

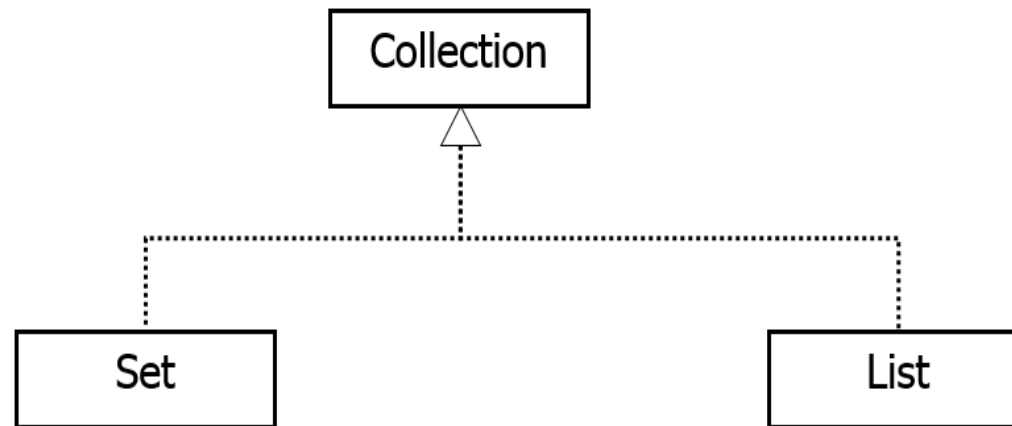
Java Collections (Sets und Bags)

Algorithmen und Datenstrukturen 2

- Grüne Farbe: Bitte im Script nachtragen

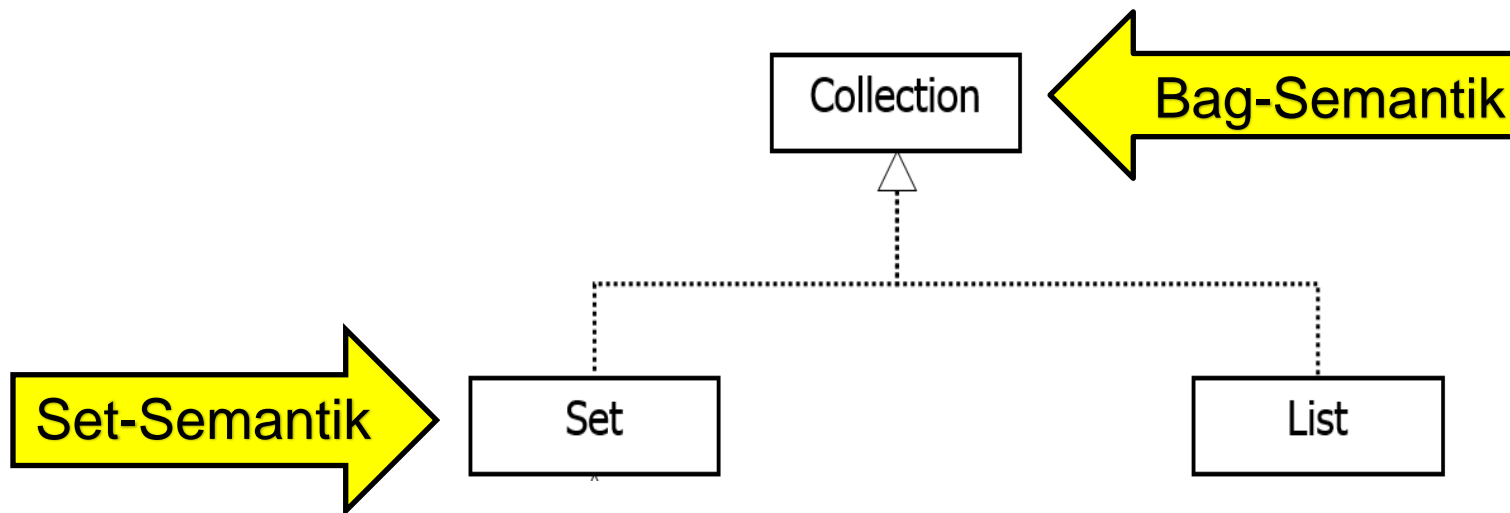
Java Collection Interface

- Ansammlung mehrerer Elemente der selben Klasse (meistens)
- Standard-Operationen wie Hinzufügen, Suchen, Entfernen



1.4 Bag-Semantik vs. Set-Semantik

- **Bag-Semantik**
 - Mehrfachvorkommen eines Elements sind erlaubt
- **Set-Semantik**
 - Element darf nur 1x in der Collection erscheinen (DB Unique)



1.4 Einfügen in Collections (Script 1.4, Seite 2)

- Fügen Sie folgende Werte nacheinander in die Datenstrukturen ein:
 - 1, 5, 3, 2, 4, 1, 3, 6
- Bag-Semantik
- Set-Semantik:

1.4 Einfügen in Collections

- Fügen Sie folgende Werte nacheinander in die Datenstrukturen ein:
 - 1, 5, 3, 2, 4, 1, 3, 6
- Bag-Semantik
 - 1, 5, 3, 2, 4, 1, 3, 6
- Set-Semantik:
 - 1, 5, 3, 2, 4, 6 (1 und 3 werden nur 1x eingefügt)

1.6 Einfügen in Collections in sortierter Reihenfolge

- Fügen Sie folgende Werte nacheinander in die Datenstrukturen ein:
 - 1, 5, 3, 2, 4, 1, 3, 6
- Bag-Semantik
- Set-Semantik:

1.6 Einfügen in Collections in sortierter Reihenfolge

- Fügen Sie folgende Werte nacheinander in die Datenstrukturen ein:
 - 1, 5, 3, 2, 4, 1, 3, 6
- Bag-Semantik
 - 1, 1, 2, 3, 3, 4, 5, 6
- Set-Semantik:
 - 1, 2, 3, 4, 5, 6 (1 und 3 werden nur 1x eingefügt)

Kombinationen

- Mögliche Kombinationen

	Sortiert	Unsortiert
Bag-Semantik	SortedBag	UnsortedBag
Set-Semantik	SortedSet	UnsortedSet

Collection und Set Interface

- Konsultieren Sie die Dokumentation zu den Java Interfaces
 - Collection
(<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collection.html>)
 - Set
(<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collection.html>)
- Allgemeines:
 - Keine Exception: Aufruf gemäss Spezifikation erfolgreich
 - Boolean-Rückgabewert: Wurde Collection verändert?
- Beantworten Sie im Script auf Seite 3 die Fragen zu:
Einige Aufrufbeispiele für eine Variable `Collection<Integer> c`:

Collection und Set Interface

1. `c.add(1)` gibt `false` zurück.
 - Bei `c` handelt es sich um ein `Set` und eine `1` war bereits vorhanden
 - (`Bag` würde mehrere `1` erlauben, `Exception` im Fehlerfall)
2. `c.add(1)` gibt `true` zurück.
 - `c instanceof Set`: `1` war noch nicht enthalten und wurde hinzugefügt
 - `!(c instanceof Set)`: Eine (weitere) `1` wurde hinzugefügt
3. `c.add(null)`;
 - Falls `null` erlaubt: `null` wird in Sammlung aufgenommen, Rückgabewert entsprechend `1. / 2.`
 - Falls `null` nicht erlaubt: `NullPointerException`

Collection und Set Interface

4. `c.remove(1); c.remove(1);` gibt beide Male `true` zurück.
 - `!(c instanceof Set)`: Element konnte mehrere Male erfolgreich entfernt werden
5. Aufruf um zu prüfen, ob `c` mindestens ein Element `1` enthält:
 - `c.contains(1);`
6. Programm zum Löschen aller Elemente `1` aus `c`, wenn gilt `c instanceof Set`:
 - Ein Set kann maximal eine `1` beinhalten.
 - `c.remove(1);`
7. Programm zum Löschen aller Elemente `1` aus `c`, wenn gilt `!(c instanceof Set)`:
 - `while (c.remove(1)){}`

Lernziele

- Sie kennen den Unterschied zwischen Set- und Bag-Semantik und können geeignete begründet eine Auswahl treffen
- Sie können sich begründet für eine sortierte oder unsortierte Bag- / Set-Variante entscheiden
- Sie können Array-basierte Collections für Bag- und Set-Semantik implementieren (sortiert und unsortierte Reihenfolge)
- Sie können für gegebenen Implementierungen die Komplexitätsklasse für Operationen wie add, contains oder remove analysieren

Einzel- oder Partnerarbeit

- Nehmen Sie das Arbeitsblatt zur Hand und lösen Sie dies für ihre Collection
 - Partnerarbeit: Jede Person programmiert am eigenen Rechner
- Ziel der Übung:
 - Implementieren einer eigenen Collection
 - Abschätzen des Zeitaufwands für verschiedene Operationen
 - Gedanken zu Einsatzzwecken