Arbeitsblatt: Collections mit Arrays

Eine einfache Art, Collections zu implementieren, ist die Elemente in einem Array abzulegen. Enthält eine Collection *n* Elemente, sind diese im Array unter den Indices 0 bis *n*-1 abgelegt. Speichert man zusätzlich den aktuellen Wert für *n*, kann man die Collection-Grundoperationen *add, contains* und *remove* leicht implementieren. Das sollen Sie nun tun.

Wir haben gesehen, dass es vier Möglichkeiten gibt, eine Collection in einem Array zu implementieren. Für die beiden Definitionen *Set* und *Bag* je eine Version mit sortierter Speicherung und eine ohne. Also:

		Sortierung	
		Sortiert	**Nicht sortiert**
Semantik	**Set-Semantik**	*SortedSet*	*UnsortedSet*
	Bag-Semantik	*SortedBag*	*UnsortedBag*

Ihr Auftrag ist es nun, die in der Tabelle gekennzeichnete Variante zu implementieren. Als Rahmen steht Ihnen dazu auf dem Dateiserver das Programmier-Projekt *ch.fhnw.algd2.arraycollections* zur Verfügung.

Das *java.util.Collection* Interface definiert das erwartete Verhalten der Datenstrukturen mit Vor- und Nachbedingungen, lässt aber trotzdem Raum für verschiedene Implementierungen. Beispielsweise darf die Speicherkapazität der Collection begrenzt sein. Ein allfälliger „Überlauf" muss dann jedoch durch eine bestimmte Exception angezeigt werden. Es ist wichtig, sich an die Vorgaben zu halten, da andere Komponenten das im Interface definierte Verhalten erwarten.

Um das zu überprüfen, stehen Ihnen Unit-Tests zur Verfügung.

Hinweise zur Implementierung:

- Verwenden Sie *Generics*. So vermeiden Sie bei der *add*-Methode eine Class-Cast Exception.
- *null*-Werte sollen in Ihrer Collection *nicht* erlaubt sein.
- Es sind Unit-Tests vorhanden. Nicht immer werden optionale Exceptions geprüft. Ändern Sie die Unit-Tests ruhig ab, wenn Sie optionale Exceptions in Ihrer Implementation verwenden.
- In den Klassen sind jeweils *main*-Methoden vorhanden, die zeigen, wie Sie den Inhalt Ihrer Collection auf der Konsole ausgeben können.

Gehen Sie wie folgt vor:

1. Bilden Sie Gruppen, die zusammen die gleiche Variante bearbeiten.
2. Zeigen Sie die Verhaltensweise Ihrer Datenstruktur beim Einfügen der folgenden Zahlen-Sequenz: 12, 5, 28, 47, 12, 8

3. Importieren Sie das Projekt *ch.fhnw.algd2.arraycollections* in Ihre IDE (vgl. separate Anleitung).
4. Implementieren Sie die Methoden für Ihre Variante der Collection. Das Klassentemplate steht Ihnen bereits zur Verfügung.
5. Sie brauchen keine Vergrösserungsstrategien zu implementieren. Wenn es keinen Platz mehr hat, können eben keine Elemente mehr eingefügt werden. Testen Sie ihre Implementation mit den *Unit Tests*.

   Hinweis: Für eine binäre Suche kann die Methode *Arrays.binarySearch(…)* oder eine eigene Implementation verwendet werden. Begründen Sie Ihre Wahl.
6. Ergänzen Sie die Tabelle am Ende der Unterrichtsunterlagen für Ihre Implementierung.

Arbeitsblatt: Collections mit Arrays

Eine einfache Art, Collections zu implementieren, ist die Elemente in einem Array abzulegen. Enthält eine Collection *n* Elemente, sind diese im Array unter den Indices 0 bis *n*-1 abgelegt. Speichert man zusätzlich den aktuellen Wert für *n*, kann man die Collection-Grundoperationen *add, contains* und *remove* leicht implementieren. Das sollen Sie nun tun.

Wir haben gesehen, dass es vier Möglichkeiten gibt, eine Collection in einem Array zu implementieren. Für die beiden Definitionen *Set* und *Bag* je eine Version mit sortierter Speicherung und eine ohne. Also:

		Sortierung	
		Sortiert	**Nicht sortiert**
Semantik	**Set-Semantik**	*SortedSet*	*UnsortedSet*
	Bag-Semantik	*SortedBag*	*UnsortedBag*

Ihr Auftrag ist es nun, die in der Tabelle gekennzeichnete Variante zu implementieren. Als Rahmen steht Ihnen dazu auf dem Dateiserver das Programmier-Projekt *ch.fhnw.algd2.arraycollections* zur Verfügung.

Das *java.util.Collection* Interface definiert das erwartete Verhalten der Datenstrukturen mit Vor- und Nachbedingungen, lässt aber trotzdem Raum für verschiedene Implementierungen. Beispielsweise darf die Speicherkapazität der Collection begrenzt sein. Ein allfälliger „Überlauf" muss dann jedoch durch eine bestimmte Exception angezeigt werden. Es ist wichtig, sich an die Vorgaben zu halten, da andere Komponenten das im Interface definierte Verhalten erwarten.

Um das zu überprüfen, stehen Ihnen Unit-Tests zur Verfügung.

Hinweise zur Implementierung:

- Verwenden Sie *Generics*. So vermeiden Sie bei der *add*-Methode eine Class-Cast Exception.
- *null*-Werte sollen in Ihrer Collection *nicht* erlaubt sein.
- Es sind Unit-Tests vorhanden. Nicht immer werden optionale Exceptions geprüft. Ändern Sie die Unit-Tests ruhig ab, wenn Sie optionale Exceptions in Ihrer Implementation verwenden.
- In den Klassen sind jeweils *main*-Methoden vorhanden, die zeigen, wie Sie den Inhalt Ihrer Collection auf der Konsole ausgeben können.

Gehen Sie wie folgt vor:

1. Bilden Sie Gruppen, die zusammen die gleiche Variante bearbeiten.
2. Zeigen Sie die Verhaltensweise Ihrer Datenstruktur beim Einfügen der folgenden Zahlen-Sequenz: 12, 5, 28, 47, 12, 8

3. Importieren Sie das Projekt *ch.fhnw.algd2.arraycollections* in Ihre IDE (vgl. separate Anleitung).
4. Implementieren Sie die Methoden für Ihre Variante der Collection. Das Klassentemplate steht Ihnen bereits zur Verfügung.
5. Sie brauchen keine Vergrösserungsstrategien zu implementieren. Wenn es keinen Platz mehr hat, können eben keine Elemente mehr eingefügt werden. Testen Sie ihre Implementation mit den *Unit Tests*.

   Hinweis: Für eine binäre Suche kann die Methode *Arrays.binarySearch(…)* oder eine eigene Implementation verwendet werden. Begründen Sie Ihre Wahl.
6. Ergänzen Sie die Tabelle am Ende der Unterrichtsunterlagen für Ihre Implementierung.

Arbeitsblatt: Collections mit Arrays

Eine einfache Art, Collections zu implementieren, ist die Elemente in einem Array abzulegen. Enthält eine Collection *n* Elemente, sind diese im Array unter den Indices 0 bis *n*-1 abgelegt. Speichert man zusätzlich den aktuellen Wert für *n*, kann man die Collection-Grundoperationen *add, contains* und *remove* leicht implementieren. Das sollen Sie nun tun.

Wir haben gesehen, dass es vier Möglichkeiten gibt, eine Collection in einem Array zu implementieren. Für die beiden Definitionen *Set* und *Bag* je eine Version mit sortierter Speicherung und eine ohne. Also:

		Sortierung	
		Sortiert	**Nicht sortiert**
Semantik	**Set-Semantik**	*SortedSet*	*UnsortedSet*
	Bag-Semantik	*SortedBag*	*UnsortedBag*

Ihr Auftrag ist es nun, die in der Tabelle gekennzeichnete Variante zu implementieren. Als Rahmen steht Ihnen dazu auf dem Dateiserver das Programmier-Projekt *ch.fhnw.algd2.arraycollections* zur Verfügung.

Das *java.util.Collection* Interface definiert das erwartete Verhalten der Datenstrukturen mit Vor- und Nachbedingungen, lässt aber trotzdem Raum für verschiedene Implementierungen. Beispielsweise darf die Speicherkapazität der Collection begrenzt sein. Ein allfälliger „Überlauf" muss dann jedoch durch eine bestimmte Exception angezeigt werden. Es ist wichtig, sich an die Vorgaben zu halten, da andere Komponenten das im Interface definierte Verhalten erwarten.

Um das zu überprüfen, stehen Ihnen Unit-Tests zur Verfügung.

Hinweise zur Implementierung:

- Verwenden Sie *Generics*. So vermeiden Sie bei der *add*-Methode eine Class-Cast Exception.
- *null*-Werte sollen in Ihrer Collection *nicht* erlaubt sein.
- Es sind Unit-Tests vorhanden. Nicht immer werden optionale Exceptions geprüft. Ändern Sie die Unit-Tests ruhig ab, wenn Sie optionale Exceptions in Ihrer Implementation verwenden.
- In den Klassen sind jeweils *main*-Methoden vorhanden, die zeigen, wie Sie den Inhalt Ihrer Collection auf der Konsole ausgeben können.

Gehen Sie wie folgt vor:

1. Bilden Sie Gruppen, die zusammen die gleiche Variante bearbeiten.
2. Zeigen Sie die Verhaltensweise Ihrer Datenstruktur beim Einfügen der folgenden Zahlen-Sequenz: 12, 5, 28, 47, 12, 8

3. Importieren Sie das Projekt *ch.fhnw.algd2.arraycollections* in Ihre IDE (vgl. separate Anleitung).
4. Implementieren Sie die Methoden für Ihre Variante der Collection. Das Klassentemplate steht Ihnen bereits zur Verfügung.
5. Sie brauchen keine Vergrösserungsstrategien zu implementieren. Wenn es keinen Platz mehr hat, können eben keine Elemente mehr eingefügt werden. Testen Sie ihre Implementation mit den *Unit Tests*.

   Hinweis: Für eine binäre Suche kann die Methode *Arrays.binarySearch(…)* oder eine eigene Implementation verwendet werden. Begründen Sie Ihre Wahl.
6. Ergänzen Sie die Tabelle am Ende der Unterrichtsunterlagen für Ihre Implementierung.

Arbeitsblatt: Collections mit Arrays

Eine einfache Art, Collections zu implementieren, ist die Elemente in einem Array abzulegen. Enthält eine Collection *n* Elemente, sind diese im Array unter den Indices 0 bis *n*-1 abgelegt. Speichert man zusätzlich den aktuellen Wert für *n*, kann man die Collection-Grundoperationen *add, contains* und *remove* leicht implementieren. Das sollen Sie nun tun.

Wir haben gesehen, dass es vier Möglichkeiten gibt, eine Collection in einem Array zu implementieren. Für die beiden Definitionen *Set* und *Bag* je eine Version mit sortierter Speicherung und eine ohne. Also:

		Sortierung	
		Sortiert	**Nicht sortiert**
Semantik	**Set-Semantik**	*SortedSet*	*UnsortedSet*
	Bag-Semantik	*SortedBag*	*UnsortedBag*

Ihr Auftrag ist es nun, die in der Tabelle gekennzeichnete Variante zu implementieren. Als Rahmen steht Ihnen dazu auf dem Dateiserver das Programmier-Projekt *ch.fhnw.algd2.arraycollections* zur Verfügung.

Das *java.util.Collection* Interface definiert das erwartete Verhalten der Datenstrukturen mit Vor- und Nachbedingungen, lässt aber trotzdem Raum für verschiedene Implementierungen. Beispielsweise darf die Speicherkapazität der Collection begrenzt sein. Ein allfälliger „Überlauf" muss dann jedoch durch eine bestimmte Exception angezeigt werden. Es ist wichtig, sich an die Vorgaben zu halten, da andere Komponenten das im Interface definierte Verhalten erwarten.

Um das zu überprüfen, stehen Ihnen Unit-Tests zur Verfügung.

Hinweise zur Implementierung:

- Verwenden Sie *Generics*. So vermeiden Sie bei der *add*-Methode eine Class-Cast Exception.
- *null*-Werte sollen in Ihrer Collection *nicht* erlaubt sein.
- Es sind Unit-Tests vorhanden. Nicht immer werden optionale Exceptions geprüft. Ändern Sie die Unit-Tests ruhig ab, wenn Sie optionale Exceptions in Ihrer Implementation verwenden.
- In den Klassen sind jeweils *main*-Methoden vorhanden, die zeigen, wie Sie den Inhalt Ihrer Collection auf der Konsole ausgeben können.

Gehen Sie wie folgt vor:

1. Bilden Sie Gruppen, die zusammen die gleiche Variante bearbeiten.
2. Zeigen Sie die Verhaltensweise Ihrer Datenstruktur beim Einfügen der folgenden Zahlen-Sequenz: 12, 5, 28, 47, 12, 8

3. Importieren Sie das Projekt *ch.fhnw.algd2.arraycollections* in Ihre IDE (vgl. separate Anleitung).
4. Implementieren Sie die Methoden für Ihre Variante der Collection. Das Klassentemplate steht Ihnen bereits zur Verfügung.
5. Sie brauchen keine Vergrösserungsstrategien zu implementieren. Wenn es keinen Platz mehr hat, können eben keine Elemente mehr eingefügt werden. Testen Sie ihre Implementation mit den *Unit Tests*.

   Hinweis: Für eine binäre Suche kann die Methode *Arrays.binarySearch(…)* oder eine eigene Implementation verwendet werden. Begründen Sie Ihre Wahl.
6. Ergänzen Sie die Tabelle am Ende der Unterrichtsunterlagen für Ihre Implementierung.