Aufgabe 1

1.1 Ruby-Code in Java-Code übersetzen

Lernziele:

- Unterschiede zwischen den Sprachen Java und Ruby am Beispiel kennenlernen.
- Umgang mit der JUnit-Test-Umgebung

Material: Script zur Vorlesung v1

Aufgabe:

1. Übersetzen Sie das mit gelieferte Ruby-Counter Beispiel in lauffähigen Java Code.
2. Schreiben Sie JUnit-Tests für das Counter Beispiel.

1.2 Datentypen und Wertebereiche

Lernziele:

- Wertebereiche einzelner Datentypen kennenlernen
- Bewusstsein für Überlauf bei Operationen mit integralen Datentypen entwickeln
- Gefahren des Cast-Operators kennenlernen
- Ungenauigkeiten und Unterlauf bei Gleitkommazahlen kennenlernen

Material: Script zur Vorlesung v1 und v2

Aufgaben: Erweitern Sie für diese Aufgabe die Klasse `BasetypeRanges` in Package `a12`

1. Überlauf bei Addition von ganzzahligen Werten:
   - Schreiben Sie eine statische Methode `add`, die zwei `short`-Werte addiert und als Ergebnis einen `short`-Wert zurückliefert.
   - Schreiben Sie eine statische Methode `add`, die zwei `byte` Werte addiert und als Ergebnis einen `byte`-Wert zurückliefert.
   - Rufen Sie die beiden `add`-Methoden so auf, dass beim Addieren ein Überlauf entsteht, jeweils für den positiven und den negativen Zahlenbereich.
2. Prüfen auf Wertebereich:
   - Schreiben Sie eine statische Methode `isShort(long l)`, die prüft ob der Wert von `l` im Wertebereich von `short` liegt. Geben Sie zwei Varianten der Implementierung an.
   - Wie viele Methoden müssen Sie für Datentypen ganzzahliger Werte insgesamt schreiben?
3. Prüfen auf Überlauf:

   - Schreiben Sie eine statische Methode `boolean overflowAdd(short s1, short s2)`, die prüft, ob bei der Addition zweier `short`-Werte ein Überlauf entstanden ist.
   - Schreiben Sie die statische Methode `overflowAdd` auch für zwei `byte`-Werte.
4. Korrekte Binärdarstellung von `byte`, `char` und `short` Werten als String:
   - `Byte`, `Short` und `Character` haben keine Methode `toBinaryString`.
   - Schreiben Sie eine statische Methode `toBinaryString(byte b)`, die eine korrekte binäre Darstellung von dem Wert `b` erzeugt. Eine korrekte binäre Darstellung verwendet nur die untersten 8 Bit der Bitdarstellung eines Integers. Verwenden Sie die Methoden `Integer.toBinaryString` und `substring` der Klasse `String`.
   - Schreiben Sie diese Methode auch für `short` und `char`.
   - Für einen der beiden Datentypen `short` oder `char` kann die Methode `Integer.toBinaryString` verwendet werden. Für welchen Datentyp? Begründen Sie kurz Ihre Antwort.
5. Rufen Sie im `main` der Klasse `BasetypeRanges` die statischen Methoden `floatPrecision` und `doubleUnderflow` auf. Was beobachten Sie? Wie lassen sich die Beobachtungen erklären?

1.3 Zufallszahlen

Lernziele:

- Umgang mit wiederholbaren Folgen von Zufallszahlen
- Zufallszahlen in beliebigen Intervallen erzeugen
- Arrayindizes transformieren

Material: Script zur Vorlesung v1 und v2

Aufgabe: Ergänzen Sie die Klasse `BasetypeGenerator`. Die Klasse generiert im `main` ein Array von verschiedenen Werten der Basisdatentypen und eines beliebigen Referenztyps, mischt die Elemente und schreibt die zufällig gemischten Elemente in die Datei `basetype.data`.

Sie sollen das Array mit zufälligen Werten der Basisdatentypen in vorgegebenen Intervallen befüllen.

1. Schreiben Sie dazu zunächst für jeden der Basisdatentypen `byte, short, int, long, float, double` eine statische Methode, die eine Zufallszahl im Intervall `[a,b]` würfelt und dazu einen Zufallszahlengenerator verwendet, der beim Aufruf übergeben wird. `a` und `b` sollen alle Werte des Wertebereichs für den jeweiligen Datentyp annehmen können. Der Methodenkopf für `byte` sieht wie folgt aus: `byte random(Random r, byte a, byte b)`. Testen Sie Ihre Methoden und stellen Sie sicher, dass die gewürfelte Zahl im geforderten Intervall liegt.
2. Lesen Sie im `main` der Klasse die Integer-Größe `n` ein, die die Anzahl der zufälligen Werte für die einzelnen Typen steuert.
3. Ergänzen Sie die statische Methode `generate(int n)` der Klasse so, dass zuerst `bools`* zufällige boolesche Werte in das Array `collection` geschrieben werden, dann ab dem

Index `bools bytes`*zufällige `byte` Werte in `collection` geschrieben werden usw. Erzeugen Sie die zufälligen Werte mit den Methoden aus 1.3.1.

4. Am Ende schreiben Sie noch `notClassifiedOnes`* beliebige Werte in das Array. Am einfachsten ist es Strings zu verwenden und diese durchzunummerieren.
5. Rufen Sie das Programm in der Konsole auf und übergeben Sie einen Wert für `n`.

Hinweis: Das Programm gibt am Ende eine Statistik aus. Damit können Sie prüfen, ob Ihr Programm unter 1.4 korrekt arbeitet.

Beispiel: für n = 10

bool:9 byte:4 short:10 int:3 long:4 float:3 double:8 not classified:7

1.4 Benutzereingaben typsicher Lesen

Lernziele:

1. Funktionen der Klasse `Scanner` für das typsichere Einlesen von Benutzereingaben kennenlernen
2. Eingabe in eine Datei umlenken (wird hier erklärt).

Material: Script zur Vorlesung v1 und v2

Aufgabe: Sie sollen in der Klasse `BasetypeReader` eine Methode implementieren, die mit dem `Scanner` die Eingabe liest und ermittelt, wieviel `boolean`, `byte`, `short`, `int`, `long`, `float`, `double` und Werte, die keinem Basisdatentypen zugeordnet werden können, in der Eingabe vorhanden sind. Damit Sie nicht jedes Mal Kolonnen von Zahlen und anderen Werten eingeben müssen, werden wir die Daten aus Aufgabe 1.3 nutzen.

Umleiten des Eingabestroms auf eine Datei:

Wir wollen die Eingabe der Daten in eine Datei umleiten. Das erreichen wir durch den folgenden Aufruf des Programms `BasetypeReader`:

```
java a14.BasetypeReader <basetype.data
```

Das `<` lenkt den Eingabestrom um.

1. Implementieren Sie mit geeigneten Methoden der Klasse `Scanner` die statische Methode `countBaseTypes()`, die die Anzahl der Werte der oben genannten Basisdatentypen und der Nicht-Basisdatentypen zählt und am Ende ausgibt.
   Ausgabe: bool:9 byte:4 short:10 int:3 long:4 float:11 double:0 not classified:7
2. Rufen Sie das Programm `BasetypeReader` in der Konsole auf und lenken Sie den Eingabestrom auf die Datei `basetype.data` um.

Hinweis: Wenn die Datei `basetype.data` nicht gefunden wird, dann müssen Sie das Programm aus **1.3** nochmals ausführen.

1.5 Arrayinhalte unendlich oft lesen

Lernziele:

1. Verwenden von Modulo und Autoinkrement.

Material: Script zur Vorlesung v1 und v2

Aufgabe:

1. Sie sollen in der Klasse `RingReader` in der Endlosschleife `while(true)` die Elemente des Arrays `a` lesen und ausgeben. Der Zugriff erfolgt über einen Zähler `ct`, der bei jedem Lesevorgang inkrementiert wird. Wenn der letzte gültige Index des Arrays erreicht ist, soll bei Index 0 weitergelesen werden. Entwickeln Sie eine 1-zeilige Lösung!
2. Lesen Sie aus dem Programmparameter die Länge des Arrays.
3. Starten Sie das Programm in der Konsole und übergeben Sie dabei die Länge des Arrays.

Anwendungsbeispiel: Eine sich wiederholende Playlist eines Mediaplayers.