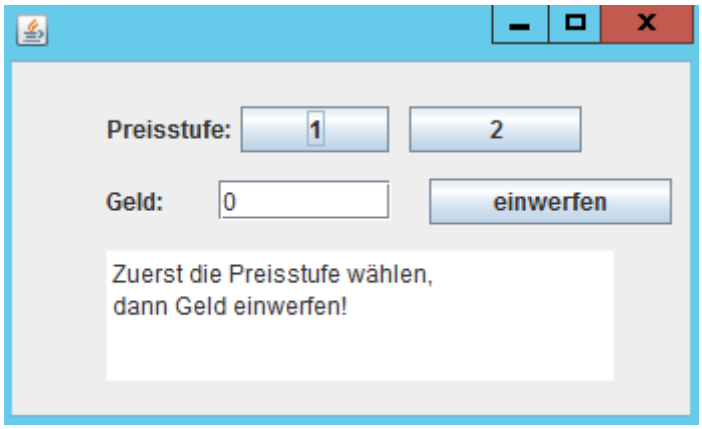
Window-Builder Cheat Sheet



Hier wird im Hauruck-Verfahren vorgestellt, wie man für den Fahrkartenautomat mit dem Window-Builder eine graphische Oberfläche (graphical user interface = GUI) erstellen und mit dem Fahrkartenautomat verknüpfen kann.

Die **Grundidee** dabei ist, dass, die GUI völlig „dumm“ ist, sie enthält überhaupt keine Programmlogik. Die Programmlogik wird nur in Fahrkartenautomat realisiert. Das ist ein Standard-Design, was zu beachten sich wirklich lohnt, weil man so die Entwicklung der Logik von der Entwicklung der Oberfläche trennen kann.

Zwischen Fahrkartenautomat und FahrkartenautomatGUI besteht eine 1:1-Beziehung, wobei der Fahrkartenautomat die FahrkartenautomatGUI besitzt (denn eine GUI ohne Fahrkartenautomat ist sinnlos).

Die GUI erstellen:

a) **Das Package `wiederholungEF` anklicken**

b) **Einen `JFrame` erstellen und zur Bearbeitung öffnen:**

1. File → New → Other → Window Builder → **`JFrame`**
2. Name: `FahrkartenautomatGUI`
3. In der Mitte **`Design`** (statt Source) anklicken.

b) Das richtige Layout wählen

1. Unter Layouts **`AbsoluteLayout`** anklicken, dann die Bühne anklicken. Damit ist das Layout festgelegt.

c) Die Components hinzufügen:

1. Wichtig sind folgende Components:
   - `JLabel`: ein „Schild“, nicht editierbar.
   - `JButton`: ein Button
   - `JTextField`: ein einzeiliges editierbares Textfeld
   - `JTextArea`: ein mehrzeiliges editierbares Textfeld
2. So fügt man hinzu:
   1. Die jeweilige Component anklicken, dann auf die Bühne klicken und ggf. verschieben und vergrößern. **Testen!**
   2. WICHTIG: unter **Properties** bei **Variable** einen vernünftigen Variablennamen angeben! Am besten beginnt der Variablenname mit dem Typ, z.B. `buttonEinwerfen`

d) Die `ActionListener` für die Buttons hinzufügen:

1. Auf der Bühne einen Button anklicken, und dann:
   rechte Maustaste → **Add Event Handler → Action → actionPerformed**
2. Es öffnet sich der Source-Code und die Maus steht an der Stelle
   ```
   public void actionPerformed(ActionEvent arg0) {
   }
   ```
3. In die geschweiften Klammern direkt eine Konsolenausgabe als Hinweis reinschreiben, z.B.: `System.out.println("Einwerfen Button angeklickt");`

Von jetzt an geht es im Bereich <u>Source</u> weiter!

e) Alle Components, bei denen es notwendig ist, zu Attributen machen.

1. Notwendig ist das bei allen Components, aus denen man Informationen auslesen möchte bzw. in die man Informationen reinschreiben möchte, d.h. z.B. für `JTextField` und `JTextArea`. Nicht notwendig ist das für die Buttons.

2. Oben in der Klasse nachschauen, welche Components noch nicht bei den Attributen erscheinen.

3. Für die fehlenden Components muss man die Stelle suchen, wo sie als lokale Variable deklariert werden.

4. Die Deklaration wegnehmen, z.B.:
~~`JTextField`~~ `geldTextField = ...`
Der Variablenname ist jetzt rot unterschlängelt.

5. Links die Glühbirne anklicken → **Create Field**
(„Field“ übersetzt sich mit „Attribut“.)

f) get-Methoden erstellen, dabei ggf. in den richtigen Datentyp konvertieren:
Das braucht man, um das eingeworfene Geld auszulesen

```
public double getEingeworfenesGeld(){
  //Text auslesen
  String geldString = textFieldGeld.getText();
  // in double konvertieren
  double geldDouble = Double.parseDouble(geldString);
  // zurückgeben
  return geldDouble;
}
```

g) set-Methoden erstellen:
braucht man, um den Infokasten mit Text zu versorgen.

```
public void setInfoText(String pText){
  textAreaInfo.setText(pText);
}
```

Damit ist die Oberfläche fertig!

Jetzt muss FahrkartenautomatGUI.java mit der Klasse Fahrkartenautomat.java verknüpft werden. Zwischen den beiden besteht eine 1:1-Beziehung, was die Sache einfach macht.

Die 1:1-Beziehung zwischen FahrkartenautomatGUI und Fahrkartenautomat herstellen:

h) Attribute für die jeweils andere Klasse eintragen:

1. In Fahrkartenautomat:

```
private FahrkartenautomatGUI fahrkartenautomatGUI;
```

In FahrkartenautomatGUI:

```
private Fahrkartenautomat fahrkartenautomat;
```

i) **Die beiden Klassen miteinander „bekannt“ machen.**
Da der Fahrkartenautomat die main-Methode hat, erzeugt er die GUI und macht sie sichtbar.

1. Im Konstruktor von Fahrkartenautomat ergänzen:

```
// die GUI erzeugen
fahrkartenautomatGUI = new FahrkartenautomatGUI();
// die GUI sichtbar machen
fahrkartenautomatGUI.setVisible(true);
// der GUI mitteilen, welcher Automat zu ihr gehört:
// „this“ bedeutet so viel wie „ich“
fahrkartenautomatGUI.setAutomat(this);
```

Die letzte Zeile wird rot unterschlängelt, weil es die Methode setAutomat(..) in der Klasse FahrkartenautomatGUI noch gar nicht gibt!

2. Die Methode `setAutomat()` in FahrkartenautomatGUI ergänzen:
- bei dem Fehler (s.o.) die **Glühbirne** anklicken
- **Create Method** auswählen und Enter
- In der neu entstandenen Methode ergänzen:

```
this.fahrkartenautomat = fahrkartenautomat;
```

Jetzt „kennen“ sich der Fahrkartenautomat und die FahrkartenautomatGUI, d.h. sie können jetzt Methoden für die jeweils andere Klasse aufrufen. In FahrkartenautomatGUI macht man das immer, wenn ein Button geklickt wurde, im Fahrkartenautomat immer dann, wenn etwas an die Konsole ausgegeben wird.

j) Die Programmlogik in FahrkartenautomatGUI ergänzen (Beispiel):
z.B. für den buttonEinwerfen:

```
public void actionPerformed(ActionEvent arg0){
   double geld = getEingeworfenesGeld();
   fahrkartenautomat.geldEinwerfen(geld);
}
```

k) Die Programmlogik in Fahrkartenautomat ergänzen (Beispiel):
z.B. in der Methode geldEinwerfen:

```
public void geldEinwerfen(double pGeld){
   ...
   fahrkartenautomatGUI.setInfoText
         („Schon eingeworfen: “+eingeworfen);
```