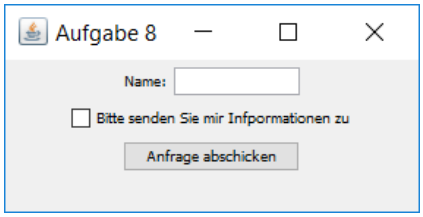
Bitte lösen Sie Aufgaben 8-10 bis zum 03.05.2020 und geben Sie Aufgabe 10 ab!

Aufgabe 8

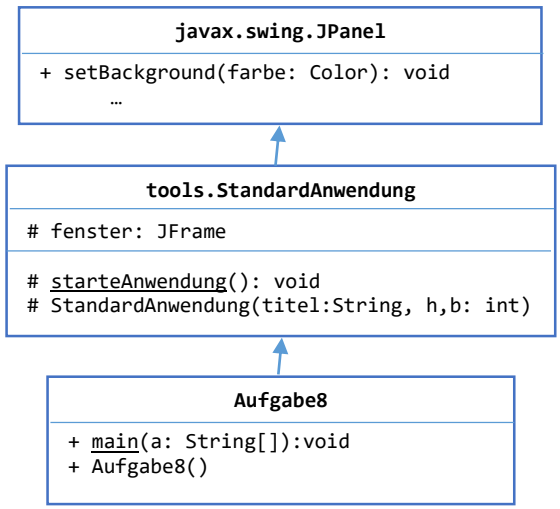
Anordnung verschiedener Swing-Objekte

Schreiben Sie nebenstehende Java Anwendung der Größe 300*150 Pixel, die folgende Objekte darstellt:



1. Einen *Jlabel* mit der Aufschrift „Name“.
2. Ein *JTextField* der Breite 12 Zeichen.
3. Eine *JCheckBox* mit der Aufschrift „Bitte senden Sie mir Informationen zu“.
4. Einen *JButon* mit der Aufschrift „Anfrage abschicken“.

Das Programm hat die gleiche Struktur wie die Aufgaben 6 und 7. Das nebenstehende UML Diagramm zeigt die Hierarchie der Klassen.



Dabei steht **+** für ***public*** und # für ***protected***. Der Pfeil zeigt jeweils auf die Basisklasse. **Unterstrichene** Methoden sind ***static***.

Denken Sie daran, dass der Konstruktor von *Aufgabe8* mit der Anweisung

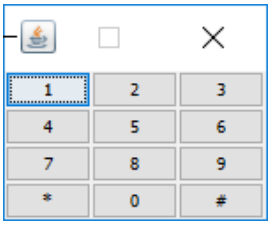
```
super("Aufgabe 8", 300, 150);
```

beginnen muss.

Aufgabe 9

GridLayout, pack

Schreiben Sie die gezeigte Java Anwendung, die das Tastenfeld eines Telefons nachempfindet.



Aufgabe9 wird wie gewohnt von *StandardAnwendung* abgeleitet. Im Konstruktor von *Aufgabe9* legen Sie zunächst einen neuen Layout-Manager fest:

```
this.setLayout(new GridLayout(4,3));
```

der die Elemente in einem Raster von 4 Zeilen und 3 Spalten anordnet.

Dann legen Sie nacheinander mit `this.add(…)` 12 *JButton*-Objekte mit der Aufschrift "1"," 2", "3" u.s.w. auf die Oberfläche.

Anschließend rufen Sie `this.fenster.pack();`

Das Fenster wird jetzt auf die minimale Größe reduziert. Der abschließende Aufruf

```
this.fenster.setResizable(false);
```

verhindert, dass die Größe vom Anwender verändert wird.

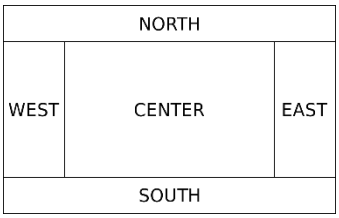
Aufgabe 10

Methode, return-Typ, BorderLayout Abgabe bis 03.05.!

In dieser Aufgabe geht es um einen weiteren wichtigen Layout-Manager: ***BorderLayout***.

Ein *BorderLayout* unterteilt ein Panel in die fünf Bereiche *NORTH, WEST, CENTER, EAST* und *SOUTH*.

Viele Anwendungen (z.B Eclipse, das selbst ja auch in Java programmiert ist) verwenden so ein Schema. Im Bereich NORTH liegen bei Eclipse die Schaltflächen, in WEST der Package Explorer, auf CENTER die Registerkarten mit den Klassen, SOUTH enthält die Konsole und wenn man weitere Fenster einblendet erscheinen sie meist im Bereich EAST.



Wir üben in dieser Aufgabe zwei Dinge:

1. Wir schreiben eine Methode vom return-Typ *JPanel*, die ein *JPanel* Objekt erzeugt, es mit einer bestimmten Farbe belegt und einen Text darauf anzeigt. Diese Methode erspart es uns, die gleichen Programmzeilen mit copy/paste immer wieder zu schreiben.

2. Wir sehen, wie der Layout-Manager *BorderLayout* funktioniert, indem wir farbige JPanel Objekte damit anordnen.

Erstellen Sie zunächst die von *StandardAnwendung* abgeleitete Klasse ***Aufgabe10*** im Paket *aufgaben*.

Fügen Sie zu Aufgabe10 die folgende Instanzmethode hinzu:

```
private JPanel farbigesPanel(Color farbe, String text){
```

Wichtig: achten Sie beim Import darauf, dass *java.awt.Color* verwendet wird!

welche die folgenden Schritte ausführt:

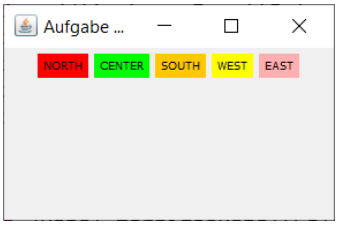
- Sie erzeugt ein neues JPanel Objekt und speichert es in der lokalen Variable *panel*.
- Sie setzt den Hintergrund des Panels auf die Farbe *farbe* (Methode *panel.setBackground*).
- Sie erzeugt ein neues *JLabel* Objekt, dessen Aufschrift der Parameter *text* ist und legt es auf das Panel (panel.add(…)).
- Sie gibt *panel* mit *return* an das rufende Programm zurück.

Im Konstruktor von Aufgabe10 fügen rufen wir diese Methode auf:

```
JPanel p = farbigesPanel(Color.RED, "NORTH");
this.add(p);
```

Oder wenn Sie wollen geht es auch kompakter:

```
this.add(farbigesPanel(Color.RED, "NORTH"));
```

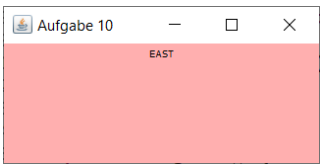


Duplizieren Sie diese Zeile(n) und erzeugen Sie fünf farbige JPanel Objekte mit den Labeln „NORTH“, „CENTER“, „SOUTH“, „WEST“ und „EAST“. Ihre Anwendung sieht jetzt aus wie rechts gezeigt.

Die Methode *farbigesPanel* spart uns offensichtlich einiges an Schreibarbeit und macht unsere Anwendung übersichtlicher. Da wir noch das Standard-Layout (FlowLayout) verwenden, werden die Objekte von links nach rechts in der Reihenfolge angeordnet, in der sie auf die grafische Oberfläche gelegt wurden. Außerdem werden die Panels nur so groß wie nötig dargestellt, also an den Text des Labels angepasst.

Jetzt fügen Sie im Konstruktor unmittelbar hinter super(…) die folgende Zeile ein, um das Layout der Anwendung auf *BorderLayout* zu setzen:

```
this.setLayout(new BorderLayout());
```

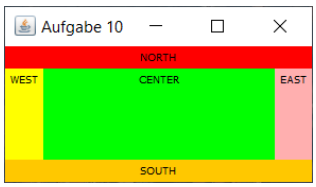


Den Effekt sehen wir rechts: es wird nur noch das zuletzt hinzugefügte Panel dargestellt, das allerdings dehnt sich über das ganze Fenster aus.

Zum Schluss legen wir die fünf Panel explizit auf die Bereiche des BorderLayout:

```
this.add(farbigesPanel(Color.RED, "NORTH"), BorderLayout.NORTH);
this.add(farbigesPanel(Color.GREEN, "CENTER"), BorderLayout.CENTER);

u.s.w.
```



und erhalten das gewünschte Aussehen wie rechts gezeigt.

Sie kennen jetzt die folgenden drei Layout-Manager:

FlowLayout (Standard)**:** Anordnung von links nach rechts, Elemente werden auf minimale Größe geschrumpft,
GridLayout**:** Anordnung im festen Raster, Elemente dehnen sich maximal aus,
BorderLayout**:** Aufteilung auf bis zu fünf Bereiche, Elemente dehnen sich maximal aus

Durch die Kombination dieser Layout-Manager kann man komplexte Benutzerschnittstellen realisieren.