4BKIF WMC Kolloquium Sommersemester

SPENGERGASSE
ausbildung mit zukunft

10. Februar 2024 / Themen: Javascript, DOM, Event Handler

Ticketautomat

Die Angabe beinhaltet einen Ticketautomaten. Bei dem Projekt fehlt noch die (Javascript-) Funktionalität, diese ist zu implementieren. Beachten Sie, dass die meisten Änderungen des Zustandes (Ziel ändern, Geld einwerfen, Ticket drucken, usw.) Änderungen in anderen Feldern bewirken.

Der Automat hat 2 Bargeld-Orte:

1. Bar- und Wechselgeldreserve .. Dies ist der Speicher für alles bisher eingenommene Geld und nötiges Wechselgeld. Wird initial mit zumindest 100€ befüllt. Es ist sicherzustellen, dass dies gewährleistet ist.
2. Das Guthaben (”credit”) des aktuellen Kunden.

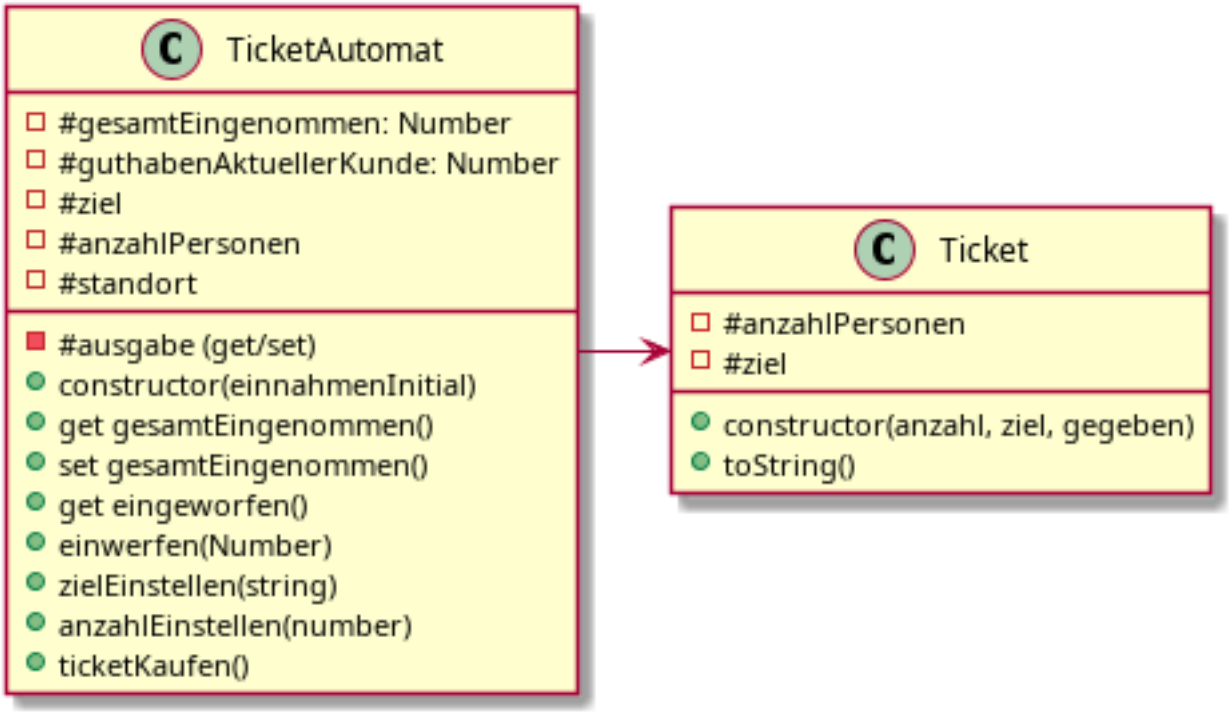
Die möglichen Reiseziele (Dropdown) sind der Datei ”ziele_preise.json” zu entnehmen.

Die Implementierung des ”State” ist als Javascript-Klasse gewünscht.

- Im constructor gibt übergibt man, wieviel Geld ürsprünglich im Automaten ist (in €), damit der Automat Restgeld geben kann.
- Man kann Geld `einwerfen()`, dies erhöht sich das Feld ”Guthaben” um den eingeworfenen Betrag.
- Man kann das Fahrziel einstellen, in einem `select` Feld.
- Man kann die Anzahl der Fahrgäste einstellen (1-10)
- man kann den Button ”Ticket kaufen” drücken, dann wird ein Ticket oder eine Fehlermeldung in die TicketAusgabe gedruckt.

Klassen Ticket und TicketAutomat

Erstelle für diese Aufgabe die folgenden 2 Klassen:



Im `constructor(initialBetrag)` sollen folgende Dinge passieren:

- `this.#gesamtEingenommen` soll auf den gegebenen Wert gesetzt werden, indem `setgesamtEingenommen()` aufgerufen wird.

- this.eingeworfen auf 0.
- this.ziel auf `undefined`

Weiters:

`zielEinstellen()` soll nur eines der oben genannten Ziele in die private Variable `ziel` speichern können.

`anzahlEinstellen()` soll die Variable `#anzahlPersonen` setzen, sofern der übergebene Wert gültig ist.

`ticketKaufen()` soll ein neues Ticket erstellen mit `let ticket = new Ticket (ziel, anzahlPersonen, gegeben)`, soferne der eingeworfene Betrag ausreicht. Im Fehlerfall soll eine Exception geworfen werden.Die Ausgabe des Tickets möge in das vorgesehene Textfeld erfolgen. Im erfolgreichen Fall wird das Feld `#gesamtEingenommen` des Automaten um den Ticketpreis erhöht.

Beispiel für ein Ticket

```
==============================
=== Fahrkarte nach Salzburg ===
==============================
Einzelpreis: € 30.-
Anzahl der Fahrgäste: 3
==============================
Summe: € 90.-
==============================
gegeben: € 100.-
Restgeld: € 10,-
==============================
```

Eine Instanz der oben erstellten Klasse dient als ”State” für die zu erstellende Applikation. Die von Ihnen implementierten Methoden der Klasse dienen als State- Accessors bzw. State-Modifiers.

Elemente der Benutzeroberfläche:

- einwerfen (input type=numer ... submit)
- ziel einstellen (option / select)
- anzahl Personen einstellen
- Anzeige Fahrpreis (ändert sich bei Ziel Änderung oder Anzahl Änderung)
- Anzeige des Guthabens (ändert sich durch einwerfen)
- Ausgabefeld des Tickets, dort kann auch angezeigt werden ”Es fehlen noch XX € damit ich das Ticket drucken kann”.
- Anzeige der gesamten Einnahmen des Automatens, ändert sich bei jedem Ticketkauf (`#gesamtEingenommen`)
- button ”Ticket Kaufen” (wenn das Geld reicht wird Ticket gedruckt und die `#gesamtEingenommen` vergrößern sich entsprechend)

gutes Gelingen!

wichtig für die Beurteilung

- es sollen keine JS-Errors auf die Konsole durchkommen
- Implementierung der Klassen Ticket und TicketAutomat
- Keine inkonsistenten Zustände in der Anzeige

Referenz: 7 Punkte

1.APPLICATION STATE OBJECT

- Holds the state of the application
- This is the single source of truth for the application state
- Functions that allow us to get and set the state
- Functions to interact with the state

2.DOM Node Refs

- Static references to DOM nodes needed after the start of the application

3.DOM Node Creation Fn's

- Dynamic creation of DOM nodes needed upon user interaction
- Here we will possibly create a function that will create a new item

4.RENDER FN

- These functions will render the application state to the DOM
- IMPORTANT TAKEAWAY: The state drives the UI, any state change should trigger a re-render of the UI

5.EVENT HANDLERS

- These functions handle user interaction e.g. button clicks, key presses etc.
- These functions will call the state mutators and then call the render function
- The naming convention for the event handlers is `on<Event>`
- Here we will create a functions that will handle e.g. a ”click” event on a button.

6.INIT BINDINGS

- These are the initial bindings of the event handlers, i.e. register the handlers of Pt. 5 with the DOM Node Refs of Pt. 2

7.INITIAL RENDER

- Here will call the render function (Pt. 4) to render the initial state of the application