Prüfungsleistung WWI218 Programmieren II - Programmentwurf mit Dokumentation

Allgemeine Informationen

Diese Prüfungsleistung bildet gemeinsam mit dem durchgeführten Kurztest die Gesamtnote der Portfolioprüfung für die Vorlesung Programmieren II des Kurses WWI218 der DHBW Ravensburg.
Die Teilleistung untergliedert sich in einen Programmentwurf und die dazu zu erstellende Dokumentation. Der Programmentwurf geht mit insgesamt **60%** in die Gesamtnote ein, die Dokumentation mit insgesamt **20%**. Zusammen mit dem 20% aus dem Kurztest ergibt sich so die Gesamtnote.
Die Durchführung der Prüfungsleistung erfolgt in Gruppen zu **3 oder maximal 4 Personen**. Die Gruppenaufteilung ist spätestens zum **26.04.2019** festzulegen und wird in der Vorlesung besprochen.
In der Regel wird davon ausgegangen, dass alle Gruppenmitglieder zu gleichen Anteilen an der Bearbeitung der Aufgabe beteiligt waren. **Daher wird auch für alle Gruppenmitglieder die gleiche Bewertung festgesetzt**. Sollte gewünscht sein, dass die Bewertung für jedes Gruppenmitglied differenziert erfolgt, so muss dies in der Abgabe explizit vermerkt sein. Außerdem muss aus den abgegebenen Ergebnissen eindeutig erkennbar sein, welche Teile durch welches Gruppenmitglied bearbeitet wurden.
Die Abgabefrist für den Programmentwurf ist der **15.07.2019 23:59:59 MESZ**. Zur Abgabe müssen alle zur Bewertung benötigten Dateien bereitgestellt werden. Dies ebinhaltet:

- JAR Datei für den Programmentwurf
- Alle Quelldateien
- Vollständige Dokumentation im PDF bzw. HTML(Für Javadocs) Format

Die Abgabe **muss per Mail** an lukas.abelt@airbus.com erfolgen. Zur Abgabe können die benötigten Dateien **entweder als ZIP Datei** komprimiert und abgegeben werden. In diesem Fall bitte jedoch nicht die ZIP Datei als Anhang zur Mail hinzufügen, sondern auf einem externen Fileshare(Dropbox, Google Drive, Nextcloud etc.) ablegen und den Link zum Download bereitstellen (Anhänge im ZIP Format werden durch den Mailfilter gelöscht). Sofern während der Entwicklung mit einem Git Repository gearbeitet wurde, kann die Abgabe auch erfolgen, indem ich zum entsprechenden Repository hinzugefügt werde (GitLab bzw. GitHub Accounts könnt ihr den Folien entnehmen). **Bitte in diesem Fall auch unbedingt trotzdem die Abgabe per Mail bestätigen!** Bei der Abgabe als Git Repository wird zur Bewertung der letzte Commit vor Maileingang genutzt.
Die Mail zur Abgabe **muss die folgenden Informationen enthalten**:

- Namen aller Gruppenmitglieder
- Gewählte Aufgabenstellung
- Link zum ZIP Download bzw. Link zum GitHub/GitLab Repository.

Pro Gruppe reicht die Bestätigungsmail durch ein Mitglied. Bitte nehmt die Gruppenmitglieder mit in den CC.

Zur Bearbeitung des Programmentwurfs stehen zwei Aufgaben zur Auswahl. Von diesen muss **eine Aufgabe vollständig bearbeitet werden**. Ein Bearbeiten beider Aufgaben ist **nicht** nötig und bringt **keine** zusätzlichen Punkte in der Bewertung. Die Implementierung der Aufgabe sollte in Java erfolgen. Sollte eine Gruppe die Umsetzung in einer anderen Sprache bevorzugen ist dies vorher mit mir abzuklären.
Die Dokumentation dient dem Zweck, den Entwicklungsprozess nachzuverfolgen. In dieser soll festgehalten werden, warum bestimmte Designentscheidungen getroffen wurden, welche Alternativen ggf. zur Auswahl standen und welche Vor- und Nachteile diese geboten hätten. Außerdem sollen Schwierigkeiten und Herausforderungen, die während der Entwicklung auftraten, mit in der Dokumentation aufgeführt werden. Auch können hier Beispiele zur Benutzung mit aufgenommen werden, sowie entwickelte Algorithmen erläutert werden. Die Dokumentation sollte pro Gruppe 10 Seiten nicht überschreiten. Diese 10 Seiten beinhalten nicht ggf. automatisch über Javadoc (o.Ä.) generierte Dokumentation.

Aufgabenvorschlag 1 - Listenbibliothek

In der Vorlesung wurden diverse Listenstrukturen besprochen. Ziel dieses Aufgabenvorschlags ist es, dass Ihr einige dieser Listenstrukturen selbst implementiert, sodass als Produkt eine fertig nutzbare Bibliothek bereitsteht, die die Grundlegenden Listenstrukturen zur Verfügung stellt(Ähnlich zum Collections Framework). Die genau zu implementierenden Listenstrukturen mit den entsprechenden Anforderungen werden im folgenden erläutert.

Allgemeine Rahmenbedingungen

Für die Umsetzung der Aufgabe darf keine der Klassen oder Interfaces aus dem `Collections` Framework der Java Standardbibliothek (Oder ein Äquivalent einer anderen Sprache) verwendet werden. Auch dürfen keine Third-Party Bibliotheken verwendet werden, die eine ähnliche Funktionalität abbilden. Die einzige Ausnahme bilden das `Iterator` und `Iterable` Interface, die zur Bearbeitung der Zusatzaufgaben verwendet werden können. Natürlich kann die Struktur des `Collection` Interfaces herangezogen werden.
Eure entwickelte Bibliothek sollte die folgenden Listenstrukturen abbilden:

- Linked List
- Double Linked List
- Queue
- Stack
- Set
- Suchbaum

Die Listenstrukturen sollen für die Speicherung von Objekten jeder beliebigen Klasse verwendet werden können. Alle Listenstrukturen sollten die folgenden Methoden implementieren(ggf. mit den speziellen Eigenschaften für diese Struktur):

- Eine Funktion um zu überprüfen ob die Liste leer ist
- Eine Funktion um die Anzahl der Elemente in der Liste zu ermitteln
- Eine Funktion um eine neues Element hinzuzufügen
- Eine Funktion die alle Elemente einer anderen Listenstruktur zu der aktuellen hinzufügt
- Eine Funktion um alle Elemente zu entfernen
- Eine Funktion um zu überprüfen, ob ein bestimmtes Objekt Teil der Liste ist
- Eine Funktion, der eine Listenstruktur übergeben wird. Die Funktion prüft, ob jedes Element aus der übergebenen Struktur auch Teil dieser Struktur ist
- Eine Funktion um ein bestimmtes Objekt zu entfernen
- Eine Funktion um den Inhalt der Datenstruktur als Array zurückzugeben

Außerdem soll es möglich sein, die Inhalte der Listenstrukturen (Mit Ausnahme des Suchbaums) zu sortieren. Entwerfen sie hierfür eine geeignete Klasse bzw. Interface. Erläutern Sie in der Dokumentation ihre Entscheidung, warum Sie zur Umsetzung eine Klasse oder Interface gewählt haben. Der Nutzer soll in jedem Fall verschiedene Sortieralgorithmen zur Auswahl haben. Zu implementieren sind:

- BubbleSort
- QuickSort
- Ein Algorithmus eurer Wahl

Zum Sortieren ist die Nutzung des `Comparator` bzw. `Comparable` Interfaces notwendig.
Die einzelnen Besonderheiten der Listen werden folgend definiert.

Linked List

Die Linked List sollte entsprechend der in der Vorlesung vorgestellten Struktur implementiert werden. Zusätzlich zu den zuvor genannten Grundfunktionen soll die Linked List noch die folgenden Funktionen enthalten:

- Methode um das vorderste Element der Linked List zu lesen
- Einfügen eines Elements an einer gegebenen Position
- Entfernen eines Elements an einer gegebenen Position
- Manipulieren eines Elements an einer gegebenen Position
- Finden der Position eines bestimmten Objects (Sofern dies in der Liste vorhanden ist)

Hierbei ist besonders darauf zu achten und dokumentieren, wie sich die Funktionen verhalten wenn ungültige Werte angegeben werden. Hierfür können Exceptions genutzt werden. Da diese jedoch nicht im Detail besprochen wurden, führt das nichtverwenden von Exceptions jedoch nicht zu Punktabzug.

Linked List

Die Double Linked List sollte entsprechend der in der Vorlesung vorgestellten Struktur implementiert werden. Zusätzlich zu den zuvor genannten Grundfunktionen soll die Double Linked List noch die folgenden Funktionen enthalten:

- Funktion um das erste Element der Double Linked List zu lesen
- Funktion um das letzte Element der Double Linked List zu lesen
- Einfügen eines Elements an einer gegebenen Position
- Entfernen eines Elements an einer gegebenen Position
- Manipulieren eines Elements an einer gegebenen Position
- Finden der Position eines bestimmten Objects (Sofern dies in der Liste vorhanden ist)

Hierbei ist besonders darauf zu achten und dokumentieren, wie sich die Funktionen verhalten wenn ungültige Werte angegeben werden. Hierfür können Exceptions genutzt werden. Da diese jedoch nicht im Detail besprochen wurden, führt das nichtverwenden von Exceptions jedoch nicht zu Punktabzug.

Queue

Die Queue soll entsprechend ihrer in der Vorlesung besprochenen Eigenschaften implementiert werden. Außerdem sollen zusätzlich folgende Methoden bereitgestellt werden:

- Eine Methode, die das erste Element der Queue zurückgibt und aus der Queue entfernt
- Eine Methode, die das erste Element der Queue zurückgibt ohne es zu entfernen

Stack

Der Stack soll entsprechend ihrer in der Vorlesung besprochenen Eigenschaften implementiert werden. Außerdem sollen zusätzlich folgende Methoden bereitgestellt werden:

- Eine Methode, die das letzte Element des Stacks zurückgibt und aus dem Stack entfernt
- Eine Methode, die das letzte Element des Stacks zurückgibt ohne es zu entfernen

Set

Bei einem Set handelt es sich um eine Menge wie sie in der Mengenlehre definiert wird. Es handelt sich somit um eine Listenstruktur, in der jedes Element genau einmal existieren darf. Zusätzlich zu den Standardfunktionen sollen hier die handelsüblichen Mengenoperationen implementiert werden:

- Bestimmen der Schnittmenge
- Bestimmen der Vereinigungsmenge
- Bestimmen der Differenzmenge
- Bestimmen der Symmetrischen Differenz

Suchbaum

Der Suchbaum ist entsprechend der in der Vorlesung besprochenen Eigenschaften zu implementieren. Der Suchbaum verfügt lediglich über die Standardfunktionen.

Dokumentation

In der Dokumentation sollte eure umgesetzte Klassenstruktur inklusive aller implementierten Klassen und Interfaces ersichtlich sein. Erläutert, warum ihr euch für die gewählte Struktur entschieden habt. Geht auf die Schwierigkeiten bei der Umsetzung ein.
Besonderer Augenmerk steht bei dieser Aufgabe auf der Dokumentation des Codes. Ergebnis sollte neben der Dokumentation im PDF Format auch eine umfangreiche API Dokumentation sein, in der die Funktionen der Klassen und Methoden verfügbar sind. Nutzt zur Generierung dieser API Dokumentation die Javadoc Notation!

Zusätzliche Punkte

Durch verschiedene Erweiterungen können ggf. zuvor „verlorene“ Punkte ausgeglichen werden. Hier bitte der Hinweis, dass diese zusätzlichen Aufgaben keineswegs im Fokus der Entwicklung stehen sollten und, wenn überhaupt, erst bearbeitet werden sollten, wenn die vorherigen Aufgaben nach bestem Wissen und Gewissen bearbeitet wurden.
Zusätzliche Punkte können zum Beispiel erreicht werden durch:

- Bereitstellen einer Demo Anwendung (In der Kommandozeile) die die Funktionen der Bibliothek demonstriert.
- Implementierung des `Iterable` bzw. `Iterator` Interfaces in die Listenstruktur
- Implementierung einer Klasse, die einen AVL-Baum darstellt

Sonst könnt ihr natürlich weitere Funktionen einbauen die euch, sofern ihr eine logische Begründung habt, warum diese Teil der Bibliothek sein sollten, zusätzliche Punkte verschaffen können. Bitte geht auf die zusätzlichen Features noch einmal separat in der Dokumentation ein.

Aufgabenvorschlag 2 - Set Kartenspiel

Allgemeine Informationen

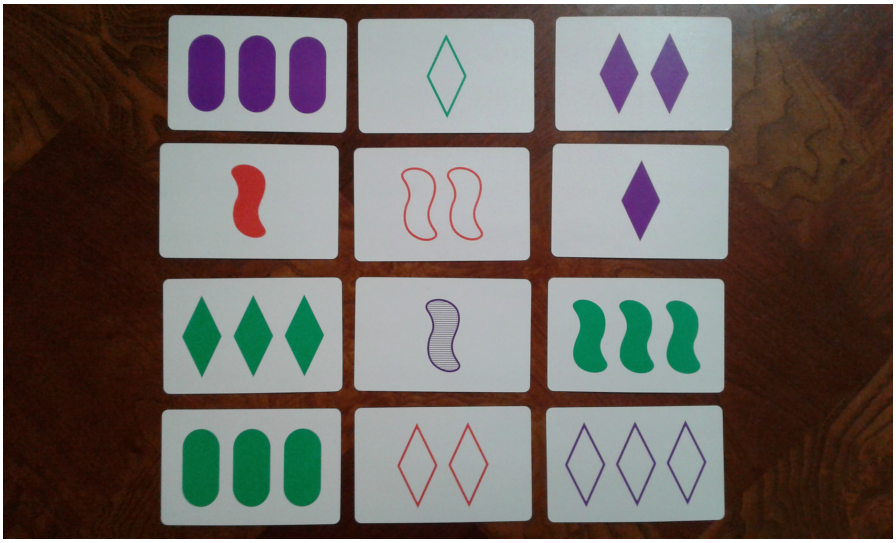
Im zweiten Aufgabenvorschlag geht es um die Implementierung des Kartenspiels „Set“ in einer geeigneten GUI.
„Set“ ist ein Kartenspiel, bei dem das Deck aus insgesamt 81 Karten besteht. Jeder der Karten unterscheidet sich in mindestens einem von vier Eigenschaften. Jede Eigenschaft hat genau drei Ausprägungsgrade. Die vier Eigenschaften sind:

- Das gezeigte Symbol (Bspw.: Kreis, Diamant, Rechteck)[1]
- Die Anzahl der Symbole (1, 2, 3)
- Die Farbe der Symbole (Bspw.: Rot, Grün, Blau)[2]
- Den Ausfüllungsgrad der Symbole (Nur Umrisse, Teilweise ausgefüllt/gestreift, komplett gefüllt)

Zu Beginn des Spiels werden nun zwölf Karten offen aufgedeckt. Ziel ist es nun, aus diesen zwölf Karten ein „Set“ aus drei Karten zu finden. Drei Karten bilden genau dann ein Set, wenn für jede Eigenschaft der Drei Karten gilt: itemize

Die Eigenschaft hat auf allen drei Karten den gleichen Ausprägungsgrad

Die Eiigenschaft hat auf allen drei Karten einen anderen Ausprägungsgrad (=Alle Ausprägungsgrade sind vorhanden)
Zusammenfassen lässt sich das ganze auch unter der Bedingung: Wenn für eine Eigenschaft gilt „zwei von _ _ _ und eins von _ _ _“, so handelt es sich nicht um ein Set.
Im folgenden Bild, bilden beispielsweise die drei Karten die jeweils 3 grüne, voll ausgefüllte, Symbole zeigen ein Set.



[1] Die Symbole können natürlich frei gewählt werden
[2] Die Farbe kann natürlich frei gewählt werden

Umfang der Aufgabenstellung

Aufgabe ist es, eine GUI Anwendung zu erstellen, die das spielen von SET ermöglicht. Hierbei sollen immer die Variante gespielt werden, in der zwölf Karten offen liegen. Die GUI Umsetzung soll für einen einzelnen Spieler ausgelegt sein.
Die GUI Umsetzung sollte mindestens die folgenden Funktionen beinhalten:

- Jedes Spiel startet mit einem zufällig gemischten Deck
- Zu Beginn werden zwölf Karten aufgedeckt
- Anzeigen der offenen Karten entsprechend ihrer Eigenschaften
- Visualisierung des restlichen Decks (Mit Anzeige der im Deck verbleibenden Karten)
- Visualisierung eines Ablagestapels (Mit Anzeige der im Ablagestapel liegenden Karten
- Anzeigen der bisher durch den Spieler gefundener Sets
- Anzeige, wie viele Sets aus den aktuell offenen Karten möglich sind
- Unterstützung von zwei Spielmodi:
  - Normaler Spielmodus: Das Spiel läuft bis keine Karten mehr im Deck vorhanden sind und kein Set mehr vorhanden ist
  - Zeitmodus: Der Spieler hat nur eine begrenzte Zeit um so viele Sets wie möglich zu finden
- Bei klicken auf eine Karte wird diese „ausgewählt“ (mit passender Visualisierung). Bei erneutem klicken wird diese „abgewählt“
- Bei drei ausgewählten Karten wird geprüft, ob diese drei Karten ein SET ergeben
  - Wenn Sie ein Set ergeben, wandern die drei Karten auf den Ablagestapel und vom Deck werden drei neue Karten aufgedeckt
  - Wenn Sie kein Set ergeben, werden die Karten abgewählt. Die ausgewählten Karten werden nicht abgelegt.
- Anzeige der bisher verstrichenen Zeit bzw. verbleibenden Zeit (Je nach Spielmodus)
- Sollten die offenen Karten kein SET ergeben, so werden die aktuell aufgedeckten Karten auf den Ablagestapel gelegt und zwölf neue Karten aufgedeckt

Dokumentation

Aus der Dokumentation sollte ersichtlich sein, welche Datenstrukturen und Klassen ihr nutzt bzw. selbst entworfen habt um die Umsetzung des Spiels zu realisieren. Geht insbesondere auf die Visualiserung der einzelnen Karten in der GUI ein. Welche Möglichkeiten habt ihr erwogen und was waren schlussendlich die ausschlaggebenden Faktoren für die Auswahl?
Erläutert, wie eure implementierten Algorithmen, die prüfen ob drei Karten ein Set bilden und der prüft, wie viele Sets in den offenen Karten vorhanden sind.
Begründet, welche GUI Elemente und Klassen ihr für eure Umsetzung genutzt habt.

Zusätzliche Punkte

Durch verschiedene Erweiterungen können ggf. zuvor „verlorene“ Punkte ausgeglichen werden. Hier bitte der Hinweis, dass diese zusätzlichen Aufgaben keineswegs im Fokus der Entwicklung stehen sollten und, wenn überhaupt, erst bearbeitet werden sollten, wenn die vorherigen Aufgaben nach bestem Wissen und Gewissen bearbeitet wurden.
Zusätzliche Punkte könnte man zum Beispiel erreich durch:

- Änderbare Farben der Symbole aus der GUI heraus
- Einbinden von Animationen in das Spiel
- Besonders kreative Umsetzung der GUI

Neben den hier genannten Möglichkeiten gibt es natürlich noch etliche weitere Möglichkeiten, wie sich das Spiel erweitern ließe. Hier sind eurer Kreativität keine Grenzen gesetzt. Falls ihr zusätzliche Funktionen einbaut, so geht auf diese jedoch explizit noch einmal in der Dokumentation ein