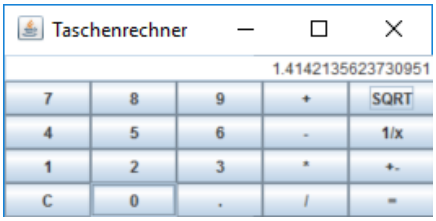
Bitte lösen Sie Aufgaben 24-25 bis zum 14.6. 2020 **und geben Aufgaben 24/25 ab!**

Aufgabe 24

eigener Taschenrechner mit erster Funktionalität



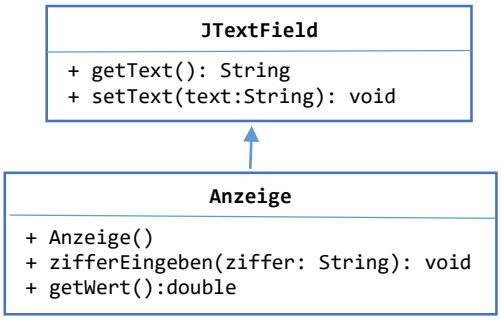
Erstellen Sie zunächst eine Kopie des Pakets *aufgabe14* als ***aufgabe24***.

Schreiben Sie darin die von *JTextField* abgeleitete öffentliche Klasse ***Anzeige***. Im **Konstruktor** rufen Sie setText("0") und *setHorizontalAlignment(RIGHT)* damit die Zahlen rechtsbündig stehen und *setEditable(false)* damit der Taschenrechner nur über die Tasten bedienbar ist. Setzen Sie die Hintergrundfarbe der Anzeige auf Weiß, damit sie besser lesbar ist.

Die Instanzmethode ***zifferEingeben*** hängt *wert* rechts an die Anzeige an: Inhalt mit *getText()* holen, *ziffer* mit + daran hängen und mit *setText()* wieder schreiben).

Die Methode ***getWert***verwendet *Double.parseDouble* um den Text der Anzeige in eine Zahl umzuwandeln. Falls die Anzeige leer ist (`if("".equals(this.getText())`), geben Sie 0.0 zurück.

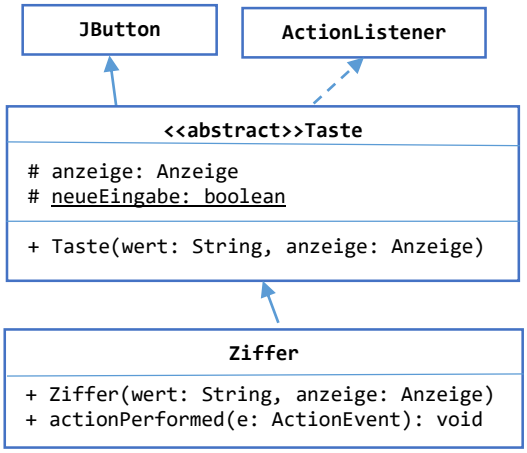
Definieren Sie in ***Taschenrechner*** eine Instanzvariable vom Typ *Anzeige* und ersetzen Sie das *JTextField* durch dieses Objekt.



Die abstrakte öffentliche Klasse ***Taste***, sie wird als Basisklasse für alle Tasten dienen. Der **Konstruktor** gibt *wert* an den Basisklassenkonstruktor weiter, speichert *anzeige* in der Instanzvariable und registriert sich als sein eigener *ActionListener*. **Beachten Sie, dass *neueEingabe* eine Klassenvariable ist!**

Dann schreiben Sie die von *Taste* abgeleitete Klasse ***Ziffer***. Der ***Konstruktor*** gibt beide Parameter an den Konstruktor der Basisklasse weiter. Die Methode ***actionPerformed*** prüft, ob *neueEingabe true* ist, wenn ja löscht sie die Anzeige, indem sie mit *setText* einen leeren Text setzt. Danach setzen Sie *neueEingabe* auf *false* und rufen *anzeige.zifferEingeben(this.getText())*.

Verwenden Sie jetzt Objekte vom Typ *Ziffer* für die Ziffertasten. Beim Drücken einer Ziffertaste erscheint der entsprechende Text in der Ausgabe.



Für den Dezimalpunkt schreiben wir die von *Taste* abgeleitete Klasse ***Punkt***. Der **Konstruktor** hat nur einen Parameter vom Typ Anzeige den er zusätzlich zum Text *"."* an den Konstruktor der Basisklasse weitergibt.
Die Methode ***actionPerformed*** fügt den Punkt nur hinzu wenn noch kein Punkt in der Anzeige steht:

```
if(!anzeige.getText().contains(".")) {
    anzeige.zifferEingeben(".");
}
```

Implementieren Sie als nächstes die Funktionalität für die Taste C, dazu schreiben Sie die von *Taste* abgeleitete Klasse ***Clear***. Der **Konstruktor** hat einen Parameter vom Typ *Anzeige*. Die Methode ***actionPerformed*** setzt den Text *"0"* in die Anzeige.

Die Klasse ***PlusMinus*** wird ebenfalls von *Taste* abgeleitet. Die Methode ***actionPerformed*** holt den Wert aus der Anzeige (*getWert*), multipliziert ihn mit -1 und schreibt ihn mit *setText* zurück, dazu rufen Sie *anzeige.setText(""+wert);*
Abschließend setzen Sie *neueEingabe* auf *true*, damit beim nächsten Drücken einer Ziffertaste eine neue Zahl eingegeben wird.

Die Klasse ***Kehrwert*** mit der Aufschrift *"1/x"* arbeitet analog zu *PlusMinus*. Falls der Wert in der Anzeige 0 ist, macht *actionPerformed* nichts.

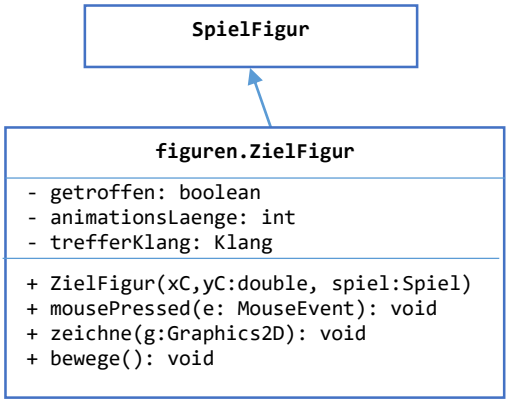
Die letzte Klasse heißt ***Wurzel***: verwenden Sie die Methode *Math.sqrt*, um die Wurzel aus dem Wert in der Anzeige zu ziehen. Falls der Wert in der Anzeige negativ ist, zeigen Sie *"0"* an.

Die Klassenhierarchie ist ein gutes Beispiel für abstrakte Klassen und Vererbung.

Aufgabe 25

Animation

In dieser Aufgabe schreiben Sie die Klasse ***ZielFigur***. Diese Figur reagiert auf Mausklick, indem sie immer kleiner wird und schließlich verschwindet.



Der **Konstruktor** ruft den Basisklassenkonstruktor und legt dabei seine Größe auf 40*40 Pixel fest. Dann setzt er seine Bewegung auf (1,0). Der Wert von *animationsLaenge* wird auf 25 gesetzt.
Die Instanzvariable ***trefferKlang*** wird mit der Datei *treffer.mp3* aus dem online Kurs initialisiert.

Die Methode ***mousePressed*** prüft, ob der Mausklick innerhalb der Figur erfolgt ist. Wenn ja, setzt sie *getroffen* auf *true* und spielt den treffer-Klang ab.

In ***zeichne*** stellen Sie die Figur nur dann dar, wenn *animationsLaenge* größer 0 ist.

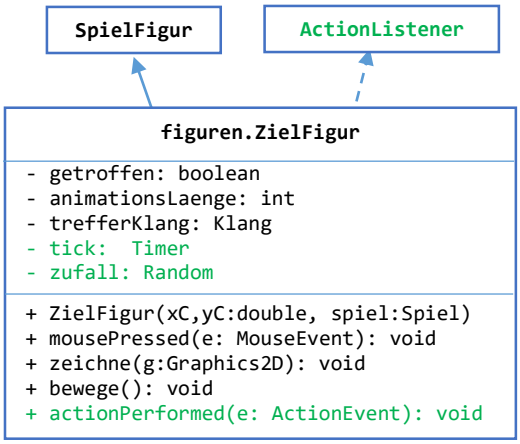
Die Methode ***bewege*** ruft zunächst *super.bewege()*. Falls *geroffen true* ist, teilt sie *witdh* und *height* jeweils durch 1.5 und reduziert den Wert von *animationsLaenge* um 1.

Die getroffene Figur scheint jetzt innerhalb von einer Sekunde (25 Bildern) nach hinten zu verschwinden.

Aufgabe25 wird wie gewohnt von *Spiel* abgeleitet, hat eine Instanzvariable vom Typ *ZielFigur* die initialisiert, bewegt und gezeichnet wird.

Reinkarnation der Spielfigur

Wie wäre es, wenn die Spielfigur, nachdem sie getroffen wurde, an einer zufälligen Stelle links neu erscheint? Dazu brauchen wir zwei alte Bekannte: ein Timer- und ein Random-Objekt.



Beide Instanzvariablen werden im Konstruktor initialisiert. Der Timer bekommt eine Periode von 3000 Millisekunden **wird aber nicht gestartet**.

In ***mousePressed*** wird der Timer gestartet, wenn die Figur getroffen wurde.

In ***actionPerformed*** wird der Timer gestoppt, *animationsLaenge* auf 25, *this.width* und *height* auf 40, *this.x* auf 20 und *this.y* auf 100 sowie *getroffen* auf *false* gesetzt. Jetzt erscheint nach 3 Sekunden die Figur wieder am linken Rand.

Im nächsten Schritt verwenden wir in *actionPerformed* den Zufallsgenerator um:

- Die Anfangsperiode des Timers auf eine zufällige Zahl zwischen *2000* und *6000* zu setzen (*setInitialDelay),*
- Die y-Position der Figur auf einen zufälligen Wert zwischen *10* und *spiel.getHeight()-50* zu setzen.

Auch die Geschwindigkeit könnte dynamisch angepasst werden (*nextDouble*), wobei für die y-Komponente eine kleine positive Zahl in Frage kommt, wenn die Figur in der oberen Hälfte des Spielfeldes ist und umgekehrt.

Packen sie alles, was in *actionPerformed* passiert, in die eigene Methode ***reset()*** und rufen sie in *actionPerformed* und am Ende des Konstruktors *reset*, um sofort zufällige Werte zu haben.

Wenn Sie jetzt noch mehrere Figuren ins Spiel bringen, haben wir schon Moorhuhn schießen für Arme…