

# Test 1 Gebruikersinterfaces

31 maart 2020

## Inleiding

Open het gegeven project in IntelliJ. *Vul voornaam en familienaam in in het bestand naam.txt.* Als het opstarten van het project niet loopt zoals verwacht: tips op het laatste blad van deze opgave.

In deze test implementeren we het spel 'Memory' voor één speler.

Laat alle gegeven code ongewijzigd (tenzij gevraagd).

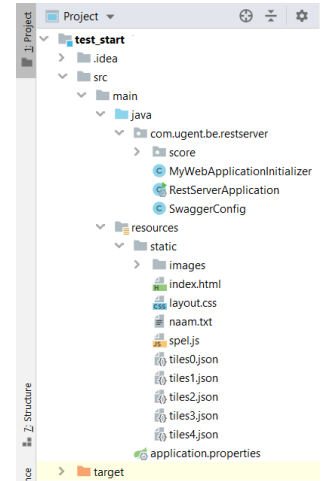
Jouw aanvullingen komen in het css-bestand en het JavaScript-bestand:

layout.css

spel.js

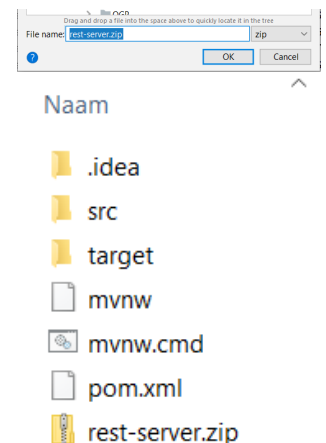
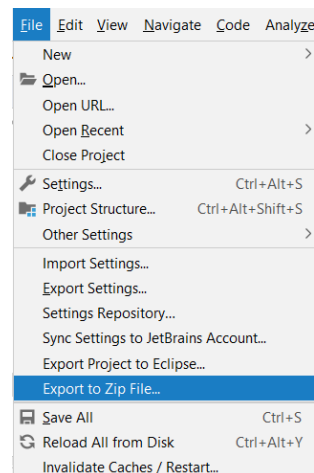
De link naar deze bestanden werd al voorzien in het bestand index.html, daar hoeft je niets meer aan te passen.

Als de test afgelopen is, dan dien je in Indianio zoals hieronder uitgelegd.

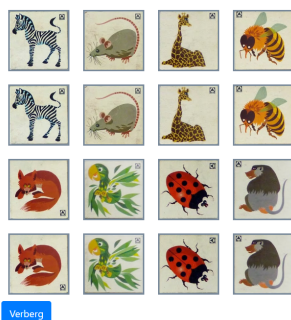


## Indienen in indianio

1. Vul het bestand `naam.txt` verder aan met de status van je oplossing.
2. Maak een zip van je project in IntelliJ: kies File - Export to Zip File... en klik onmiddellijk op OK. De voorgestelde naam is waarschijnlijk `rest-server`, laat dit zo. Nu vind je het `rest-server.zip`-bestand via een verkenner in de map van het project.
3. Surf naar [indiano.ugent.be](http://indiano.ugent.be). Daar vind je een *indienmogelijkheid*. We verwachten hier het `rest-server.zip`-bestand. Je zal een overzicht van de geschreven code te zien krijgen in *indiano*. Als dat niet zo is, moet je opnieuw (aankomen en) uploaden.



## Spelregels van Memory



Memory wordt gespeeld met 8 sets van 2 gelijke kaarten. In het begin worden de 16 kaarten gedekt op tafel gelegd. De speler draait 2 kaarten om. Als die gelijke afbeeldingen tonen, dan blijven ze open liggen (dit noemen we een 'verworven set kaarten'). Anders worden ze door de speler weer gedekt gelegd (in onze versie met de

knop **Verberg**).

Het spel is afgelopen als alle kaarten open liggen. De score van de speler is het aantal keer dat hij een kaart open gelegd heeft. De beste score is dus 16, al wat groter is is 'slechter'.

# 1 Spel (zonder server)

## 1.1 Voor je begint met coderen

De kaarten hebben elk een nummer tussen 0 en 7, wat hun afbeelding bepaalt. Denk na over het spelverloop:

1. hoe houd je bij of de speler nog 1 (of zelfs 2) kaarten mag open leggen?
2. hoe weet je dat de knop **Verberg** actie mag ondernemen, nl. dat hij 2 ongelijke kaarten terug toe mag leggen?
3. hoe weet je welke kaarten dat zijn?
4. hoe zie je dat het spel afgelopen is?

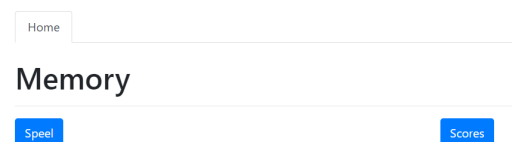
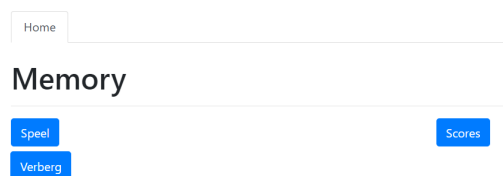
In het JavaScriptbestand vind je enkele tips die hierbij kunnen helpen (niet verplicht, eigen ideeën zijn ook goed):

```
let eerste_kaart = null;
let tweede_kaart = null;
let dubbels_gevonden = 0;
let score = 0; // verhoogt (+1) elke keer dat een kaart wordt omgedraaid
```

## 1.2 Webpagina bij de start


Start van het gegeven project. Als je het project laat runnen, dan vind je op [localhost:8080](http://localhost:8080) de webpagina die hieronder links afgebeeld is. (Runnen gaat niet? Raadpleeg de laatste pagina. In hoge nood, contacteer de lesgevers via Slack.)

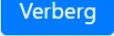
Ga eerst in de css-code aangeven dat alle elementen met de klasse **onzichtbaar** ook effectief onzichtbaar moeten zijn. (Gebruik de display-eigenschap.) Dan ziet de webpagina bij de start er dus zo uit als rechts.



*scroll door naar de volgende blz*

## 1.3 Kaarten toevoegen aan de webpagina

Als de gebruiker op de knop  klikt, dan gebeuren er twee zaken:

1. de knop  verschijnt weer (haal de klasse `onzichtbaar` uit de `classList`), en
2. er verschijnen 16 gedekte kaarten.

De 16 kaarten zijn geordend in een table:

```
<table>
  <tr>
    <td><img ...></td>
    <td><img ...></td>
    <td><img ...></td>
    <td><img ...></td>
  </tr>
  <tr>
    ...
  </tr>
  ...
</table>
```

Voeg de kaarten toe met het DOM-principe, dus met DOM-elementen en niet met html-tekst.

Bij de start is de afbeelding van elke kaart gelijk aan `gedekt.png` (zie in de map `images`).

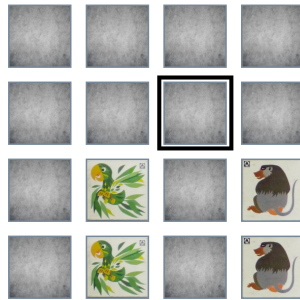
De kaarten maak je 100px breed en hoog, de margin zet je op 10px. Het nummer van de ongedekte afbeelding (0-7) bewaar je in een custom data-attribuut. Deze nummers hoeft je zelf niet toe te kennen, ze worden opgehaald vanuit het bestand `tiles0.json`. Dit bestand bevat deze code:

```
[
  [0,1,2,3],
  [0,1,2,3],
  [4,5,6,7],
  [4,5,6,7]
]
```

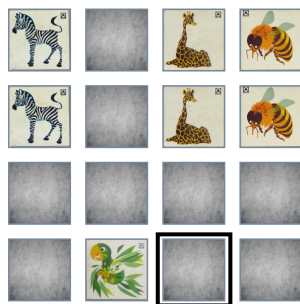
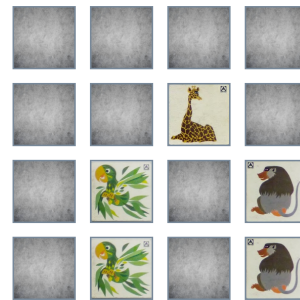
Het is dus een itereerbare collectie van getallen, en bepaalt waar de afbeeldingen komen te liggen. In dit geval is het niet erg random: gelijke kaarten liggen onder elkaar. Dit is echter ideaal als testcase; de andere `tilesY.json`-bestanden bevatten wel random gegevens (maar heb je in deze test niet nodig).

## 1.4 Het spel zelf: kaarten open

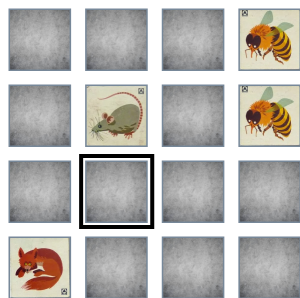
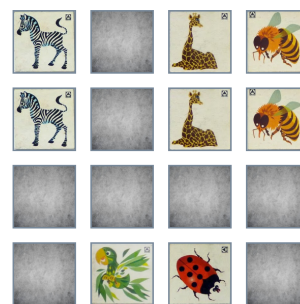
Hieronder wordt er getoond wat er gebeurt als er op een kaart geklikt wordt. Links zie je de startsituatie, de kaart die aangeklikt wordt is in het screenshot aangeduid met een zwarte kader (die zal in het spel *niet* verschijnen). Rechts zie je het resultaat na het klikken. Tussen beide afbeeldingen vind je uitleg over de actie. Implementeer deze werking.



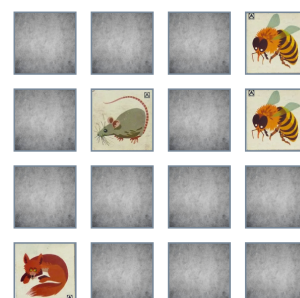
Als er nog geen kaart open lag (buiten de verworven sets van dubbels), dan wordt de afbeelding `imgX.png` getoond (zie map `images`, de X vervang je uiteraard door het juiste cijfer).



Als er al een kaart open ligt, wordt deze tweede kaart ook geopend. Indien beide kaarten gelijk zijn, blijven ze open liggen (de speler zal weer 2 nieuwe kaarten kunnen openen). Anders zal de speler ze straks weer kunnen verbergen.

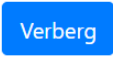


Als er al 2 kaarten open liggen (bovenop de reeds verworven sets van dubbels), gebeurt er niets.



**Tip bij het uitproberen van het spel.** Tijdens het uitproberen van het spel op de webpagina, kan het gebeuren dat het aanklikken van de vier kaarten uiterst rechts op het speelbord niet lukt (de andere 3 kolommen wel). Dan staat allicht het inspect-venster open; als je dit sluit zal het aanklikken wel lukken.

## 1.5 Het spel zelf: kaarten toe

Als er twee ongelijke kaarten open liggen (bovenop de reeds verworven sets van dubbels), dan moet de speler op de -knop klikken om deze weer om te draaien. Implementeer deze werking.

Afhankelijk van de methode waarop je dit doet (let op, er zijn efficiënte en minder efficiënte methodes), kan je eventueel onderstaande tip gebruiken.

Om na te gaan of een kaart gedekt is, kan je controleren of zijn afbeeldingsbron gelijk is aan `"gedekt.png"`. Het is echter veiliger om na te gaan of zijn afbeeldingsbron de string `"gedekt.png"` bevat; gebruik de methode `includes()` van de klasse `String`.

## 1.6 Speleinde

Indien de speler gewonnen is, worden zijn kaarten na 2 seconden vervangen door de afbeelding `bravo.png`.

1. Maak de tabel leeg en voeg vlak na de tabel (in dezelfde `div`-tag) een canvas toe van 430 op 430 pixels
2. Plaats de afbeelding `hoera.png` op het canvas (ook 430 op 430 pixels).
3. Bovenop die afbeelding komt nog een tekst: 24px hoog, lettertype Verdana, en de boodschap "u opende ... sets tevergeefs". Een set die 'vergeefs' geopend werd, is een set van ongelijke kaarten die nadien weer verborgen moesten worden. Als de speler dus alle 16 kaarten meteen goed omdraaide, komt er "u opende 0 sets tevergeefs".



## 2 Scores op de server

In het tweede deel van de opdracht is het de bedoeling om de score op te slaan via een REST API.

### 2.1 Voorbereiding

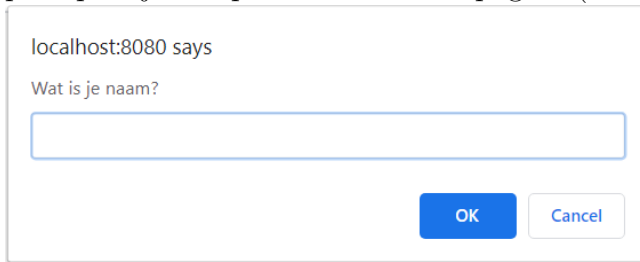
De data zal verpakt zitten in een JSON object en zal er als volgt uitzien:

```
{
  "name": "string",
  "score": 0,
  "uuid": "string"
}
```

Je kan via swagger de API bekijken en uitproberen: <http://localhost:8080/swagger-ui.html>.

Naast de score wordt ook de naam en een ID opgeslagen. De ID wordt door de back-end applicatie gegenereerd wanneer er een nieuw score-object wordt aangemaakt. De naam van de speler moeten we wel meegeven. Zorg ervoor dat de speler zijn naam kan ingeven via een

prompt bij het openen van de webpagina (zie screenshot hieronder).

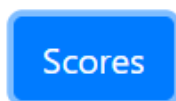


## 2.2 Opslaan van de score

Bij het speleinde moet de score worden verzonden naar de server. Gebruik hiervoor de `fetch` API. Voordat je begint met programmeren beantwoord je best deze vragen:

1. Welke HTTP methode moet gebruikt worden?
2. Hoe ziet de body van de HTTP request eruit?
3. Welk path moet ik meegeven?

## 2.3 Opvragen van de scores: plain text



arne 22  
bert 26  
cyriel 16

Wanneer op de `Scores`-knop wordt geklikt, moeten alle scores opgehaald worden en getoond worden. We zullen hiervoor gewoon 'plain text' aanmaken, wie tijd over heeft kan voor de uitbreiding gaan (volgende paragraaf).

1. In het html-bestand vind je een `div`-tag met `id idScoresPlainText`. Je zorgt dat de `innerText` van deze (lege) tag vervangen wordt door een `String` die alle namen en scores aan elkaar plakt (gebruik `"\n"` voor *new line*).
2. Voor het ophalen van de scores vanop de server beantwoord je opnieuw de vragen van paragraaf 2.2.
3. Zorg voor een update van de tekst op het einde van het spel. Doe dit enkel als de operatie om een score te verzenden naar de server succesvol was.

Het resultaat vind je hiernaast.

Wat hieronder staat levert bonuspunten op. Enkel te doen bij tijd over.

## 2.4 Uitbreiding A: scoretabel met opmaak

```
<div id="idScoresPlainText">  
</div>
```

Vervang in het html-bestand de code door een tabel met bootstrap-opmaak. Zorg in het html-bestand ook voor twee kolommen met elk een hoofding (`Naam` en `Score`). Met JavaScript voeg je een rij toe aan de tabel voor elke score die op de server gevonden

wordt. Je krijgt dit als resultaat (allicht met meer rijen):

Scores

Naam	Score
els	16

## 2.5 Uitbreiding B: zoeken in de tabel

Voeg een inputveld en een knop toe aan je webpagina zodat de tabel gefilterd kan worden op naam.

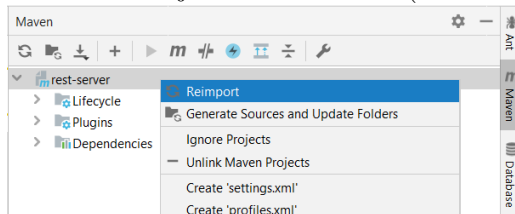
# Tips als het project niet runt zoals het hoort

## RestServerApplication verschijnt niet naast groene run-pijl

Indien er foutmeldingen verschijnen zoals

```
Error(3, 52) java: package org.springframework.beans.factory.annotation does not exist
Error(4, 46) java: package org.springframework.context.annotation does not exist
Error(5, 46) java: package org.springframework.context.annotation does not exist
Error(6, 78) java: package org.springframework.security.config.annotation.authentication.builders does not exist
Error(7, 75) java: package org.springframework.security.config.annotation.method.configuration does not exist
Error(8, 67) java: package org.springframework.security.config.annotation.web.builders does not exist
Error(9, 67) java: package org.springframework.security.config.annotation.web.builders does not exist
Error(10, 72) java: package org.springframework.security.config.annotation.web.configuration does not exist
```

dan herlaad je de rest-server (rechtermuisklik; dit duurt even):



Heb je later nog last met het starten van de server, sluit dan het project en heropen het.

## Poort al in gebruik

Indien poort 8080 al in gebruik is door een ander proces, krijg je deze foutmelding:

```
*****
APPLICATION FAILED TO START
*****
```

Description:

The Tomcat connector configured to listen on port 8080 failed to start. The port may already be in use or the connector may be misconfigured.

Action:

Verify the connector's configuration, identify and stop any process that's listening on port 8080, or configure this application to listen on another port.

Vraag via de command line (cmd) informatie over poort 8080 op

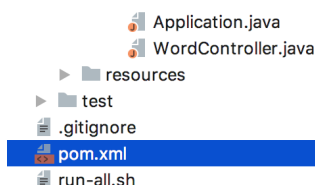
```
netstat -aon | find "8080"
```

```
C:\Users\leen>netstat -aon | find "8080"
TCP 0.0.0.0:8080 0.0.0.0:0 LISTENING 18120
TCP [::]:8080 [::]:0 LISTENING 18120
```

en leid daaruit het id af van het proces dat op deze poort loopt (zie laatste getal). Open dan de task manager en controleer in het tabblad *Details* met welk proces het gevonden 'Process Id' overeenkomt. In de task manager kan je dit proces stoppen (zoek eventueel eerst op of het een belangrijk proces zou kunnen zijn).

## 'Add Configuration' en rode cirkeltjes

Als er in plaats van  de boodschap **Add Configuration** staat, en een aantal java-bestanden zijn gemarkeerd met kleine (oranje)rode cirkeltjes



dan klik je rechts op het bestand `pom.xml` en kies je voor **Add as Maven Project** (onderaan). Zie ook [stackoverflow](https://stackoverflow.com), of gebruik de zoektermen `intellij red circle filename`.



## De nieuwe code wordt niet herkend in de browser

Als je nieuwe code geschreven hebt maar het effect daarvan op de website blijft uit, kijk dan in het inspect-venster bij *Sources* of de bestanden hun nieuwe inhoud vertonen (screenshot links). Is het nog de oude inhoud, ga dan naar *Network* en vink *Disable cache* aan (screenshot rechts).

