

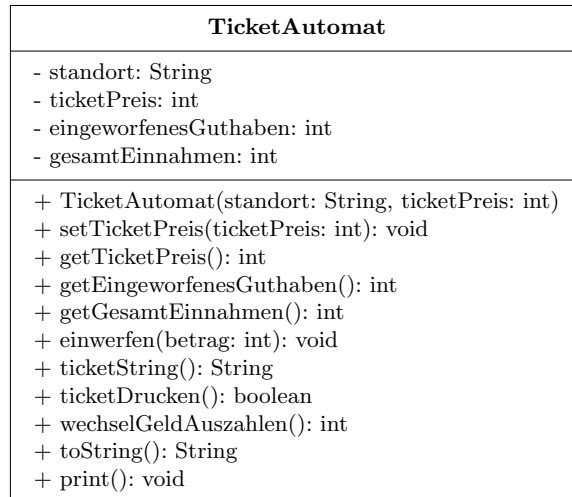
# Übungsblatt zu Klassen, Testklassen und Berechnungen

Klasse: Ticket-Automat

SPENGERGASSE 

## Aufgabe 1 Implementierung der Klasse *TicketAutomat*

Die Klasse `TicketAutomat` modelliert einen Fahrschein-Automaten, die Eigenschaften und Methoden sind durch folgendes UML-Diagramm gegeben:



Mittels der Methode `einwerfen` wird Geld in den Automaten eingeworfen. Sobald der eingeworfene Betrag dem Ticketpreis entspricht, kann ein Ticket gedruckt werden (Methode `ticketDrucken`). Das Ticket soll wie folgt formatiert sein:

#####

FAHRSCHEIN

Standort: Westbahnhof  
Preis: 3 Euro

Gute Fahrt!

#####

Implementieren Sie die gegebenen Methoden mit der entsprechenden Programmlogik und aller sinnvoller Parameterprüfungen! Wurde noch zu wenig Geld eingeworfen um ein Ticket zu drucken, soll die Methode `ticketDrucken` ausgeben, wieviel Geld noch einzuwerfen ist. Ist genügend eingeworfenes Guthaben vorhanden, so kann das Ticket gedruckt werden; das verbleibende eingeworfene Guthaben kann dann entweder für etwaige weitere Tickets verwendet werden, oder als Wechselgeld ausgezahlt werden.

*Hinweis:* Da es für den Standort keine Set-Methode gibt, können Sie diesen (inklusive Parameterprüfung) direkt im Konstruktor setzen. Der Rückgabewert bei `ticketDrucken` gibt an, ob ein Ticket gedruckt wurde oder nicht. Die Methode verwendet die Methode `ticketString` um das Ticket auf die Konsole zu "drucken".

*Beispiele:*

Ticketpreis 30c

```
Einwerfen 70c ==> Wechselgeld Auszahlen -> 70c (eingeworfen: 0c)
Einwerfen 20c ==> TicketDrucken -> false (eingeworfen: 20c)
Einwerfen 40c ==> TicketDrucken -> true (eingeworfen: 10c, gesamt 30c)
                ==> Wechselgeld Auszahlen -> 10c (eingeworfen 0c, gesamt 30c)

(gesamt: 30c)
Einwerfen 100c ==> TicketDrucken -> true (eingeworfen: 70c, gesamt: 60c)
```

```
==> TicketDrucken -> true (eingeworfen: 40c, gesamt: 90c)
==> Wechselgeld Auszahlen -> 40c (eingeworfen: 0c, gesamt: 90c)
```

---