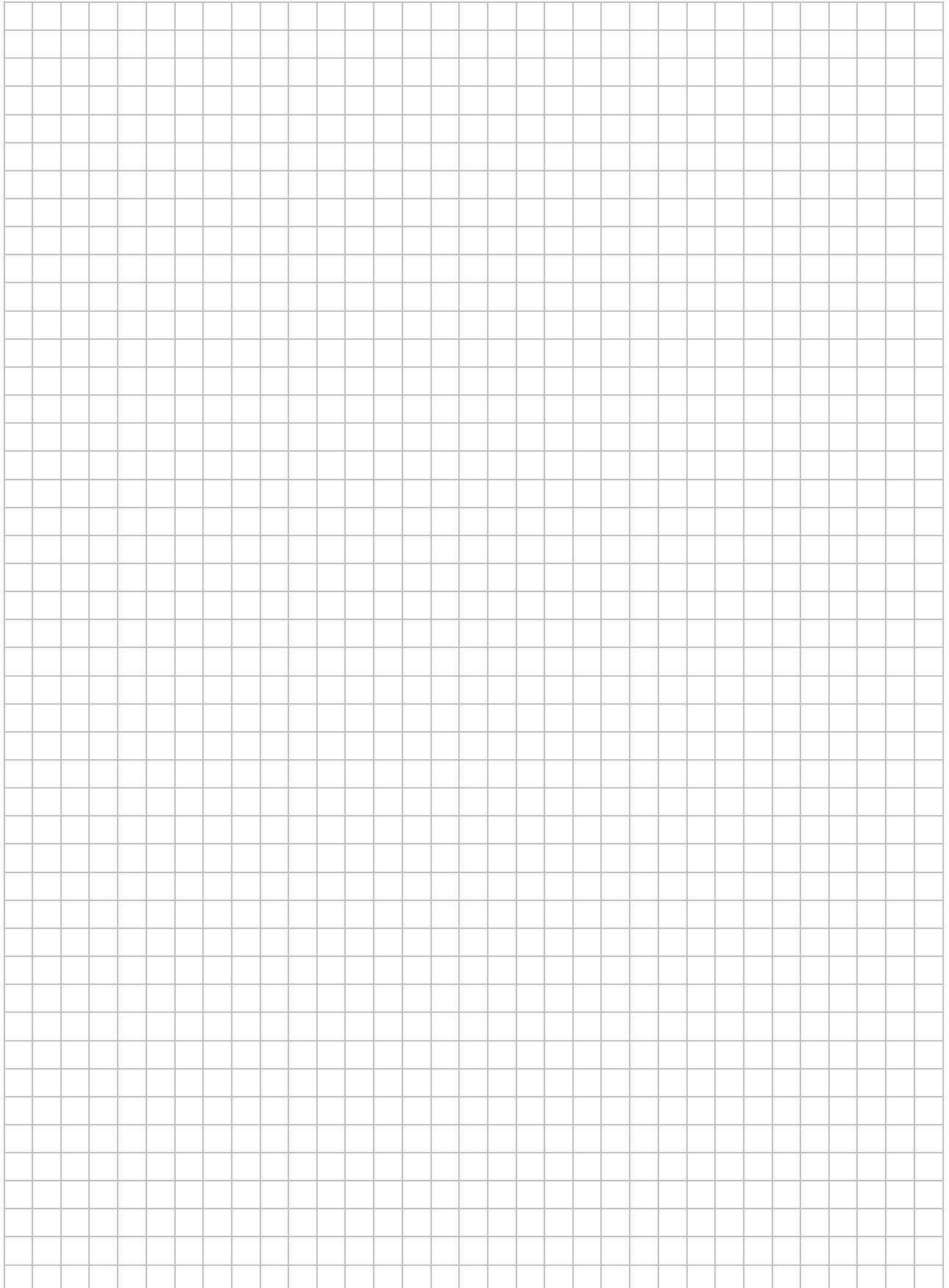
Hinweis: die Methode soll zur eingegebenen Bücher-Id den entsprechenden Eintrag des Assoziativspeichers *books* zurückgeben.

```
public class Library implements Loanable<Book> {  
    private String name;  
    private HashMap<Book, Status> books;  
    public Library(String name) {  
        books = new HashMap<>();  
        this.name = name;  
    }  
    ...  
    public HashMap<Book, Status> getBook(String id) {
```

$$\left. \begin{array}{l} \{ \\ \} \end{array} \right\}$$

Aufgabe 2.4 (6 Punkte)

Erstellen Sie die Schnittstelle *Loanable* anhand des gegebenen Klassendiagramms.



Aufgabe 2.5 (20 Punkte)

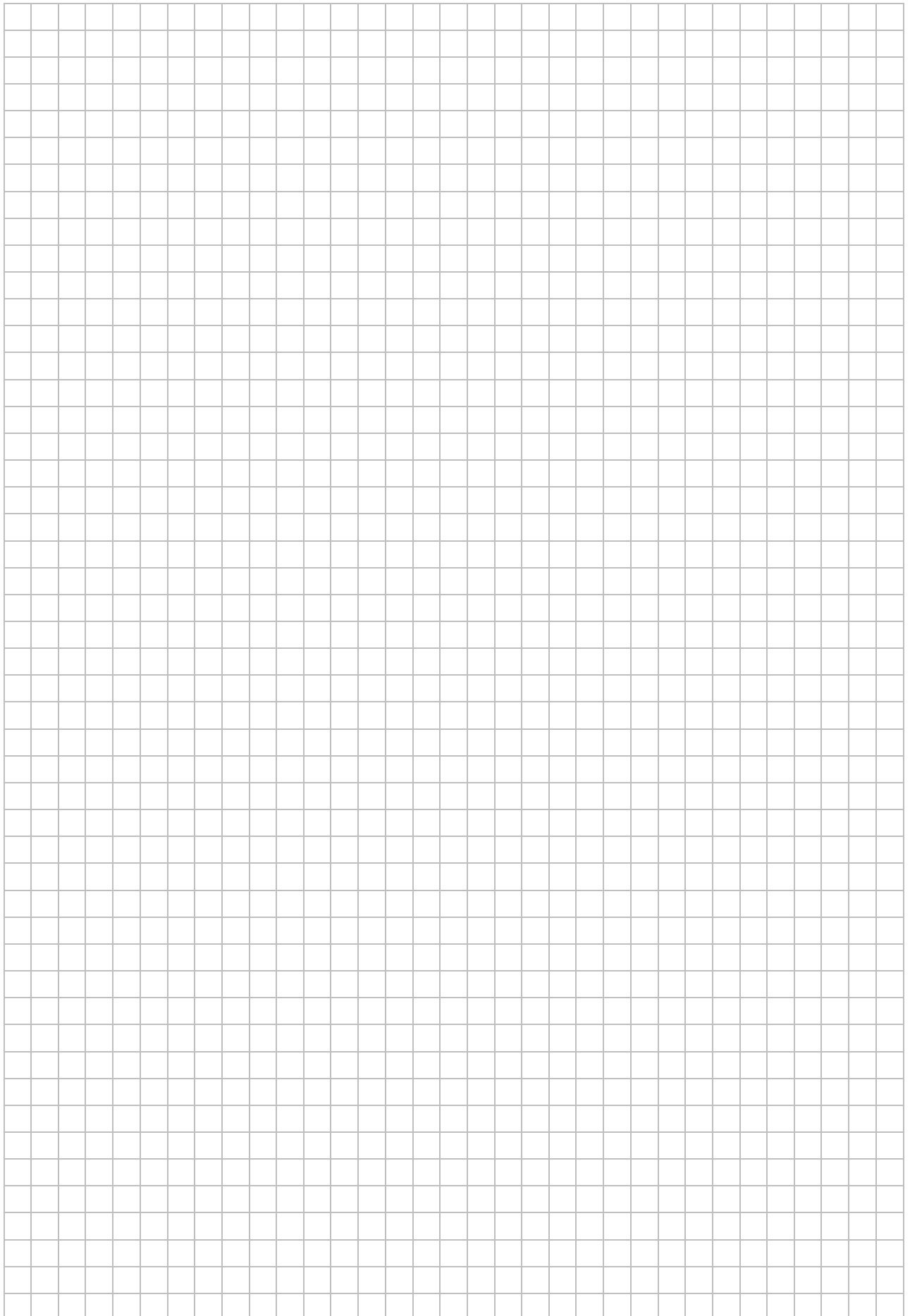
Erweitern Sie die Klasse *CreateBookDialog* so, dass die gegebene grafische Benutzeroberfläche erzeugt wird.

Hinweise:

- Sie können sich bei der Implementierung des Konstruktors an folgendem Aufbau orientieren:
 1. Auswahlgruppe festlegen
 2. Drop-Down-Liste für Autoren festlegen
 3. Drop-Down-Liste für Dateiformate festlegen
 4. Buchtyp-Container festlegen (Layout und Inhalte)
 5. Buchattribute-Container festlegen (Layout und Inhalte)
 6. Drucktasten-Container festlegen (Layout und Inhalte)
 7. Haupt-Container festlegen (Layout und Inhalte)
 8. Darstellung einzelner Oberflächenelemente anpassen
 - Der Autor und der Titel sollen immer eingabebereit sein
 - Das Dateiformat und die Dateigröße sollen nur bei eBooks eingabebereit sein
 - Die Anzahl Seiten soll nur bei „normalen“ Büchern eingabebereit sein
 9. Ereignisbehandler definieren und implementieren
- Die statische Methode *getAuthors()* der Klasse *DataProvider* gibt eine *ArrayList* mit allen Autoren zurück!
- Die Implementierung einer Ereignisbehandlung für die Drucktaste *createBookBttn* ist nicht erforderlich!
- Die Ereignisbehandlung soll anhand von lokalen oder anonymen Klassen erfolgen!

```
public class CreateBookDialog extends JDialog {
    private JLabel bookTypeLbl = new JLabel("Buchtyp:");
    private JRadioButton eBookRdBtn = new JRadioButton("eBook");
    private JRadioButton paperBookRdBtn = new JRadioButton("Buch");
    private ButtonGroup bookTypeButtonGroup = new ButtonGroup();
    private JPanel bookTypeContainer = new JPanel();
    private JLabel authorLbl = new JLabel("Autor:");
    private JComboBox<String> authorCmbBx = new JComboBox<>();
    private JLabel titleLbl = new JLabel("Titel:");
    private JTextField titleTxtFld = new JTextField();
    private JLabel fileFormatLbl = new JLabel("Dateiformat:");
    private JComboBox<String> fileFormatCmbBx = new JComboBox<>();
    private JLabel fileSizeLbl = new JLabel("Dateigröße:");
    private JTextField fileSizeTxtFld = new JTextField();
    private JLabel noPagesLbl = new JLabel("Anzahl Seiten:");
    private JTextField noPagesTxtFld = new JTextField();
    private JPanel bookAttributesContainer = new JPanel();
    private JButton createBookBtn = new JButton("Buch anlegen");
    private JPanel buttonContainer = new JPanel();
    private JPanel mainContainer = new JPanel();

    public CreateBookDialog () {
        setTitle("neues Buch anlegen");
        setSize(500, 325);
        setLocationRelativeTo(null);
        add(mainContainer);
    }
}
```



$$\left. \begin{array}{l} \{ \\ \} \end{array} \right\}$$

Aufgabe 2.6 (10 Punkte)

Implementieren Sie die Methode *actionPerformed(ActionEvent)* der Klasse *LibraryFrame* so, dass beim Betätigen der Drucktaste *loanOutBookBtn* das ausgewählte Buch ausgeliehen wird.

Hinweise:

- Beim Versuch, ein Buch auszuleihen, dass bereits verliehen ist, soll der gegebene Fehlerdialog angezeigt werden!
- Bei Erfolg soll das Ausgabefeld *statusLbl* entsprechend aktualisiert werden!
- Die Methode *getValueAt(int, int)* der Klasse *DefaultTableModel* gibt den Inhalt der eingegebenen Zelle (Zeile, Spalte) als Instanz der Klasse *Object* zurück!
- Die Methode *getSelectedRow()* der Klasse *JTable* gibt die ausgewählte Zeile als ganze Zahl zurück!
- Die statische Methode *showMessageDialog(Component, Object, String, int)* der Klasse *JOptionPane* erzeugt einen Nachrichtendialog!
- Verwenden Sie für den Nachrichtentyp die Konstante *JOptionPane.ERROR_MESSAGE*!

```
...
public class LibraryFrame extends JFrame implements
ListSelectionListener, ActionListener {
...
    private JButton loanOutBookBtn = new JButton("Buch ausleihen");
    private DefaultTableModel booksTblMdl = new DefaultTableModel();
    private JTable booksTbl = new JTable(booksTblMdl);
    private JLabel = new JLabel();
    private Library library;

    public LibraryFrame(Library library) {
        this.library = library;
...
    }

    public void actionPerformed(ActionEvent e) {
```


Aufgabe 2.7 (12 Punkte)

Wenden Sie das Quicksort-Verfahren auf die nachfolgende Buch-Id-Liste an.
Schreiben Sie hierzu für jeden Durchlauf die jeweilige Reihenfolge der Buch-Ids auf
und kreisen Sie den jeweiligen Teiler ein!

Hinweis: Verwenden Sie als Teiler jeweils das Ergebnis der Rechenoperation $\text{Index Links} + (\text{Index rechts} - \text{Index links}) / 2!$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7															
4															
23															
9															
20															
14															
17															
37															
2															
35															