Projektblatt P6 (19 P)

Abgabe: Freitag 3. November 2023, 12:00h
Entpacken Sie zunächst die Archiv-Datei `vorgaben-p07.zip`, in der sich die Rahmendateien für die zu lösenden Aufgaben befinden. Ergänzen Sie die Dateien durch Ihre Lösungen gemäß der Aufgabenstellung unten bzw. fügen Sie in einigen Fällen neue Klassen (ggf. in neuen Dateien) hinzu. Der hinzuzufügende Java-Sourcecode sollte **syntaktisch richtig** und **vollständig formatiert** sein. Alle Dateien sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die `.java` Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.P07.zip`, welches Sie auf Ilias hochladen.
Führen Sie dazu in dem Verzeichnis, in dem Sie die Dateien bearbeitet haben, folgenden Befehl auf der Kommandozeile aus:

```
zip  IhrNachname.IhrVorname.P07.zip *.java
```

Aufgabe 1: Strings 7 (=2+2.5+2.5) Punkte

"*Als Palindrom werden in der Sprachwissenschaft Wörter, Wortreihen oder Sätze bezeichnet, die rückwärts gelesen genau denselben Text oder zumindest einen Sinn ergeben. Groß-/Kleinschreibung, Wortgrenzen und Satzzeichen sind beim Rückwärtslesen gegebenenfalls zu ändern.*" (Quelle: Wikipedia)
Im folgenden sollen Sie mehrere Methoden in der Klasse `StringTest` implementieren, mit deren Hilfe überprüft werden kann, ob ein vorgegebener Satz ein Palindrom darstellt. Gehen Sie hierzu wie folgt vor:

(a) In der Methode `filterLetters` sollen zunächst alle Buchstaben aus einem als Parameter übergebenen String ausgefiltert und zu einem String zusammengefügt werden, der am Ende als Ergebnis zurückgegeben wird.

Legen Sie hierzu zunächst eine Instanz der Klasse `StringBuffer` an, dessen Inhalt Sie am Ende zurückgeben. Durchlaufen Sie dann den als Parameter übergebenen String zeichenweise. Fügen Sie jedes Zeichen, das einen Buchstaben darstellt, am Ende des `StringBuffer` Objekts ein. Wandeln Sie dabei Großbuchstaben in Kleinbuchstaben um.
Hinweis: Verwenden Sie hierbei die folgenden Methoden:

- Die statische Methode `isLetter` aus der Klasse `Character` prüft, ob ein als Parameter übergebener `char`-Wert einen Buchstaben darstellt (Rückgabe `true`) oder nicht (Rückgabe `false`).
- Die statische Methode `toLowerCase` aus der Klasse `Character` wandelt einen als Parameter übergebenen `char`-Wert in einen Kleinbuchstaben um, wenn der Wert ein Großbuchstabe ist, und gibt diesen als Ergebnis zurück. Ansonsten wird der Wert unverändert zurückgegeben.

(b) Implementieren Sie die Methode `isPalindromeIt`, die **iterativ** prüfen soll, ob die Folgen der Zeichen in einem als Parameter übergebenen String von links nach rechts und von rechts nach links gelesen, identisch sind. Definieren Sie hierzu zwei Indexvariablen, die Sie mir der ersten und letzten Position in dem String initialisieren. Überprüfen Sie, ob die Zeichen an diesen beiden Positionen die gleichen sind und geben Sie den Wert `false` zurück, wenn dies nicht der Fall ist. Ansonsten inkrementieren (bzw. dekrementieren) Sie die erste (bzw. zweite) Indexvariable.

Wiederholen Sie dies solange, bis der Wert der ersten Indexvariable größer oder gleich der zweiten ist. Wenn dieser Fall eintritt, geben Sie anschließend den Wert `true` zurück.

(c) Implementieren Sie die Methode `isPalindromeRek`, die **rekursiv** prüfen soll, ob ein als Parameter übergebener String, der nur aus Kleinbuchstaben besteht, ein Palindrom darstellt. Verwenden Sie hierbei folgende Regeln :

(1) Ein String, der höchstens ein Zeichen enthält, ist ein Palindrom.

(2) Ein String ist genau dann ein Palindrom, wenn das erste und das letzte Zeichen gleich sind und wenn der Teilstring, der vom zweiten bis zum zweitletzten Zeichen in dem String reicht, ein Palindrom ist.

Testen Sie ihre Implementierung mit den in der `main`-Methode der Klasse vorgegebenen Beispielen. Die ersten sechs Strings in dem Feld `pBeispiele` stellen Palindrome dar, die restlichen beiden nicht.

Aufgabe 2: Datumsklassen 10 (= 1+2+1+2+1+3) Punkte

In dieser Aufgabe sollen Sie mehrere Methoden implementieren, die dazu dienen ein Feld von Instanzen einer Klasse für bekannte Personen aus der Geschichte zu erstellen, zu sortieren und auszugeben.
Hierzu ist in der Klasse `HistorischePerson` Folgendes (teilweise) vorgegeben:

- Eine Historische Person wird durch vier Attribute beschrieben, die den Namen, eine Kurzbeschreibung sowie Geburts- und Todesdatum umfassen.
- Der (noch leere) Konstruktor soll mit Hilfe eines als Parameter gegebenen Strings (mit allen erforderlichen Daten) alle Attribute initialisieren.
- Die (noch leere) Methode `alter` soll das Alter einer Person in Jahren, Monaten und Tagen berechnen.
- Die (noch leere) Methode `toString` soll einen String generieren und zurückgeben, der eine Historische Person repräsentiert.

- Die (noch leere) Methode `compareTo` soll zwei Personen nach ihrem Alter (gemessen in Tagen) vergleichen.
- Die (noch leere) Methode `sort` soll ein Feld von Historischen Personen nach ihrem Alter aufsteigend sortieren.

In der Klasse `TestHistorischePerson` finden Sie Sie darüberhinaus noch Folgendes:

- Die Klassenvariable `persData` beinhaltet 15 Strings, die jeweils eine einzelne historische Person beschreiben. Jeweils getrennt durch ein Semikolon, besteht jeder String in dem Feld aus vier Einträgen für den Namen, eine Kurzbeschreibung sowie das Geburts- und das Todesdatum. Die beiden Datumsangaben sind Ziffernfolgen, die aus vier Ziffern für das Jahr und je zwei Ziffern für den Monat und den Tag bestehen.
- In der main-Methode wird ein Feld für die Historischen Personen angelegt und mit Instanzen der Klasse `HistorischePerson` gefüllt. Dieses Feld wird dann sortiert und ausgegeben.

Ergänzen Sie nun die beiden Klassen wie im Folgenden beschrieben.

(a) Ergänzen Sie das Feld `persData` in der Klasse `TestHistorischePersonen` durch mindestens drei weitere Strings, die verschiedene Historische Personen in dem oben beschriebenen Format repräsentieren.

(b) Implementieren Sie den Konstruktor in der Klasse `HistorischePerson`: Teilen Sie den als Parameter übergebenen String zunächst mit Hilfe der Methode `split` aus der Klasse `String` in vier Teile auf. Initialisieren Sie mit den in dem resultierenden Feld enthaltenen vier Strings die Attribute der Klasse. Gehen Sie bei den beiden Datumsstrings dabei so vor, das sie zunächst die Teilstrings mit den jeweiligen Ziffernfolgen für das Jahr, den Monat und den Tag extrahieren und zu `int` Werten umwandeln, aus der Sie dann mit Hilfe einer passenden Klassenmethode aus der Klasse `LocalDate` ein Datum erzeugen. Verwenden Sie hier **nicht** die Methode `parse` der Klasse `LocalDate`, auch wenn dies einfacher wäre.

(c) Implementieren Sie die Methode `alter` in der Klasse `HistorischePerson`: Die Methode soll eine Instanz der Klasse `Period`, die den Zeitraum zwischen der Geburt und dem Tod der Person repräsentiert, für die die Methode aufgerufen wird.

(d) Implementieren Sie die Methode `toString` in der Klasse `HistorischePerson`: Legen Sie zunächst ein Formatobjekt an, welches Daten wie im folgenden Beispiel formatiert und bestimmen Sie das Alter einer Person in Jahren, Monaten und Tagen mit Hilfe der Methode aus Aufgabenteil (c). Generieren Sie dann einen String, der eine Historische Person wie folgt repräsentiert:

```
Isaac Newton (Englischer Physiker, Astronom und Mathematiker):
    4. Jan. 1643 - 31. März 1727
    Alter: 84 Jahr(e), 2 Monat(e) und 27 Tag(e)
```

(e) Implementieren Sie die Methode `compareTo` in der Klasse `HistorischePerson`: Diese Methode soll einen positiven (negativen) Wert zurückgeben, wenn die Person, für die die Methode aufgerufen wird, eine, in Tagen gemessene, längere (kürzere) Lebenszeit hatte als die Person, die als Parameter übergeben wird. Sind zwei Personen exakt gleich alt geworden, soll die Methode den Wert 0 zurückgeben. Hinweis: Verwenden Sie das Objekt `ChronoUnits.DAYS` und die Klassenmethode `between`, die Sie auf diesem Objekt aufrufen können.

(f) Implementieren Sie die Methode `sort` in der Klasse `HistorischePerson`: Sortieren Sie den Inhalt des als Parameter übergebenen Felds aufsteigend nach der Lebenzeit der Personen, gemessen in Tagen. Verwenden Sie hierzu folgenden Algorithmus:

(1) Wiederholen Sie Schritt 2 genau $n-1$ Mal, wobei n die Länge des als Parameter übergebenen Felds ist.

(2) Durchlaufen Sie das als Parameter übergebene Feld von links nach rechts und vergleichen Sie jeweils zwei aufeinanderfolgende Historische Personen mit Hilfe der Methode `compareTo` (aus Aufgabenteil (e)). Ist die Lebenszeit der ersten Person länger als die der zweiten, tauschen Sie die beiden Personen in dem Feld aus.

Die Ausgabe der Klasse `TestHistorischePersonen` sollte dann (mit dem ursprünglichen Feldinhalt) wie folgt aussehen:

```
artin Luther King (US-amerikanischer Bürgrrechtler):
    15. Jan. 1929 - 4. Apr. 1968
    Alter: 39 Jahr(e), 2 Monat(e) und 20 Tag(e)
Friedrich Schiller (Deutscher Dichter und Philosoph):
    10. Nov. 1759 - 9. Mai 1805
    Alter: 45 Jahr(e), 5 Monat(e) und 29 Tag(e)
James Cook (Britischer Seefahrer und Entdecker):
    7. Nov. 1728 - 14. Feb. 1779
    Alter: 50 Jahr(e), 3 Monat(e) und 7 Tag(e)
Maria Theresia von Österreich (Österreichische Kaiserin):
    13. Mai 1717 - 29. Nov. 1780
    Alter: 63 Jahr(e), 6 Monat(e) und 16 Tag(e)
Johann Sebastian Bach (Deutscher Komponist und Musiker):
    31. März 1685 - 28. Juli 1750
    Alter: 65 Jahr(e), 3 Monat(e) und 28 Tag(e)
Robert Koch (Deutscher Nediziner und Mikrobiologe):
    11. Dez. 1843 - 27. Mai 1910
    Alter: 66 Jahr(e), 5 Monat(e) und 16 Tag(e)
Marie Curie (Physikerin und Chemikerin):
    7. Nov. 1867 - 4. Juli 1934
    Alter: 66 Jahr(e), 7 Monat(e) und 27 Tag(e)
Leonardo Da Vinci (Italienischer Maler, Bildhauer und Architekt):
    15. Apr. 1452 - 2. Mai 1519
    Alter: 67 Jahr(e), 0 Monat(e) und 17 Tag(e)
Charles Darwin (Britischer Naturforscher):
    12. Feb. 1809 - 19. Apr. 1882
    Alter: 73 Jahr(e), 2 Monat(e) und 7 Tag(e)
Alexander Fleming (Britischer Mediziner und Bakteriologe):
    6. Aug. 1881 - 11. März 1955
    Alter: 73 Jahr(e), 7 Monat(e) und 5 Tag(e)
Albert Einstein (Theoretischer Physiker):
    14. März 1879 - 18. Apr. 1955
    Alter: 76 Jahr(e), 1 Monat(e) und 4 Tag(e)
Mahadma Gandhi (Indischer Rechtsanwalt, Asket und Pazifist):
    2. Okt. 1869 - 30. Jan. 1948
```

```
    Alter: 78 Jahr(e), 3 Monat(e) und 28 Tag(e)
Isaac Newton (Englischer Physiker, Astronom und Mathematiker):
    4. Jan. 1643 - 31. März 1727
    Alter: 84 Jahr(e), 2 Monat(e) und 27 Tag(e)
Alexander von Humboldt (Deutscher Forschungsreisender):
    14. Sept. 1769 - 6. Mai 1859
    Alter: 89 Jahr(e), 7 Monat(e) und 22 Tag(e)
Nelson Mandela (Südafrikanischer Staatsmann):
    18. Juli 1918 - 5. Dez. 2013
    Alter: 95 Jahr(e), 4 Monat(e) und 17 Tag(e)
```

Aufgabe 3: Strings — 2 Punkte

Immer wieder wird der Vorschlag gemacht, die deutsche Sprache zu vereinfachen, beispielsweise durch eine Kürzung des Alphabets.
In der Klasse Alphabet wird ein Feld von Feldern mit je zwei String-Werten definiert, welche mögliche Ersetzungen definieren. So soll beispielsweise der Buchstabe C durch den Buchstaben K ersetzt werden oder der Buchstabe Z durch die Buchstabenkombination TS.
Ergänzen Sie die Methode `mapText` in der Klasse Alphabet so, dass ein String generiert und zurückgegeben wird, der aus dem als Parameter übergebenen String entsteht, indem nacheinander alle Ersetzungen durchgeführt werden, die in dem Feld `map` definiert sind. Berücksichtigen Sie dabei, dass Objekte der Klasse `String` nicht veränderbar sind. Man kann aber einer String-Variablen immer wieder das Ergebnis einer Operation auf diesem String zuweisen.
In der `main`-Methode der Klasse ist ein Beispieltext vorgegeben, der vor und nach den Ersetzungen ausgeben wird. Die Ausgabe bei einem Programmlauf sollte dann wie folgt aussehen:

```
Eine Kürzung des Alphabets um dreißig Prozent hat zahlreiche Vorteile.
Kinder lernen das Alphabet in Rekordzeit, Computertastaturen werden
schlanker und die Rechtschreibung wird logisch nachvollziehbarer.
->
Eine Kürtsung des Alphabets um dreissig Protsent hat tsahlreikhe Forteile.
Kinder lernen das Alphabet in Rekordtseit, Komputertastaturen werden
skhlanker und die Rekhtskhreibung wird logiskh nakhfolltsiehbarer.
```