Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Software-Qualität

Sommersemester 2024
2. Aufgabenblatt

Aufgabe 0.2 (0 Punkte)
Geben Sie das Lösungswort des Quiz aus der Lernnotiz an.

Aufgabe 2 (3 Punkte)
Gegeben Sei das Java-Projekt qsAufgabeRegulaereAusdruecke von der Web-Seite der Veranstaltung. Ergänzen Sie die gegebene Klasse entity.Ausdruecke so, dass Sie einen passenden regulären Ausdruck statt des leeren Strings zurückgeben, der die im Kommentar geforderte Eigenschaft erfüllt.

```
package entity;

public class Ausdruecke {

    /* Ausdruck beschreibt alle Strings, die das Wort "Erfolg" beinhalten */
    public String a(){
        return "";
    }

    /* Ausdruck beschreibt alle Strings, die das Wort "Erfolg" mit kleinem
    oder großem ersten Buchstaben beinhalten */
    public String b(){
        return "";
    }

    /* Ausdruck beschreibt alle Strings, die mindestens drei a enthalten */
    public String c(){
        return "";
    }

    /* Ausdruck beschreibt alle Strings, die genau zwei vierstellige
       Ziffernfolgen enthalten, zwischen denen mindestens ein anderes
       Zeichen ist und sich sonst keine weitere Ziffer irgendwo im
       String befindet*/
    public String d(){
        return "";
    }

    /* Ausdruck beschreibt alle Strings in denen genau ein Datum der
       Form Tag.Monat.Jahr steht, dabei sind führende Nullen erlaubt, der
       Tag und der Monat sind ein- oder zweistellig, das Jahr ein- bis
       vierstellig, weitere Ziffern dürfen nicht im String vorkommen.
       Weitere Datumseigenschaften, wie, dass der Tag maximal den Wert
       31 hat, sollen nicht berücksichtigt werden.*/
    public String e(){
        return "";
    }

     /* Ausdruck beschreibt alle Strings, die mindestens zwei a
        und mindestens zwei o in beliebiger Reihenfolge enthalten */
    public String f(){
        return "";
    }
}
```

Nutzen Sie zur Prüfung die gegebenen Testfälle (Details zum Testen später in der Veranstaltung) indem Sie diese mit einem Rechtsklick auf AusrueckeTest.java und „Run As > JUnit Test“ starten.

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Software-Qualität

Sommersemester 2024
2. Aufgabenblatt

Aufgabe 3 (2 Punkte)

Gegeben sei das folgende funktionale Interface.

```
@FunctionalInterface
public interface Calculate {
  public double calc(double pars);
}
```

Nutzen Sie das Interface, um einen Taschenrechner zu programmieren, dabei sollen alle Programmbefehle in einem Objekt vom Typ Map gespeichert werden. Das folgende Programmgerüst mit vorbereiteter Ausgabe finden Sie im Projekt qsAufgabeFunctionalInterface.

```
public class Rechner {
  private double speicher;
  private Map<String, Calculate> funktionen = new LinkedHashMap<>();
```

Die gesamte Implementierung soll in der Klasse Rechner erfolgen. Eine Fehlerkorrektur bei nicht verwendbaren Eingaben muss nicht erfolgen.

Ein Beispieldialog mit der zumindest anzubietenden Funktionalität sieht wie folgt aus, Eingaben sind umrandet:

```
aktueller Wert: 0.0
0: Programmende
1: Reset
2: Addieren
3: Subtrahieren
4: Multiplizieren
5: Dividieren
6: Sinus
7: Cosinus
2
addiere: 42,42
aktueller Wert:
42.42
0: Programmende
1: Reset
2: Addieren
3: Subtrahieren
4: Multiplizieren
5: Dividieren
6: Sinus
7: Cosinus
4
multipliziere: 10
aktueller Wert:
424.20000000000005
0: Programmende
1: Reset
2: Addieren
3: Subtrahieren
4: Multiplizieren
5: Dividieren
6: Sinus
7: Cosinus
6
aktueller Wert:
-0.0848894779071619
0: Programmende
1: Reset
2: Addieren
3: Subtrahieren
4: Multiplizieren
5: Dividieren
6: Sinus
7: Cosinus
0
```

Aufgabe 4 (4 Punkte)

Hinweis: Zum Ergebnis dieser Aufgabe werden Sie auf dem folgenden Aufgabenblatt Tests schreiben, die dann in ILIAS hochladen werden. Auf dem übernächsten Aufgabenblatt werden Sie dann Tests anderer Gruppen nutzen, um Ihre eigene Software zu testen.

In Java 8 wurde eine neue Date Time API ergänzt, die u. a. die Klasse Instant zur Definition eindeutiger Zeitpunkte enthält. Schreiben Sie auf Grundlage dieser Klasse eine eigene Klasse Zeitintervall, die das folgende Interface Intervall implementiert, das von der Veranstaltungsseite als Projekt geladen werden kann.

```
public interface Intervall<T>
```

Ein Intervall ist begrenzt von zwei (Zeit)-Punkten, von denen der erste vor dem zweiten liegen muss, damit es sich um ein korrektes Intervall handelt. Die linke und die rechte Intervallgrenze gehoeren jeweils zum Intervall dazu. Zwei Intervalle koennen in verschiedenen Verhaeltnissen zueinander liegen und gemeinsame Punkte umfassen.

```
boolean istIntervall()
```

Ueberueft, ob es sich um ein korrektes Intervall handelt, liegt der Startpunkt vor dem Endpunkt.

```
boolean istDisjunktMit(Intervall<T> intervall)
```

Ueberprueft, ob die Intervalle getrennte Bereiche beschreiben, dabei soll dies auch der Fall ein, wenn Anfangs- und Endpunkt beziehungsweise andersherum der Intervallgrenzen uebereinstimmen. Sollte es sich bei mindestens einem der Intervalle um kein echtes Intervall handeln, ist das Ergebnis false.

`boolean istEnthaltenIn(Intervall<T> intervall)`
Ueberprueft, ob dieses Intervall (this), in dem uebergebenen Intervall liegt, dabei gilt dies auch, wenn die Intervalle gemeinsame Grenzen haben. Sollte es sich bei mindestens einem der Intervalle um kein echtes Intervall handeln, ist das Ergebnis false.

`Intervall<T> schneidenMit(Intervall<T> intervall)`
Liefert das Intervall, in dem sich beide Intervalle ueberschneiden, sollte dies nicht der Fall oder es nur ein Punkt sein, ist das Ergebnis null. Sollte es sich bei mindestens einem der Intervalle um kein echtes Intervall handeln, ist das Ergebnis null.

`T getStart()`
Gibt den Startpunkt des Intervalls.

`T getEnde()`
Gibt den Endpunkt des Intervalls.

`void setStart(T punkt)`
Setzt den Startpunkt.

`void setEnde(T punkt)`
Setzt den Endpunkt.

Beschäftigen Sie sich gegebenenfalls erst mit Generics in Java, Ihre Klasse sollte wie folgt anfangen: **`public class Zeitintervall implements Intervall<Instant>{`**
Ihre Klasse soll weiterhin die Java-üblichen Methoden **`equals()`** und **`hashCode()`** realisieren, die auch generiert werden dürfen.

Das Überschreiben einer Methode sieht exemplarisch wie folgt aus; durch die zu früheren Java-Varianten kompatible Umsetzung der Generics in Java muss der Parameter leider diesen Typ haben:

```
@Override
public boolean istDisjunktMit(Intervall<Instant> intervall){
```

Bei Rückgabetypen muss dies nicht berücksichtigt werden, dürfte es aber.

```
@Override
public Zeitintervall schneidenMit(Intervall<Instant> intervall){
```