Binärdateien

Teste deine Applikationen mit den zur Verfügung gestellten Tests.

1 CheckInt

Eine Binärdatei hat folgenden Aufbau:
Zunächst steht ein Kontrollinteger und dann nur noch doubles. Ansonsten ist in der Datei nichts gespeichert.
Die Datei ist gültig, wenn der Kontrollinteger der Anzahl der danach gespeicherten Doublewerte entspricht. Schreibe folgende Funktionen:

1. `public static void createFile(String filename, double... values) throws IOException`
   Erzeugt eine Testdatei wie oben definiert.

2. `public static boolean isValidFile(String filename) throws IOException`
   Überprüft, ob die Datei wie oben definiert aufgebaut ist.

3. `public static String getFileInfo(String filename) throws IOException`
   Liefert den Wert des Kontrollintegers und dann in der nächsten Zeile alle abgespeicherten Doublewerte mit zwei Nachkommastellen. Beispielausgabe:

   ```
   Saved values: 5
   1,20 7,80 3,50 6,80 4,50
   ```

4. `public static void append(String filename, double toAppend) throws IOException`
   Schreibt einen weiteren Doublewert ans Ende der Datei und erhöht den Kontrollinteger um 1.

2 Numbers

Eine Binärdatei hat folgenden Aufbau:
Zunächst steht ein Byte mit Wert 0 oder 1. Hat das Byte den Wert 0, so folgt ein Integer, hat das Byte den Wert 1, so folgt ein Double. Diese Sequenz (Byte - Integer/Double) wiederholt sich bis zum Dateiende. Schreibe folgende Funktionen:

1. `public static List<Number> getContents(String filename) throws IOException`
   Liefert eine Liste aller Werte, welche in der Datei gespeichert sind. Ist die Datei nicht nach obigem Schema aufgebaut, so wirft die Methode eine `IllegalArgumentException`.

2. `public static Map<String, Set<Number>> groupByType( List<? extends Number> numbers)`
   Liefert eine Map, deren Keys die Subklassen die von Number sind. Der jeweilige Value sind aufsteigend sortiert alle Werte dieser Subklasse in der übergebenen Liste.