1 Binärer Suchbaum

Erstellen Sie eine Klasse `de.hska.iwi.ads.solution.tree.BinarySearchTree`, welche die Klasse `de.hska.iwi.ads.dictionary.AbstractBinaryTree` als Basisklasse besitzt. Implementieren Sie einen binären Suchbaum ohne Löschen. Im Gegensatz zu den beiden vorangehenden analogen Aufgaben, benötigen Sie nun die beim generischen Typparameter über `Comparable<K>` angegebenen Totalordnung der Schlüssel für Größenvergleiche.

Über die Vererbung besitzt `AbstractBinaryTree` bereits:

- Eine verschachtelte Member-Klasse `Node` mit linken und rechten Verweis auf die Teilbäume sowie das gespeicherten Schlüssel-/Wertepaar als Entry-Objekt.
- Den Verweis auf dem Wurzelknoten `protected Node root`. Dieser ist initial `null`.
- Eine Iterator-Implementierung, mit dessen Hilfe die Methode `entrySet()` implementiert ist.

Es müssen wieder die folgenden zwei Methoden implementiert werden:

- `public V get(Object o)`: Gibt den Wert der sich unter dem angegebenen Schlüssel `o` im Binärbaum zurück. Es wird null zurückgegben, falls kein Wert mit diesem Schlüssel existiert. Casten Sie den Parameter innerhalb Ihrer Implementierung auf den generischen Typ K.
- `public V put(K key, V value)`: Fügt den Wert `value` in einem neuen Blatt innerhalb des Binärbaums ein. Falls jedoch schon ein Wert mit dem angegebenen Schlüssel `key` im Binärbaum vorhanden ist, so wird in dessen Knoten der Wert mit `value` überschrieben und der alte Wert zurückgegeben.

Ihre Implementierung darf keine Methoden, der Oberklassen aufrufen. Achten Sie auch darauf, bei Deklaration mit generischen Klassen die generischen Typen anzugeben.

Der Zeitaufwand für Einfügen und Suchen soll im Durchschnitt $\Theta(\log n)$ betragen. Sie dürfen den Baum auf keinen Fall sequentiell durchsuchen.

Testen Sie Ihre Implementierung mit JUnit. `de.hska.iwi.ads.dictionary.MapTest` kann als Basisklasse für Ihre Testklasse verwendet werden.