Hogeschool van Amsterdam

**HBO – ICT**

**TENTAMENVOORBLAD**

**Voor aanvang van het tentamen s.v.p. het gehele tentamen goed doorlezen om eventuele misverstanden te voorkomen!**

| | | |
|---|---|---|
| **Studieonderdeel** | : | Object Oriented Programming 1 |
| **Datum** | : | 27 januari 2021 |
| **Tijd** | : | 16:00 – 17:40 (100 minuten) + 10 min uploadtijd |
| **Propedeuse/Hoofdfase** | : | Propedeuse |
| **Leerroutes** | : | SE(inclusief herkansers) |

| | | |
|---|---|---|
| **Aantal bladz. (incl.voorblad)** | : | 7 |
| **Toegestane hulpmiddelen** | : | • Boek van Liang (mag digitaal)<br>• IntelliJ & Java Quick Reference (beschikbaar via TestVision)<br>• Uitwerkingen van je eigen practicumopdrachten en oefentoetsen |
| **Normering tentamen** | : | Zie toetstekst. |
| **Bijzonderheden** | : | Als je klaar bent met de opdracht, upload dan een .zip bestand van het hele project naar Testvision. |

# Veel succes!

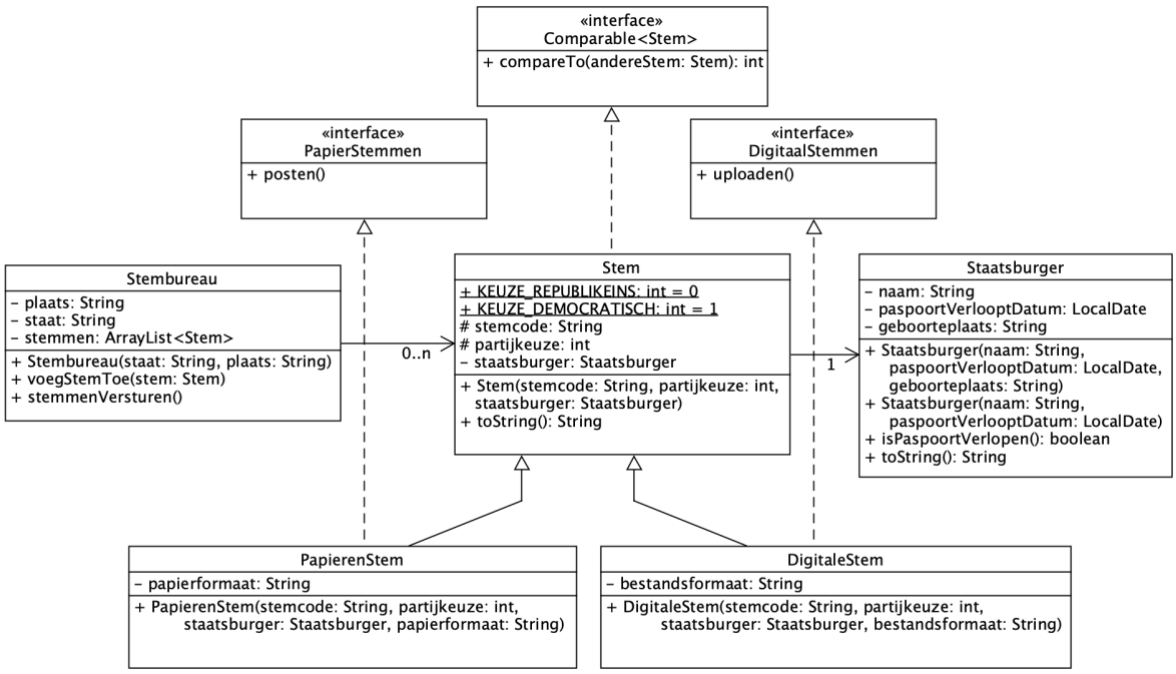
# Stembureau

## Vooraf

- Voor dit tentamen is er geen startproject. Je maakt in IntelliJ een nieuw project aan net zoals je dat gewend bent te doen bij de practicumopdrachten. Bij package name vul je het volgende in: `nl.hva.stembureau`.
- In totaal kun je 100 punten halen. Cijfer = punten / 10.
- Als je klaar bent met de opdracht, upload een .zip van je hele project in TestVision.

## Klassendiagram

Jij bent ingehuurd als Software Engineer om bij een stembureau digitaal stemmen mogelijk te maken voor de presidentsverkiezing. Het stembureau waarvoor je dit doet bevindt zich in het dorpje Nikolai in Alaska. Dit dorpje heeft maar 88 inwoners waarvan maar een klein deel een stem uitbrengt. Het bestaande stemmen via papier moet ook nog steeds mogelijk blijven. Jouw taak is om een eerste versie van deze applicatie te gaan maken. Het klassendiagram voor deze applicatie ziet er als volgt uit:



## Algemene aanwijzingen

- Gebruik de klasse `Main` voor het testen van je code. Voor de meeste stappen schrijf je je eigen testcode, dit staat omschreven in de instructies.
- Omwille van de tijd hoef je alleen `Javadoc` toe te voegen aan de klasse `Stembureau`.
- Implementeer klasse attributen en constructor(s) zoals aangegeven in het klassendiagram.
- In het klassendiagram zijn geen getters en setters opgenomen. Voeg zelf de getter(s) en setter(s) toe die je nodig denkt te hebben. Meer mag, maar hoeft niet. Ook niet alle methodes die vanuit een parent/interface worden verplicht zijn opgenomen, deze dien je wel te implementeren als een stap dat vereist.
- Je mag gebruik maken van de "*IntelliJ Quick Reference voor OOP1*" en van het boek van Liang.
- Als je zelf nog zaken wilt toevoegen (omdat je denkt dat ze nodig zijn) die niet in het class diagram staan, geef dat dan duidelijk met commentaar aan in de code.

# Instructies per stap

## Stap 0: Maak de welkomstboodschap

Voeg in de `Main` klasse een welkomstbericht toe via een print statement. Zorg dat je je eigen gegevens invult tussen de < >.

```
Welkom bij het tentamen Stembureau: <Naam> <Klas> <Studentnummer>
```

## Stap 1: Staatsburger en Stem (25 punten)

Bij deze stap testen we het weergeven van een `Stem` met bijbehorende `Staatsburger`. Voer de volgende instructies uit aan de hand van het UML en de gegeven voorbeeldoutput.

1. Maak de klasse `Staatsburger`.
    a. In het klassendiagram staan twee constructoren aangegeven. Maak beide constructoren en zorg dat je *constructor chaining* toepast. Je kunt de tekst "`onbekend`" gebruiken als default waarde voor `geboorteplaats`.
    b. Zorg dat de methode `isPaspoortVerlopen()` controleert of het attribuut `paspoortVerlooptDatum` een datum is in het verleden.

    **HINT:** Je kunt hiervoor methode `isBefore(..)` gebruiken uit de `LocalDate` klasse

    c. Zorg dat de `toString()` methode de onderstaande String kan teruggeven. Gebruik hierbij de methode die je bij punt b hebt gemaakt i.c.m. de attributen uit deze klasse. Je hoeft het format van de datum niet aan te passen.

```
naam: Teststaatsburger, paspoort verloopt op: 2019-10-30, is paspoort
verlopen: ja, geboorteplaats: onbekend
```

De waarde achter geboorteplaats is afhankelijk van welke constructor er gebruikt wordt bij het aanmaken van de instantie!

2. Maak de klasse `Stem`. Je hoeft de *child-klassen* `DigtaleStem` en `PapierenStem` nog niet te maken.
    a. Zorg dat de `toString()` methode de volgende String kan teruggeven op basis van de attributen uit de klasse:

```
stemcode: TEST01, partijkeuze: Democratisch

staatsburger gegevens:
naam: Teststaatsburger, paspoort verloopt op: 2019-10-30, is paspoort
verlopen: ja, geboorteplaats: onbekend
```

**HINT**: Denk aan herbruik van code!

3. In de `Main` klasse:
    a. Maak een een instantie(object) van de eerder gemaakte `Staatsburger` klasse. Zorg dat de waardes van de attributen overeenkomen met de getoonde output bij punt 1.
    b. Maak een instantie van de eerder gemaakte `Stem` klasse. Zorg dat de waardes van de attributen overeenkomen met de getoonde output bij punt 2. Voor de `partijkeuze`, zorg dat je gebruik maakt van de *static* properties van de klasse `Stem`.
    c. Zorg nu dat je in een print statement de `toString()` methode van de `Stem` instantie aanroept. Geprinte output na het voltooien van stap 1:

```
---------- Stap 1: Stem en Staatsburger----------
stemcode: TEST01, partijkeuze: Democratisch

staatsburger gegevens:
naam: Teststaatsburger, paspoort verloopt op: 2019-03-04, is paspoort
verlopen: ja, geboorteplaats: onbekend
```

## Stap 2: Stembureau (10 punten)

Voer de volgende instructies uit aan de hand van het UML en de gegeven voorbeeldoutput. **Bij deze stap maak je nog niet alle methodes van de klasse** `Stembureau`**.**

1. Maak de attributen en de constructor van de klasse `Stembureau` aan. Zorg dat de `stemmen` ArrayList met een lege lijst wordt geinitialiseerd.
2. Maak de methode `voegStemToe(..)`. Deze methode moet ervoor zorgen dat een stem kan worden toegevoegd aan de `stemmen` ArrayList.
3. In de `Main` klasse:
    a. Maak een instantie aan van de `StemBureau` klasse.
    b. Zorg vervolgens dat de volgende output geprint wordt door middel van een `printf` statement. Je mag in dit geval niet de `toString()` methode gebruiken.
    **HINT:** Denk aan encapsulation

```
---------- Stap 2: Stembureau----------

Welkom bij het stembureau in: Nikolai, Alaska
```

## Stap 3: DigitaleStem en PapierenStem (30 punten)

Voer de volgende instructies uit aan de hand van het UML en de gegeven voorbeeldoutput.

1. Maak de klassen `DigtaleStem` en `PapierenStem`. Maak hierbij gebruik van inheritance.
2. Maak vervolgens de bijbehorende interfaces en implementeer deze in de juiste klassen.
    a. De methode `uploaden()` moet de volgende output kunnen printen op basis van de attributen uit de klasse:

```
Stem met code: DIGITAAL_TEST01 wordt geüpload in bestandsformaat
.pdf!
```

    b. De methode `posten()` moet de volgende output kunnen printen op basis van de attributen uit de klasse:

```
Stem met code: PAPIER_TEST01 wordt gepost, papierformaat: a4!
```

3. In de `Main` klasse gaan we eenmalig voor zowel `DigtaleStem` als `PapierenStem` een teststem uitvoeren.
    a. Maak een instantie aan van `DigtaleStem`. Zorg dat de waardes van de attributen overeenkomen met de getoonde output bij punt 2a. Zorg vervolgens dat deze stem geüpload wordt.
    **HINT:** herbruik de `staatburger` instantie uit stap 1 bij het aanmaken van de `DigtaleStem` instantie.
    b. Maak een instantie aan van `PapierenStem`. Zorg dat de waardes van de attributen overeenkomen met de getoonde output bij punt 2b. Zorg vervolgens dat deze stem gepost wordt.
    c. Dit zou de volgende output moeten genereren:

```
---------- Stap 3: DigitaleStem en PapierenStem ----------
Stem met code: DIGITAAL_TEST01 wordt geüpload in bestandsformaat .pdf!
Stem met code: PAPIER_TEST01 wordt gepost, papierformaat: a4!
```

## Stap 4: DigitaleStem en PapierenStem versturen via het Stembureau(25 punten)

Na net voor zowel de `DigtaleStem` als de `PapierenStem` een test te hebben uitgevoerd gaan we nu stemmen versturen via het `Stembureau`.

1. Voordat de stemmen verstuurd kunnen worden moeten deze nog gesorteerd worden op `stemcode`. Hierbij moet de stem met het laagste nummer in de stemcode bovenaan komen te staan in de ArrayList. Kijk naar het UML en implementeer de `Comparable` interface op zo'n manier dat dit mogelijk wordt.
2. Ga naar de klasse `Stembureau` en maak de methode `stemmenVersturen()`. Sorteer als eerste de lijst zoals bij het vorige punt is aangegeven. Vervolgens moet deze methode moet voor elke stem in de `stemmen` ArrayList controleren of het een instantie is van type `DigtaleStem` of `PapierenStem`. Ten slotte moet deze stem worden geüpload of gepost via de juiste methode. Het testen van deze `stemmenVersturen()` methode gebeurt hierna.

3. We hebben vanaf nu wat testdata nodig voor de stemmen. Voeg in de `Main` klasse onder de `main(..)` methode de volgende twee methodes toe:

```
private static DigitaleStem genereerDigitaleStem() {
    Staatsburger staatsburger = new Staatsburger("Teststaatsburger",
            LocalDate.of(1993, 11,  16), "Nikolai");

    return new DigitaleStem("DIGTAAL_TEST" + (int) (Math.random() * 901 +
100), 1, staatsburger, ".pdf");
}

private static PapierenStem genereerPapierenStem() {
    Staatsburger staatsburger = new Staatsburger("Teststaatsburger",
            LocalDate.of(1988, 10,  13), "Nikolai");

    return new PapierenStem("PAPIER_TEST" + (int) (Math.random() * 901 +
100), 0, staatsburger, "A4");
}
```

Deze methodes maken gebruik van de andere constructor van `Staatsburger`. Zorg dat beide constructoren werken zoals gevraagd werd bij stap 1 van dit tentamen.

   a. Zorg nu dat in de `main(..)` methode er 10x een `DigtaleStem` en 10x een `PapierenStem` worden toegevoegd aan het `Stembureau`. Voor het genereren van deze stemmen gebruik je de methodes die je hierboven gegeven zijn. Daarbij voeg je deze 20 instanties natuurlijk niet regel voor regel toe.

4. Ten slotte roep je de bij punt 1 gemaakte methode `stemmenVersturen()` aan. Gebruik de voorbeeldoutput hieronder om te controleren of je code werkt.

   **LET OP:** de getallen achter de stemcode zijn random, deze zijn dus anders per keer dat je het programma draait. Zorg ervoor dat het aantal digitale en papier stemmen wel overeenkomt met de voorbeeldoutput. Ook kun je hier verifiëren of je sortering correct werkt.

```
---------- Stap 4: DigitaleStem en PapierenStem versturen via het
Stembureau ----------

Stem met code: DIGTAAL_TEST146 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST184 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST214 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST251 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST416 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST426 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST482 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST556 wordt geüpload in bestandsformaat .pdf!

Stem met code: DIGTAAL_TEST665 wordt geüpload in bestandsformaat .pdf!
```

```
Stem met code: DIGTAAL_TEST716 wordt geüpload in bestandsformaat .pdf!
Stem met code: PAPIER_TEST200 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST263 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST364 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST479 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST600 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST630 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST643 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST684 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST782 wordt gepost, papierformaat: A4!
Stem met code: PAPIER_TEST907 wordt gepost, papierformaat: A4!
```

## Let op de code conventions (10 punten)

- Zorg dat je naam en het doel bij elke class bovenin staan (ICC #1).
- Gebruik de juiste inspringing (indentation) bij de lay-out (ICC #2).
- Let op juist gebruik hoofdletters en kleine letters (ICC #3).
- Gebruik goede namen (ICC #4).
- Vermijd magic numbers (ICC#5) en dode code (ICC #8).
- Gebruik Javadoc tags: @author, @param en @return (ICC #6).
- Voeg waar nodig commentaar toe dat inzicht geeft in je code (ICC#7).
- Denk aan encapsulation (ICC #9).