Aufgabe 1: Sammlungen in Sammlung

In der Klasse *CDSammlung* (siehe Blatt 2) werden Sammlungen von CDs im Attribut *sammlung* vom Typ *HashMap* verwaltet. In der Hashmap ist die Sammlung der CDs eines Künstlers unter dem Namen des Künstlers abgespeichert. Die Sammlung der CDs eines Künstlers wird ebenfalls in Form einer Hashmap organisiert. Hierin wird der Name des Albums als Schlüssel für den Zugriff auf die Instanzen der Klasse *CD* verwendet.

Schreiben Sie die Methode `loeschen(String kuenstler, String album)`, die den String mit dem Künstlernamen und den Albumnamen übergeben bekommt und die zugehörige CD aus der Hashmap löscht. (4 Punkte)

Tipp: Eine kurze Zusammenfassung der wichtigsten Methoden für die Handhabung der Klasse HashMap finden Sie auf Blatt 4.

(3)

```
public void loeschen ( String kuenstler, String album ) {

    Sammlung.get(kuenstler).remove(album);

}
```

) Fehlt: Prüfen ob Sammlung zu Künstler existiert
Sonst: NullPointerException

Aufgabe 2: Exceptions

a) Schreiben Sie die Methode

```
public void setzeInfos(String album, String kuenstler, int tAnzahl)
```

der Klasse *CD* (siehe Blatt 1), die Attribute des *CD*-Objekts mit den Werten aus den Methodenparametern belegt. Die Parameter *album* und *kuenstler* sollen allerdings vor der Zuweisung daraufhin geprüft werden, ob sie leere Zeichenketten enthalten (kein Zeichen oder nur Leerzeichen). Ist einer der beiden Werte ungültig, so soll eine *UngueltigeCDException* geworfen werden. (8 Punkte)

(8)

```
public void setzeInfos(String album, String kuenstler, int tAnzahl) {
    throws UngueltigeCDException {                                    2

    if (album.trim().isEmpty() || kuenstler.trim().isEmpty())         3
        throw new UngueltigeCDException(kuenstler, album);            2

    this.album = album;
    this.kuenstler = kuenstler;
    titelAnzahl = tAnzahl;                                            ^

}
```

b) Schreiben Sie für die Klasse *CDSammlung* (siehe Blatt 2) eine Methode hinzufuegen, die es erlaubt, ein neues CD-Objekt in die Sammlung einzufügen. Sollte das Album des Künstlers bereits in der Sammlung sein, so soll eine *AlbumBereitsVorhandenException* geworfen werden. Denken Sie daran, dass Sie eine neue *HashMap* für die CDs des Künstlers erstellen und in die Sammlung einfügen müssen, falls noch gar keine CD des Künstlers in der Sammlung enthalten ist. (15 Punkte)

(15)

```
public void hinzufuegen(CD cd) throws AlbumBereitsVorhandenException {

    if (sammlung.containsKey(cd.gibKuenstler())) {
        String k = cd.gibKuenstler();
        if (sammlung.get(K).containsKey(cd.gibAlbum())) {
            throw new AlbumBereitsVorhandenException(cd.gibAlbum(), k);
        }
        else
            sammlung.get(k).put(cd.gibAlbum(), cd);
    }
    else {
        String k = cd.gibKuenstler(); 0,5 HashMap<String, CD> neu = new HashMap<String, CD>();
        String a = cd.gibAlbum(); 0,5
        sammlung.put(k, neu HashMap<String, CD>());
        sammlung.get(K).put(a, cd);
    }
}

// schöner wäre sofort cd.gibKuenstler und cd.gibAlbum in Variablen speichern...
```

Aufgabe 3: Datei Ein-/Ausgabe

Schreiben Sie die Methode `laden(File path)` der Klasse *CDSammlung* (siehe Blatt 2). Diese bekommt eine Datei in der serialisierte *CD*-Objekte zu finden sind und soll die *CD*-Objekte aus der Datei einlesen und das Attribut *sammlung* damit befüllen (alle alten Einträge sollen verworfen werden). Die Objekte sind übrigens dann komplett ausgelesen, wenn die entsprechende Methode zum Auslesen eines Objekts aus dem Strom den Wert null liefert. (16 Punkte)

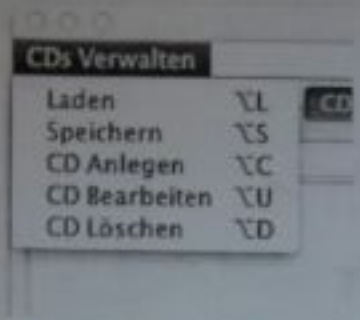
```
public void laden(File path) {                                   (11,5)

  try { 2
    FileInputStream fileStream = new FileInputStream(path); 1
    ObjectInputStream os = new ObjectInputStream(fileStream); 1
    HashMap<String, HashMap<String, CD>> neue = new HashMap<String, HashMap<String,CD>>(); 2
    while (os.readObject() != null) { 0
                  +
      CD cd = (CD)os.readObject(); 2
      neue.hinzufuegen(cd); 0,5
                  +
    }

    Sammlung = neue;
    os.close(); 1
  }
  catch (FileNotFoundException e) {
      e.printStackTrace();
  }                                                              2
  catch (IOException e1) {
      e1.printStackTrace();
  }

}
```

a) Schreiben Sie die Methode menueHinzufuegen(), die dem Fenster der *CDSammlungView* das unten eingeblendete Menü hinzufügt. Es reicht, wenn sie dem Menü nur den Menü-Eintrag *Laden* hinzufügen, die anderen Einträge können Sie weglassen. Als Listener soll eine Instanz der Klasse *MenueListener* gesetzt werden. Die Instanz können sie mit dem Standard-Konstruktor erzeugen. Denken Sie daran, dass alle Items mit demselben Listener-Objekt bestückt werden sollen. (13 Punkte)




```
private void menueHinzufuegen() {

    JMenuBar hauptMenue = new JMenuBar();                          1
    JMenu verwaltenMenu = new JMenu("CDs Verwalten");              1
    JMenuItem ladenItem = new JMenuItem("Laden");                  1
    ladenItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_L, KeyEvent.ALT_MASK
    ladenItem.setActionCommand("laden");
    ladenItem.addActionListener(new MenueListener());              2

    verwaltenMenu.add(ladenItem);                                  1
    hauptMenue.add(verwaltenMenu);                                 1
    fenster.setJMenuBar(hauptMenue);                               2

}
```

b) Ergänzen Sie die Klasse *CDSammlungView* um die innere Klasse *MenueListener*, die auf alle Menü-Aktionen des Fensters reagiert. Behandeln Sie hier nur die Aktion *Laden*, die anderen Aktionen können weggelassen werden. Es soll ein *JFileChooser* geöffnet werden (Pfad-Voreinstellung: */user/music/cds*) um die Datei auszuwählen, aus der die *CD*-Objekte gelesen werden sollen. Die Objekte der Datei sollen in die CD-Sammlung geladen und die Anzeige für die Auflistung des gesamten CD-Bestands soll danach aktualisiert werden. (15 Punkte)

15

```
class MenueListener implements ActionListener {1
    public void actionPerformed(ActionEvent e){1
        if(e.getActionCommand() == "laden"){1
            File start = new File("/user/music/cds");1
            JFileChooser chooser = new JFileChooser(start);1
            int auswahl = chooser.showOpenDialog(fenster);1
            if(auswahl == JFileChooser.APPROVE_OPTION){1
                File file = chooser.getSelectedFile();1
                ~~cdSammlung~~.laden(file);2
                db
            }...
            auflistung.setText(db.toString());2
        }
    }
}
```

```
// Aufbauen des Albumeingabe.
   JPanel albumFeld;
```

JPanel albumFeld = new JPanel(new FlowLayout());
albumEingabe = new JTextField(20);
albumEingabe.getDocument().addDocumentListener(new DocListener());
hatBild = new JCheckBox("Hat es ein Bild", false);
hatBild.addItemListener(new ItemListener());
albumFeld.add(albumEingabe);
albumFeld.add(hatBild);
Border border = BorderFactory.createTitledBorder("Album");
albumFeld.setBorder(border);
eingaben.add(albumFeld);

```
   // Aufbauen der TitelanzahlEingabe.
   JPanel titelFeld;
      ...
   eingaben.add(titelFeld);

   // Aufbauen der Kommentareingabe.
   JPanel kommentarFeld;
      ...
   eingaben.add(kommentarFeld);

// Die Buttons anlegen.
   JPanel buttonLeiste = new JPanel();
     ...

   fensterPanel.add(eingaben, BorderLayout.CENTER);
   fensterPanel.add(buttonLeiste, BorderLayout.SOUTH);
   fenster.add(fensterPanel);
}
```

b) Schreiben Sie die Listener-Klasse für die Buttons. Hier soll nur der Button mit der Aufschrift *Aktualisieren* behandelt werden. Wenn der Button gedrückt wird, soll die Methode `aktualisieren()` aufgerufen werden. (4 Punkte)

3

```
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == aktButton) {
            (e.getActionCommand() == "Aktualisieren")
                aktualisieren();
        }
    }
}
```

c) Schreiben Sie die Methode aktualisieren(). Diese wird vom *ButtonListener* aufgerufen wenn man auf den Button mit der Aufschrift *Aktualisieren* gedrückt hat (siehe Aufgabe 5b). Die im Fenster vorgenommenen Änderungen an der CD sollen in der Sammlung im Attribut *cdSammlung* aktualisiert werden. Verwenden Sie dazu eine geeignete Methode der Klasse *CDSammlung*. Fangen Sie die hierbei möglicherweise auftretenden Exceptions und zeigen Sie deren Nachricht in einem passenden Dialogelement an. (13 Punkte)

(12)

```
private void aktualisieren() {

    CD modifie = new CD();   1
    try {   2

        modifie.setzeInfos(kuenstlerEingabe.getText(), albumEingabe.getText(),
            new Integer( (int) titelanzahlEingabe.getText() ) );   2
                // muss INT sein.
        modifie.setzeBesitzInfos(kaufdatum.getDate(), kommentarEingabe.getText());   2

        cdSammlung.aktualisiereCD(cd, modifie);   2

        + AlbumBereitsVorhandenException
    }
    catch (UngueltigeCDException e1) {   0
        JOptionPane.showMessageDialog(fenster, e1, "Warnung", JOptionPane.WARNING_MESSAGE);
    }   3


    catch (Exception e3) {
        e3.printStackTrace();
    }

}
```