```java
import java.util.HashMap;

public class DocumentCollection {
    private HashMap<String, Document> documents = new HashMap<>();

    public DocumentCollection(String[] paths) {
        try {
            for (String path : paths) {
                documents.put(path, new Document(path));
            }
        } catch (Exception e) {
            System.out.println("Error Reading file!: " + e.getMessage());
        }
    }

    /**
     * calculates tf_idf
     * @param term:     term w to calculate tf_idf
     * @param document: document d in collection to calculate tf_idf
     * @return value for tf_idf
     */
    public double get_tf_idf(String term, String document) {
        double tf = documents.get(document).get_tf(term);
        try {
            double idf = get_idf(term);
            return calculate_tf_idf(tf, idf);
        } catch (Exception e) {
            System.out.println("Term does not occur: " + e.getMessage());
            return 0;
        }
    }

    public static double calculate_tf_idf(double tf, double idf) {
        return tf * idf;
    }

    /**
     * returns idf
     *
     * @param term: given term i to use to calculate idf
     * @return value for idf
     * @throws Exception
     */
    public double get_idf(String term) throws Exception {
        int n_i = 0;
        for (Document document : documents.values()) {
            if (document.doesTermAccur(term)) {
                n_i += 1;
            }
        }
        return calculate_idf(documents.size(), n_i);
    }

    /**
     * Method to calculate inverse document frequency
     * @param N:   Total number of documents
     * @param n_i: Number of occurrences of a term
     * @return value for idf
     */
    public static double calculate_idf(double N, double n_i) {
        return log2(N / n_i);
    }

    private static double log2(double x) {
        return (Math.log(x) / Math.log(2));
    }
}
```