Bitte auf jedem Blatt Ihren Vor- und Nachnamen eintragen. | Please fill in your full name on every page.

.....

Software Engineering I (DiBSE-B-3-SE1-SE1-ILV)

WS 2021

Dipl.-Ing. (FH) Kristian Hasenjäger

DiBSE 2021 v.2.

Klausur

27.Januar 2022
Bearbeitungszeit: 12.45 – 15:15 (150 Minuten

Erlaubte Unterlagen bzw. Hilfsmittel | Aids permitted: -

- Persönlichen PC/Laptop aus der Ferne
- Internet Verbindung
- Open Books

Individuelle, eigenständige Bearbeitung.
Datenaustausch jeglicher Art (Text, Sprache, Dateien) direkt oder indirekt mit Dritten Personen oder deren Ressourcen ist *nicht* zulässig.

1 Zur Beachtung

1. Die Gesamtaufgabe muss mit **`gradlew run`** (auch) von der Kommandozeile ausführbar sein, nachdem diese zuvor mit **`gradle clean`** gesäubert wurde (Achtung vor FileExplorer/Console lock!).
2. Testklassen, wo gefordert, müssen mit **`gradlew test`** ausführbar sein.
3. Die Abgabe ist mit **`gradle mcisrczip`** zu packen:
   a. Das entsprechende Paket wird in **`build/distributions/Assignment.zip`** erzeugt.
   b. Dieses bitte umbenennen in **`Nachnamen_Vorname.zip`** ...
   c. ... und in Sakai Assignments laden.

4. **Beurteilt kann nur Code werden, welcher fehlerfrei kompiliert.**
5. **Auskommentierter Code kann nicht gewertet werden.**
6. **Es gibt 6 Aufgaben mit pro Aufgabe 20 Punkte.**

Die main() Methode:

- *Arbeiten Sie Aufgabe für Aufgabe ab:*
  In der **`main()-`**Methode der Klasse App demonstrieren Sie die Funktionalitäten der erstellten Klassen. Hier bitte mit wenigen Codezeilen lediglich wesentliche Klassen instantieren, konfigurieren und verwenden. Die eigentliche Logik muss in den jeweils verantwortlichen Klassen implementiert sein.

- *Erklären Sie mittels Bildschirmausgaben, den Programmablauf:*
  Relevante Programmschritte sind schlüssig mit **`System.out.println(...)`** zu visualisieren, um Ihren Programmablauf erklärend nachvollziehbar zu machen.

- *Grenzen Sie Aufgaben voneinander ab:*
  Bei Beginn jeder neuen Aufgabe ist mit der Methode **`aufgabenabgrenzer(int beginAufgabe)`** die Aufgabennummer am Bildschirm auszugeben

- *Kommen Sie mit einer Aufgabe nicht weiter?*
  Schreiten Sie weiter zur nächsten und behelfen Sie sich ggf. mit kleinen Patches falls dies nötig sein sollte.

- *Bitte keine vollständigen Ruinen*
  Geben Sie nur Code ab, der auch einen Sinn hat.

Die Aufgaben der Klausur basieren auf einem Klassengerüst (inkl. Gradle build file), welches auf Sakai→SE1 unter Ressources → Klausur → Klausur_SE1_2021.ZIP heruntergeladen und in Ihrer IDE installiert werden sollte.

Folgende Klassenhierarchie für GUI Komponenten ist enthalten:

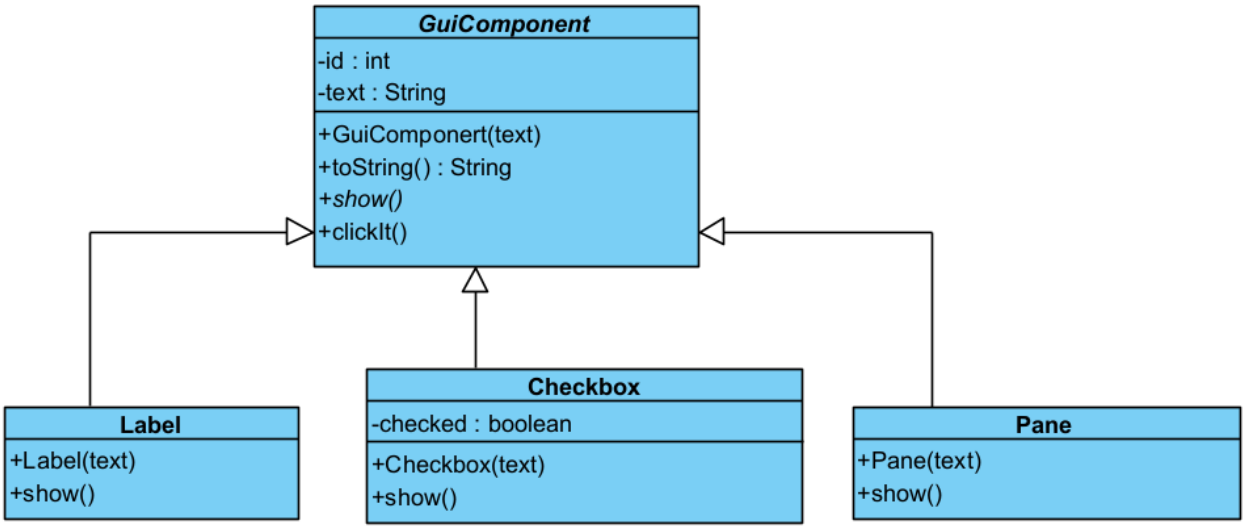

Abbildung: Die drei konkreten GUI-Komponenten erben von der *abstrakten* Superklasse `GuiComponent`.

Methoden:

- `show()` - gibt die Komponente am Bildschirm aus. Die Komponenten Label, Checkbox und Pane werden wie folgt dargestellt:

```
(Ein Label) {0}
(Bitte ankreuzen: [ ]) {0}
/>> Das Pane eins.  ---------------------------------\ {0}
\<< Das Pane eins.  ---------------------------------/
```

  Beachte in geschweiften Klammern die (hier nicht sinnvollen) ID‘s der jew. Komponente

- `toString()` – liefert die Stringdarstellung aller Eigenschaften der Komponente. Wird im Konstruktor aufgerufen, Beispiel:
  `Instanzierung --> id: {0}, Type: CheckBox, text: 'Bitte ankreuzen' , checked: false`

- `clickIt()` – Bislang leere Methode. Wird in Aufgabe 6 ergänzt (Observer Pattern)

Aufgabe 1: Singleton (20 Punkte)

Bei der Instantierung einer jeden `GuiComponent` soll der Eigenschaft `id` vom Typ `int` eine eindeutige Identifikationsnummer zugewiesen werden.
Zuständig hierfür soll eine neue Klasse `IdGenerator` sein, die nach dem Singleton Pattern eindeutige IDs liefert.

In `main()` werden im Bereich zu Aufgabe 1 bereits 3 Komponenten erzeugt, und deren Inhalte dabei am Bildschirm ausgegeben. Kontrollieren Sie sodann, ob `id`'s korrekt erscheinen.
Bitte ändern Sie in main() den bereits vorhandenen Code zu Aufgabe 1 nicht, da wir diesen später noch genau so benötigen.

Aufgabe 2: Factory (20 Punkte)

Eine Klasse `GuiFactory` soll auf Anforderung GUI Komponenten erzeugen. Hierbei kommt das Factory Pattern zur Anwendung.

Spezifikation:

Die Methode `createGuiComponent(String type)` liefert folgende Instanzen:

- type="`LBL`" → `Label`
- type="`CKB`" → `Chechbox`
- type="`PAN`" → `Pane`

Bei Fehlerhafter Parametrisierung wird eine `IllegalArgumentExpeption` geworfen, welche den Fehler genauer beschreibt.

In `main()` im Bereich zu Aufgabe 2 erzeugen Sie drei Komponenten mit gültigen Parametern, sowie einen mit ungültigem Parameter, mitsamt Auffangen eventueller `Exception`s und Ausgabe von Fehlermeldungen.

Aufgabe 3: Unit Tests (20 Punkte)

Die Klasse `GuiFactory` soll eingehend überprüft werden. Erstellen Sie die zugehörige Testklasse und sinnvoller Testabdeckung. Kontrollieren Sie das generierte Testprotokoll.

Aufgabe 4: Strategy (20 Punkte)

Die Darstellung der bereits implementierten Eigenschaft `text` der Klasse `GuiComponent` soll wahlweise

1.) unverändert,
2.) in Großschrift (`s.toUpperCase()`) oder
3.) in Kleinschrift (`s.toLowerCase()`)

erfolgen können. Diese Funktionalität muss mit dem Strategy umgesetzt werden.

Im main() Bereich zu Aufgabe 4 demonstrieren Sie ihre Implementierung, indem sie die bereits in Aufgabe 1 instantierte Komponente lbl1 und ckb1 in den drei Ausgabearten darstellen.

Aufgabe 5: Composite (20 Punkte)

Die Komponente Pane soll nach dem Composite Pattern beliebige Komponenten aufnehmen können. Ergänzen das hierzu Nötige.

Die Bildschirmausgabe von im Pane enthaltenen Komponenten soll derart erfolgen, dass diese *vor* und *nach* der Ausgabe (mittels show()) durch die (bereits implementierten) horizontalen Linien eingefasst werden.

Im main() Bereich zu Aufgabe 5 demonstrieren Sie 1.) *Erzeugung* und 2.) *Bildschirmausgabe* das geschachtelte Gui in folgender Baumstruktur mit jeweils angegebenen Factory-Typen (vgl. Aufgabe 2):

1. PAN
   1.1. PAN
      1.1.1. CKB
   1.2. PAN
      1.2.1. CKB
      1.2.2. LBL

Erläuterung: Panel 1 enthält die zwei Panels 1.1 und 1.2. Diese enthalten wiederum die angegebenen weiteren Komponenten.

Aufgabe 6: Observer (20 Punkte)

- In unserem GUI soll fortan jede Komponente auf „Mausklick“ auf beliebige Komponenten reagieren können.
- Jede Komponente kann *Observierer* und/oder auch *observiertes Subjekt gleichermaßen* sein.
- Eine angeklickte Instanz wird sodann als Ereignisquelle an alle *registrierten* Oberserver übermittelt.

- Die Klasse `GuiComponent` enthält bereits eine noch leere Methode `clickIt()`, die einen Mausklick auf die jeweilige Komponente simulieren möge.
- Empfängt ein Observer ein Ereignis, so soll die Eigenschaft `text` (über Klasse `GuiComponent` zugänglich) über die geklickte Komponente informieren, in etwa so: “`Element mit ID=“+sourceComponent.getId()+ “ wurde geklickt`“ erhalten.

Im `main()` Bereich zu Aufgabe 6 ...

1.) ... registrieren Sie Label `lbl1` als Observer bei Checkbox `ckb1` und `ckb2`.
2.) ... geben Sie *vor dem simulierten Mausklick* mit `show()` den Inhalt der Komponente `ckb1` am Bildschirm aus.
3.) ... simulieren Sie einen Mausklick durch Aufruf der Methoden `ckb1.clickiIt()`

Ende der Klausur ☺