Modul 411 - LB3

Datenstrukturen und Algorithmen

**Allgemeine Informationen**

|  |  |
| --- | --- |
| Zeit | 120 Minuten |
| Punkte maximal | 82 |
| Punkte erreicht |  |
| Note |  |
| Hilfsmittel | Moduljournal, Java-Hilfe |
| ICT - Handlungsziele | 2,4,5,6 |
| Unterschrift Lernender |  |
| Unterschrift Prüfungsleitung |  |
| Art der Prüfung | schriftl.+ praktisch am PC |
| Tools | Java SDK 8 und Eclipse Mars vorinstalliert |
| vorhandene Dateien | music.txt cCodes.ser UnCountryCodeAdmin\_Stundent.java |
| abzugebende Dateien | MusicStore\_*Nachname\_Vorname*.java UnCountryCodeAdmin\_*Nachname\_Vorname*.java |

**Handlungsziele und Aufgaben**

|  |  |  |
| --- | --- | --- |
| Nr | Handlungziel | Aufgaben |
| 2 | Ein Problem analysieren und einen geeigneten Algorithmus zur Lösung mit den Grundelementen Zuweisung, Verzweigung und Schleife entwerfen und mit Prozeduren und Funktionen umsetzen. | 2,3,4 |
| 4 | Ein komplexeres Problem auf kleinere Teilprobleme zurückführen und je nach Problemstellung Iteration oder Rekursion einsetzen. | 2,3,4 |
| 5 | Abstrakte Datentypen, wie Liste, Set, Map etc. und die darauf definierten Operationen kennen und zielgerichtet einsetzen können. | 1,3,4,5 |
| 6 | Datenstrukturen und Algorithmen mit dem Debugger und weiteren Tools untersuchen und dabei speziell die Situation auf Stack und Heap analysieren und in geeigneter Form darstellen. | 2 |

# Aufgabe: Theorie Datentypen (8 Punkte, 14 Minuten)

## Unterschied ArrayList vs. LinkedList (4 Punkte, 7 Minuten)

Beschreiben Sie in Stichpunkten die Unterschiede der Klassen ArrayList und LinkedList aus den java.utils hinsichtlich der Laufzeit beim Zugriff und beim Einfügen von neuen Elementen.

|  |  |  |
| --- | --- | --- |
|  | ArrayList | LinkedList |
| **Einfügen** |  |  |
| **Zugriff** |  |  |

**Bewertungsraster:**

ein Punkt je richtige Aussage

|  |  |  |
| --- | --- | --- |
|  | ArrayList | LinkedList |
| **Einfügen** | langsam, da Array u.U. umkopiert werden muss | kann dynamisch wachsen, einfügen relativ performant |
| **Zugriff** | sehr schnell, Grössen- unabhängig | für ein Element am Ende der Liste muss die gesamte Liste durchschritten werden |

## Abstrakte Datentypen (4 Punkte, 7 Minuten)

Unterstreichen Sie die ungültigen Code-Zeilen und begründen Sie kurz Ihre Entscheidung.

* ArrayList<Person> = new ArrayList<Person>(1000);
* LinkedList<Person> liste = new LinkedList<Person>();
* List<int> luckyNumbers = new LinkedList<int>();
* Map<String, String> myDict = new HashMap<String, String>();
* Falsche Codezeilen sind:
* `ArrayList<Person> = new ArrayList<Person>(1000);`   
  `List<int> luckyNumbers = new LinkedList<int>();`
* **Bewertungsraster:**
* Je zwei Punkte pro richtige Unterstreichung, je zwei Punkte Abzug für fehlerhafte Unterstreichung.

# Aufgabe: Iteration und Rekursion (16 Punkte, 18 Minuten)

## Iteration in Rekursion umwandeln (10 Punkte, 10 Minuten)

Gegeben ist der folgende Code:

public int cumSum(int x){  
 if (x==1)  
 return 1;  
 return x+cumSum(x-1);  
}

1. Welchen Wert gibt die cumSum-Funktion beim Aufruf mit x = 6 zurück? (2 Punkte)

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* **Bewertungskriterien / Lösung:**
* Wert = 21

1. Wandeln Sie die Funktion cumSum in eine Iteration um. Die Funktionalität soll erhalten bleiben.

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* **Bewertungskriterien:**

|  |  |  |
| --- | --- | --- |
| * Kriterium | * max. Punkte | * erreichte Punkte |
| * Schleife vorhanden | * 3 |  |
| * Schleifenabgrenzung | * 3 |  |
| * Rückgabe korrekt | * 2 |  |

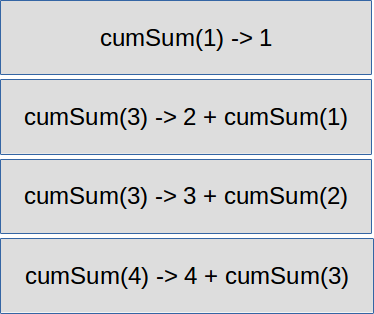
* **Lösungsvorschlag:**
* public int cumSum(int x){  
   int res = 0;  
   for (int i = 0; i <= x; i++)  
   res += i;  
   return res;  
  }

## Programmstack (6 Punkte, 8 Minuten)

1. Was versteht man unter dem Programmstack im Arbeitsspeicher? (2 Punkte)

* **Lösungsvorschlag:**
* etwa so: *für jeden Funktionsaufruf werden Rücksprungadresse und lokale Variablen auf den Stack gelegt, und beim Beenden des Aufrufs wieder frei gegeben* oder ähnlich

1. Beschreiben oder skizzieren Sie den Ablauf auf dem Programmstack, wenn obige Funktion mit x=4 aufgerufen wird. (4 Punkte)

* **Lösungsvorschlag:**
* 
* grafischer Lösungsvorschlag

# Aufgabe: Klassen-Design und Dateihandling (30 Punkte, 40 Minuten)

Sie erstellen ein Programm zum Verwalten und Anzeigen von Musikdaten. Lesen Sie zunächst alle Teilaufgaben, um das Gesamtkonzept zu verstehen. Dann machen Sie sich an die Arbeit.

Achten Sie bei allen Teilaufgaben auf Java-konforme Schreibweise und Javadoc-konforme Dokumentation der Funktionen.

1. Analysieren Sie zunächst den Aufbau der Datei *music.txt*. Erstellen Sie in Ihrem Workspace ein neues Projekt M411\_LB3 und ein src-Package lb3. In diesem Package erstellen Sie eine neue *public* Java-Klasse mit Namen MusicStore. In dieser Klasse erstellen Sie eine Klasse MusicEntry. MusicEntry soll sinnvolle Klassenvariablen für alle Spalten der *music.txt* erhalten. (6 Punkte)

* **Bewertungsraster:**
  + Klassennamen korrekt 2 Punkte
  + Klassenvariablen (artistName, albumName, trackName) 3 Punkte
  + Code-Einrückungen und Javadoc ok 1 Punkt

1. Erstellen Sie einen Konstruktor für MusicEntry, der alle Klassenvariablen als Parameter übernimmt und entsprechend schreibt. (4 Punkte)

* **Bewertungsraster:**
  + ctor vorhanden: 1 Punkt
  + Parameterliste korrekt 3 Punkte
  + Werte richtig übertragen 1 Punkt
  + Code-Einrückungen und Javadoc ok 1 Punkt

1. Erstellen Sie in der Klasse MusicStore eine geeignete Klassenvariable, um die Werte aus der Textdatei aufzunehmen. (4 Punkte)

* **Bewertungsraster:**
  + ArrayList oder LinkedList als Datentyp für Klassenvariable vorhanden 3 Punkte
  + Code-Einrückungen und Javadoc ok 1 Punkt

1. Erstellen Sie eine Klassenfunktion für MusicStore, welche die Daten aus der Textdatei einliest, parst und in Ihrer Datenstruktur abspeichert. (6 Punkte)

* **Bewertungsraster:**
  + BufferedReader oder Scanner wird korrekt zum Datei öffnen eingesetzt 0.5 Punkt
  + Zeilenweise lesen und String.split() einsetzen 0.5 Punkt
  + Werte werden korrekt in MusicEntry Konstruktor übergeben 3 Punkte
  + neue MusicEntry Objekte werden in Liste abgelegt
  + Code-Einrückungen und Javadoc ok 1 Punkt
* **Musterlösung:**
* /\*\*\*  
   \* reads fileName line by line, parses and creates appropriate MusicEntry instances and stores them in ArrayList   
   \* @param fileName  
   \*/  
  public void readDataFromFile(String fileName){  
   BufferedReader bfr = null;  
   try{  
   bfr = new BufferedReader(new FileReader(fileName));  
   while (bfr.ready()){  
   String line = bfr.readLine();  
   String[] parts = line.split("\t");  
   content.add(new MusicEntry(parts[0], parts[1],parts[2]));  
   }  
   }  
   catch (IOException ex){  
   System.out.println(ex.toString());  
   try{  
   bfr.close();  
   } catch (Exception exi){}  
   }  
    
  }

1. Erstellen Sie eine Klassenfunktion List<MusicEntry> findByArtist(String artistName), die alle Einträge mit dem angegebenen artistName zurück gibt. Nutzen Sie dazu die entsprechende Funktionen aus java.utils mit Lambda-Funktionen. (5 Punkte)

* **Bewertungsraster:**
  + Funktionskopf korrekt 1 Punkt
  + Rückgabe korrekt: return List 1 Punkt
  + filter-Funktion korrekt eingesetzt: 2 Punkte oder selbst richtig gebastelt: 1 Punkt
  + Code-Einrückungen und Javadoc ok 1 Punkt
* **Musterlösung:**
* /\*\*  
   \* liefert alle Einträge mit dem angegebenen artistName  
   \*/  
   public Set<String> findAlbumByArtist(String artistName){  
   //return this.content.stream().filter(entry->entry.artist.toLowerCase().  
   equals(artist.toLowerCase())).collect(Collectors.toList());  
   Set<String> result = new HashSet<String>();  
   this.content.forEach(p->{  
   if (p.artist.toLowerCase().equals(artistName.toLowerCase()))  
   result.add(p.album);  
   });  
   return result;  
  }

1. Erstellen Sie einen geeigneten Testrahmen in der main-Funktion. Dieser enthält Anweisungen
   * zum Anlegen des MusicStore
   * zum Einlesen der Daten
   * zur Suche und Ausgabe nach Band

* (5 Punkte)
* Bewertungsraster:
  + MusicStore instantiiert 1 Punkt
  + Datei wird eingelesen 1 Punkt
  + Suche wird ausgeführt und Ergebnisse in main ausgegeben 2 Punkte
  + Code-Einrückungen und Javadoc ok 1 Punkt

# Aufgabe: Sortier- und Filterfunktionen implementieren (20 Punkte, 30 Minuten)

Sie haben den Auftrag, einige Zusatzfunktionen für die gegebene Klasse UnCountryCodeAdmin zu implementieren. Die Klasse importieren Sie zunächst in Ihr Eclipse-Projekt. Nutzen Sie dazu die Eclipse-Importfunktion. Dann passen in der main()-Funktion den Pfad zur cCodes.cer an.

Die Klasse dient der Verwaltung und Anzeige der Länder der Vereinten Nationen. Dazu wurde bereits eine Datenklasse Country mit folgenden Attributen angelegt:

class Country{  
 int unCode;  
 String name;  
 String continent;  
 String threeLetterCode;  
 /\* ... \*/  
}

Bisher ist das Einlesen der Daten und Abspeichern beim Beenden implementiert.

Achten Sie bei allen folgenden Teilaufgaben auf Java-konforme Schreibweise und Javadoc-konforme Dokumentation der Funktionen. Nur main() und printAllCountries() sollen System.out.println(..) Aufrufe enthalten. An der Klasse country ändern Sie bitte nichts.

Erstellen Sie innerhalb UnCountryCodeAdmin die folgenden Funktionen:

1. public void printAllCountries() gibt country und continent aller Datensätze in countries absteigend sortiert nach continent auf dem Bildschirm aus. Prüfen Sie die Funktion mit einem entsprechenden Aufruf in der main()-Funktion. (7 Punkte)

* **Bewertungsraster:**
  + Funktionskopf korrekt 1 Punkt
  + Sortierfunktion korrekt umgesetzt 2 Punkte
  + Aufruf im main() vorhanden 1 Punkt
  + Funktionsweise korrekt 2 Punkte
  + Code-Einrückungen und Javadoc ok 1 Punkt
* **Musterlösung:**
* /\*\*  
   \* sort all entries by continent and print country and continent name  
   \*/  
   public void printAllCountries(){  
   countries.sort((c1,c2)->c1.continent.compareTo(c2.continent));  
   for (Country c : countries)  
   System.out.println(c.name+", "+c.continent);  
  }

1. public Country getCountryById(int unCode) gibt den gesamten Country Datensatz zurück, der zur angefragten unCode passt. Nutzen Sie wenn möglich Lambda-Funktionen. Testen Sie die Funktionalität mit einem entsprechenden Aufruf in der main()-Funktion. (6 Punkte)

* **Bewertungsraster:**
  + Funktionskopf korrekt 1 Punkt
  + Aufruf im main() vorhanden 1 Punkt
  + Funktionsweise korrekt 2 Punkte
  + Lambda Funktionen verwendet 1 Punkt, sonst 0
  + Code-Einrückungen und Javadoc ok 1 Punkt
* **Musterlösung:**
* /\*\*  
   \* gibt den gesamten `Country` Datensatz zurück, der zur angefragten `unCode` passt.  
   \* @param unCode  
   \* @return  
  \*/  
  public Country getCountryById(int unCode){  
   Country result = null;  
   result = countries.stream().filter(country->country.unCode==unCode).collect(Collectors.toList()).get(0);  
    
   return result;  
  }

1. public Map<String, Integer> getNumberCountriesPerContinent() liefert für jeden Kontinent die Anzahl der Länder zurück. Nutzen Sie wenn möglich Lambda-Funktionen. Testen Sie die Funktionalität mit einem entsprechenden Aufruf in der main()-Funktion. (7 Punkte)

* **Bewertungsraster:**
  + Funktionskopf korrekt 1 Punkt
  + Aufruf mit sinnvoller Ausgabe im main() vorhanden 2 Punkt
  + Funktionsweise korrekt 2 Punkte
  + Lambda Funktionen verwendet 1 Punkt
  + Code-Einrückungen und Javadoc ok 1 Punkt
* **Musterlösung:**
* /\*\*  
   \* creates a map of continents with number of countries based on loaded data  
   \* @return map of continents with number of countries  
  \*/  
  public Map<String, Integer> getNumberCountriesPerContinent(){  
   Map<String, Integer> result = new HashMap<String, Integer>();  
   countries.forEach(country->{  
   if ((result.get(country.continent))!=null){  
   Integer value = result.get(country.continent)+1;  
   result.replace(country.continent, value);  
   }  
   else   
   result.put(country.continent, 1);  
   });  
    
   return result;  
  }

# Datenaustausch mit Web-Services (8 Punkte, 5 Minuten)

Beantworten Sie die folgenden Fragen in Stichpunkten, bzw. kreuzen Sie die richtige Auswahl an:

1. XML und JSON sind gebräuchliche Datenformate von Web-Services. Welche Art von Datenstruktur repräsentieren Sie?

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Welche Java-Klasse wird benötigt, um eine Verbindung zu einem Web-Service aufzubauen?

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. In Java erhalten Sie die Antwort des Web-Services auf Anfragen mit welchem Datentyp:
   1. String
   2. InputStream
   3. JSON-Object
2. Was versteht man unter Traversieren?

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* **Lösung und Bewertungskriterien:**
* 1. Baumstruktur 2 Punkte  
  2. URLConnection 2 Punkte  
  3. InputStream 2 Punkte  
  4. durchschreiten einer Baumstruktur 2 Punkte