# Opdracht permanente evaluatie: simuleer een elektronisch stemsysteem

Versie: 2 (7 mei) - wijziging: geen verplichting meer tot creatie entiteit Kandidaat

In deze opdracht bouw je in Python een simulatie van een elektronisch stemsysteem. Je zult klassen modelleren die de rollen en interacties vertegenwoordigen zoals die voorkomen in een typisch Belgisch stemproces. Je zoekt op (!) hoe dit proces in de realiteit werkt en zorgt ervoor dat jouw oplossing dit proces zo goed mogelijk benadert. We blijven hier weliswaar in de context van één stembureau. Het proces omvat het creëren van kiezers, het organiseren van kandidaten op lijsten, het stemmen via een stemcomputer, het controleren van stembiljetten en het bepalen van de uitslag. Je zult hierbij gebruik maken van compositie (overerving mag!) om realistische relaties te modelleren.

## Vereisten

### Entiteiten:

- Kiezer: Een persoon die stemt, mag slechts één keer stemmen (een kiezer kan zich ook kandidaat stellen).
- Lijst: Een verzameling van kiezers (kandidaten) onder een specifieke partij of groepering.
- Stembiljet: Een document dat wordt afgedrukt na het stemmen, bevat de informatie over de keuze van de kiezer.
- Stembus: Verzamelt en verwerkt alle gescande stembiljetten.
- Scanner: Controleert en registreert een stembiljet bij deponering.
- Stemcomputer: Beheert het hele proces van stemmen en het afdrukken van stembiljetten.
- Chipkaart: Bevat unieke identificatiecodes om stemcomputers op te starten.
- USBStick: Bevat opstartcodes voor de stembus.

Je gebruikt minstens de bovenstaande entiteiten. Je mag daarnaast ook gebruik maken van één (of meerdere) overkoepelende klasse(n). Bvb Kiessysteem.

### Scenario:

- Bij het opstarten van het systeem moet de stembus worden geïnitialiseerd met een USB-stick en de opstartcodes. Deze usb-stick krijgt de code en gaat na de initialisatie terug in de stembus.
- Je zorgt voor 1200 willekeurig aangemaakte kiezers met willekeurige voor- en familienamen en leeftijden variërend van 18 tot 90.
- Je maakt 5 fictieve lijsten (geen bestaande partijen) en zorgt voor 10 kandidaten per lijst (de kandidaten komen uit de poule van 1200 kiezers).
- Maak drie stemcomputers die de input van een kiezer zullen verwerken. Ze worden bij aanvang geïnitialiseerd met de USB-stick waarop de opstartcodes staan. Nadat is gestemd wordt door de stemcomputer een stembiljet aangemaakt.
- Er zijn 60 chipkaarten en die moeten voor elke kiezer geïnitialiseerd worden. De stemcomputer checkt of de kiezer een geïnitialiseerde chipkaart bijheeft en de kiezer geeft ze na het stemmen terug af. Daarna moet ze terug geïnitialiseerd worden.
- Bij het stemmen kan de kiezer een lijststem of één of meerdere voorkeurstemmen uitbrengen.

- Gebruik de "scanner" om te controleren of het stembiljet geldig is en registreer het vervolgens.
- Het systeem moet voorkomen dat iemand meer dan één keer stemt.

## Uitwerking

- Maak klassen voor elk van de bovengenoemde entiteiten en implementeer hun bijbehorende methoden.
- Zorg ervoor dat de kiezer-klasse de mogelijkheid heeft om te stemmen, terwijl de kandidaat-klasse stemmen kan ontvangen.
- Gebruik de stemcomputers om het stemproces en het deponeren van het stembiljet te beheren.
- Simuleer een scenario waarbij kiezers hun stem uitbrengen en het systeem de stemmen correct registreert en controleert. Je respecteert de regels rond lijststemmen en voorkeurstemmen.
- Je logt het proces van de stemming via de terminal en je voorziet een gebruiksvriendelijke html-output met de uitslag van de stemming.

PAS OP:

- Je maakt GEEN gebruik van RECHTSTREEKS door ChatGPT of andere generative AI tools gegenereerde code. Alle code is jouw code.
- Je maakt GEEN gebruik van code die je van medestudenten hebt verkregen. Dit is plagiaat en wordt streng gestraft. Ook de student die de code heeft doorgegeven is even strafbaar.

## Beoordeling

Je implementatie zal worden beoordeeld op basis van:

- De correctheid van het gesimuleerde stemproces (bvb registratie en deponering).
- De relaties tussen de entiteiten en het gebruik van overerving/compositie.
- De juistheid van foutmeldingen en het controleren van de geldigheid van stemmen.
- De terminal-output en de html-output.
- De aanwezigheid van een reflectieverslag waarin je het ontwikkelproces documenteert. Je beschrijft ook de werking van je oplossing en geeft ook aan welke zaken je niet werkend hebt gekregen.
- Studenten AI: voeg een verslag toe over hoe AI in de toekomst een rol zou kunnen spelen in dit proces.
- Studenten IoT: voeg een verslag toe over hoe IoT in de toekomst een rol zou kunnen spelen in dit proces.
- Studenten CSC: voeg een verslag toe met een reflectie over de veiligheid van dergelijk stemproces. Zijn er kwetsbaarheden, en zo ja: hoe zou je dit beter kunnen beveiligen?
- Eventuele creatieve toevoegingen en uitbreidingen bovenop de basisvereisten.

Gebruik deze opdracht om na te denken over hoe software kan worden ontworpen om complexe processen te beheren en een betrouwbaar elektronisch stemsysteem te simuleren.

Veel succes!