hochschule mannheim

FAKULTÄT FÜR INFORMATIK

Prof. Dr. Peter Knauber

PR1 – Programmieren 1 | Bachelor-Studiengang Informatik | WS2017/2018

Übungsblatt 9: Klassen und Objekte, Abstrakte Datentypen

Ausgabe: 11.12.17
Abgabe: 18.12.17 12:00 Uhr elektronisch mittels Git

Aufgabe 1

Ergänzen Sie die Klasse `BigInt` vom letzten Übungsblatt um folgende Methoden:

- `void times( BigInt number )`

  Multipliziert die als Parameter übergebene Zahl vom Typ `BigInt` mit der im Objekt vorliegenden Zahl und ersetzt diese mit dem Ergebnis. Das Ergebnis der Multiplikation wird also anstelle der zuvor vorliegenden Zahl im Objekt gespeichert (so dass man damit weiterarbeiten kann). Das als Parameter übergebene Objekt bleibt unverändert. Beispiel:

  ```
  BigInt zahl1 = new BigInt("45");
  BigInt zahl2 = new BigInt("23");
  zahl1.times(zahl2);
  println(zahl1.toString());   // gibt "1035" auf der Konsole aus
  println(zahl2.toString());   // gibt "23" auf der Konsole aus
  ```

  Tipp: Vergegenwärtigen Sie sich bei der Implementierung die schriftliche Multiplikation.

- `boolean greater(BigInt number)`

  Liefert genau dann `true`, wenn die vorliegende Zahl größer ist als der Parameter `number`, sonst `false`. Beispiel:

  ```
  println(zahl1.greater(zahl2)); // gibt "true" auf der Konsole aus
  println(zahl2.greater(zahl1)); // gibt "false" auf der Konsole aus
  ```

Erweitern Sie Ihre `main`-Methode, um die neuen Methoden zu testen.

Hinweis:
Es kommt nicht darauf an, die Methoden besonders effizient zu implementieren! Sie dürfen es sich einfach machen!

Aufgabe 2

Gewünscht ist ein Abstrakter Datentyp (Lineare) *Liste*, der ganze Zahlen aufnehmen kann, mit folgenden Operationen:

- addFirst: Liste × Element → Liste

  fügt ein neues Element am Beginn einer Liste ein

- addLast: Liste × Element → Liste

  hängt ein neues Element am Ende einer Liste an

- getFirst: Liste → Element

  liefert das erste Element der Liste

  Vorbedingung: Die Liste ist nicht leer, sonst wird eine `PRException` ausgelöst

- getLast: Liste → Element

  liefert das letzte Element der Liste

  Vorbedingung: Die Liste ist nicht leer, sonst wird eine `PRException` ausgelöst

- getAt: Liste × Position → Element

  liefert das Element an der angegebenen Position der Liste

  Vorbedingung: Die Liste hat genügend Elemente, sonst wird eine `PRException` ausgelöst

- removeFirst: Liste → Liste

  löscht das erste Element der Liste

  Vorbedingung: Die Liste ist nicht leer, sonst wird eine `PRException` ausgelöst

- contains: Liste × Element → boolean

  sucht ein Element in einer Liste; Ergebnis ist `true`, wenn Element in der Liste, sonst `false`

- delete: Liste × Element → Liste

  entfernt ein Element aus der Liste, sofern es drin ist (das erste Vorkommen wird entfernt)

- clear: Liste → Liste

  entfernt alle Elemente aus der Liste

- isEmpty: Liste → boolean

  liefert `true` genau dann, wenn die Liste leer ist, sonst `false`

- size: Liste → int

  liefert die Länge der Liste, d.h. die Anzahl der Elemente

- clone: Liste → Liste

  liefert eine (flache) Kopie der Liste

- empty: → Liste

  erzeugt eine neue leere Liste

Implementieren Sie eine Klasse `LinearListWithArray`, welche die angegebenen Operationen zur Verfügung stellt. Bei den Operationen `addFirst`, `addLast`, `removeFirst`, `delete` und `clear` soll die Liste jeweils verändert werden (d.h. der Ergebnistyp ist `void`). Die Operation `empty` ist durch einen parameterlosen Konstruktor zu implementieren.
Benutzen Sie ein Array, um die jeweils in der Liste enthaltenen Elemente abzulegen. Die Länge des Arrays soll jeweils der Anzahl der enthaltenen Elemente entsprechen.
Überlegen Sie sich, wie Sie sicherstellen können, dass Ihre Operationen korrekt funktionieren.

Allgemeines

- Die Aufgaben sind in Eclipse zu bearbeiten und beim Testat vorzuführen.
- Legen Sie für die Bearbeitung dieses Übungsblattes ein Paket namens `pr1.uebung09` an, in dem Sie Ihre Klassen anlegen.
- Erlaubt sind `MakeItSimple`-Funktionen (keine nicht besprochene Funktionalität aus der Java-Standard-Bibliothek) und das bisher erworbene Wissen aus den PR1-Vorlesungen. Sie müssen alle(!) von Ihnen verwendeten Konstrukte der Sprache sowie alle verwendeten Methoden, die nicht aus der Hilfsbibliothek `MakeItSimple` stammen, gut erklären und nötigenfalls im Testat selbst programmieren können!
- Sie geben ab, indem Sie vor Ende der Abgabefrist Ihr Projekt mit den lauffähigen Programmen des Übungsblattes in das Repository auf *ovid* pushen. Achten Sie darauf, ob Sie die Kontroll-E-Mail bekommen! Die letzte hochgeladene Version Ihres Projekts wird gewertet. Andere Abgaben, ob elektronisch oder auf Papier, zählen als **nicht abgegeben**!