Faculté des Sciences
Aix*Marseille Université

Année universitaire 2021-2022

Site : ☒ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☐ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☐ 1er semestre ☒ 2ème semestre ☐ Session 2 Durée de l'épreuve : 2h
Examen de : L1 Nom du diplôme : Portail René Descartes
Code du module : SPO2U07L Libellé du module : Programmation 1
Calculatrices autorisées : NON Documents autorisés : OUI, notes de Cours, supports de cours

---

1 Consignes

— Les dernières pages du sujet sont à compléter et à rendre avec votre copie.
— Lisez bien le sujet avant de commencer à répondre.
— Pour les chaînes de caractères et les listes, vous avez une annexe donnant les méthodes utiles sur `String` et `List`.
— Si vous manquez de places dans les cases dédiées pour votre code, c'est que votre solution est trop complexe et/ou ne réutilise pas assez le code existant.

2 Gestion d'élection

Dans ce sujet d'examen, on va s'intéresser à un système permettant de simuler des votes. Il y aura donc des classes correspondant à des citoyens (*citizens*), des bureaux de vote (*polling places*) et des résultats d'élection (*election results*).

3 Citoyen : classe `Citizen`

On considère donc la classe `Citizen` permettant de modéliser des citoyens et dont le code incomplet est à la fin du sujet et sera à rendre. La classe `Citizen` contient les attributs suivant :

— attributs d'instance `firstName` (le prénom du citoyen), `lastName` (le nom de famille du citoyen), `age` (l'age du citoyen), `voterId` (le numéro d'électeur du citoyen),
— attributs de classe `VOTING_AGE` correspondant à l'age de majorité électoral (age minimal pour pouvoir voter), attributs de classe `citizenCount` correspondant au nombre d'instances de `Citizen` qui ont été créées.

Question 1 (2 points) : *Compléter les déclarations des attributs `VOTING_AGE`, `citizenCount`, `firstName`, `lastName`, `age` et `voterId`.*

Question 2 (1 point) : *Est-ce qu'il est possible de définir une méthode `setVoterId` dans la classe `Citizen` qui permettrait de changer le numéro d'électeur du citoyen (répondre sur votre copie) ?*

Question 3 (0,5 points) : *Compléter la méthode `incrementAge` de la classe `Citizen` qui augmente d'un l'age d'un citoyen.*

Question 4 (0,5 points) : *Compléter la méthode `canVote` de la classe `Citizen` qui renvoie `true` si le citoyen a l'age de voter.*

Question 5 (0,5 points) : *Compléter la méthode* `getUpperCaseLastName` *de la classe* `Citizen` *qui renvoie le nom de famille du citoyen en majuscules (par exemple pour un citoyen ayant pour nom de famille* `lAbourel`*, la méthode devra renvoyer* `LABOUREL`*).*

Question 6 (1,5 points) : *Compléter la méthode* `getCapitalizedFirstName` *de la classe* `Citizen` *qui renvoie le prénom du citoyen avec la première lettre en majuscule et toutes les autres lettres en minuscules (par exemple pour un citoyen ayant pour prénom* `aRnauD`*, la méthode devra renvoyer* `Arnaud`*).*

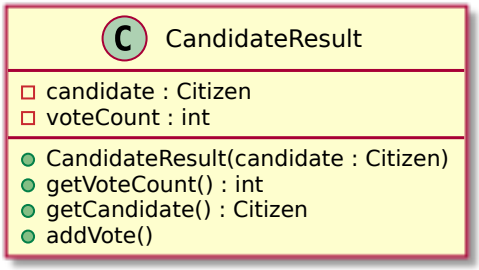
Question 7 (1 point) : *Compléter la méthode* `getName` *de la classe* `Citizen` *qui renvoie le nom complet du citoyen c'est-à-dire le prénom et le nom de famille avec un espace entre les deux. Le prénom et le nom devront être au format indiqué par les deux questions précédentes (par exemple pour un citoyen ayant pour prénom* `aRnauD` *et pour nom* `lAbourel`*, la méthode devra renvoyer* `Arnaud LABOUREL`*).*

Question 8 (1,5 points) : *Compléter le constructeur de la classe* `Citizen` *qui permet d'instancier un citoyen avec un prénom, un nom de famille et un age. Un citoyen a pour numéro d'identifiant le nombre de citoyens qui ont été instanciés avant son instanciation. Le nombre de citoyens instanciés doit évidemment être mis à jour.*

Question 9 (0,5 points) : *Compléter la méthode* `equals` *de la classe* `Citizen` *qui renvoie* `true` *si l'objet passé en argument correspond à un citoyen ayant le même numéro d'électeur.*

4 Résultat d'un candidat : classe `CandidateResult`

On considère la classe `CandidateResult` qui permet de gérer les résultats d'élection d'un candidats.



Une nouvelle instance de `CandidateResult` contient initialement un nombre de votes (`voteCount`) de 0.

Question 10 (1 point) : *Donnez la déclaration complète (avec entête de la méthode et code) de la méthode* `addVote` *de la classe* `CandidateResult` *qui permet d'ajouter un vote au résultat du candidat (répondre sur votre copie).*

5 Résultat d'élection : classe `ElectionResult`

La classe `ElectionResult` permet de représenter les résultats d'une élection. Elle contient les attributs d'instance suivants :

— `candidateResults` : la liste des résultats des candidats,
— `nullVotes` : le nombre de vote nuls et
— `voterTurnout` : le taux de participation pour l'élection (valeur entre 0 et 1 représentant la proportion d'électeurs enregistrés qui ont voté).

La classe `ElectionResult` possède un constructeur qui prend en paramètre une liste de candidats et un taux de participation.
La classe `ElectionResult` contient les méthodes d'instance suivantes :

— `addVote` qui prend en argument un bulletin de vote représenté par une chaîne de caractères et qui ne renvoie rien, si le bulletin correspond (chaînes de caractères ayant les mêmes caractères dans le même ordre) au nom complet d'un des candidats, la méthode ajoute un vote au résultat du candidat, sinon la méthode ajoute un vote aux votes nuls ;
— `print` qui n'a pas d'argument, ne renvoie rien et affiche les résultats ;
— `expressedVotes` qui n'a pas d'argument et qui renvoie le nombre de votes exprimés (votes correspondants à un candidat et donc non-nuls).

Question 11 (1,5 points) : *Dessiner le diagramme de la classe* `ElectionResult` *(répondre sur votre copie).*

Question 12 (1,5 points) : *Donnez la déclaration complète (avec entête de la méthode et code) de la méthode* `addVote` *de la classe* `ElectionResult` *(répondre sur votre copie).*

6 Bureau de vote : classe `PollingPlace`

La classe `PollingPlace` permettant de modéliser des bureaux de votes contient les attributs d'instance suivants :

— `registeredVoters` : une liste de citoyens correspondant aux électeurs enregistrés du bureau de vote,
— `participatingVoters` : une liste de citoyens correspondant aux électeurs ayant voté du bureau de vote et
— `ballots` : la liste des bulletins sachant que chaque bulletin est représenté par une chaîne de caractères.

Question 13 (0,5 points) : *Compléter les déclarations des attributs* `registeredVoters`, `participatingVoters` *et* `ballots` *de la classe* `PollingPlace`.

Question 14 (1 point) : *Compléter le constructeur de la classe* `PollingPlace` *qui permet d'instancier un bureau de vote à partir d'une liste* `possibleVoters` *de citoyens donnée en argument. Le bureau de vote aura pour liste d'électeurs enregistrés les citoyens de* `possibleVoters` *qui ont l'age de voter, une liste vide d'électeur ayant voté et une liste vide de bulletins.*

Question 15 (1 point) : *Compléter la méthode* `acceptVoteFrom` *de la classe* `PollingPlace`. *Cette méthode renvoie* `true` *si le citoyen passé en argument a le droit de voter, c'est-à-dire qu'il est enregistré dans le bureau de vote et qu'il n'a pas déjà voté.*

Question 16 (1 point) : *Compléter la méthode* `castBallot` *de la classe* `PollingPlace`. *Cette méthode prend un citoyen et bulletin en argument. Si le citoyen a le droit de voter, elle stocke son bulletin dans la liste de bulletins, ajoute le citoyen aux électeurs ayant voté et renvoie* `true`. *Si le citoyen n'a pas le droit de voter, elle ne fait rien à part renvoyer* `false`.

Question 17 (0,5 points) : *Compléter la méthode* `voterTurnout` *de la classe* `PollingPlace`*. Cette méthode renvoie le taux de participation, c'est-à-dire la proportion d'électeurs enregistrés qui ont voté (valeur entre 0 et 1).*

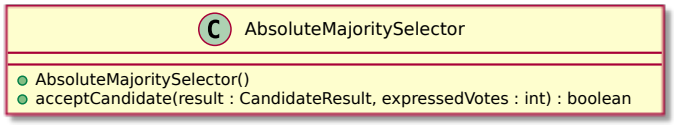
Question 18 (1 point) : *Compléter la méthode* `countTheVotes` *de la classe* `PollingPlace`*. Cette méthode crée un résultat d'élection à partir des candidats passés en paramètre et ajoute tous les bulletins aux résultats de l'élection (c'est-à-dire comptabilise les votes exprimés par les bulletins stockés).*

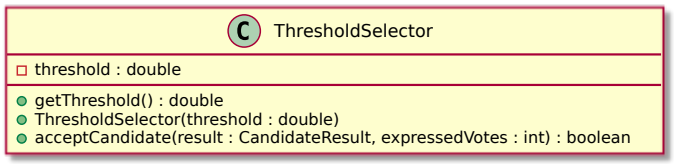
7 Interface `CandidateSelector`

On souhaite définir deux classes ayant une méthode `acceptCandidate` qui accepte ou non un candidat à partir du résultat du candidat et du nombre de vote exprimé :

— `AbsoluteMajoritySelector` qui accepte les candidats ayant strictement plus que la moitié des votes exprimés.
— `ThresholdSelector` qui accepte les candidats dont le nombre de votes correspond à une proportion supérieure ou égale à un seuil (*threshold*). Par exemple, pour un seuil `0.15`, les candidats acceptés seront ceux qui ont au moins 15% des votes.

Ces deux classes ont les diagrammes suivants :





Question 19 (1 point) : *Donner la déclaration de l'interface* `CandidateSelector` *implémentée par les deux classes* `AbsoluteMajoritySelector` *et* `ThresholdSelector` *(répondre sur votre copie).*

Question 20 (0,5 points) : *Écrivez la classe* `AbsoluteMajoritySelector` *(répondre sur votre copie).*

Question 21 (0,5 points) : *Écrivez la méthode* `List<Citizen> selectedCandidates(CandidateSelector selector)` *dans la classe* `ElectionResult` *qui renvoie parmi les candidats dans le résultat de l'élection, ceux qui sont acceptés par le* `selector`*.*

8 Annexe

8.1 Documentation de `List`

Pour cet examen, toutes les variables de type `List` seront à initialiser avec des objets de type `ArrayList`.
Vous pouvez utilisez les méthodes suivantes de `List<E>` :

- `boolean add(E e)` : Appends the specified element to the end of this list.
- `int size()` : Returns the number of elements in this list.
- `E get(int index)` : Returns the element at the specified position in this list.
- `boolean remove(Object o)` : Removes the first occurrence of the specified element from this list, if it is present.
- `boolean isEmpty()` : Returns `true` if this list contains no elements.

8.2 Documentation de la classe `String`

Voici un extrait de la documentation de la classe `String` :

```
/**
 * Returns a string that is a substring of this string. The
 * substring begins with the character at the specified index and
 * extends to the end of this string.
 * Examples:
 * "unhappy".substring(2) returns "happy"
 * "Harbison".substring(3) returns "bison"
 * "emptiness".substring(9) returns "" (an empty string)
 *
 * @param      beginIndex   the beginning index, inclusive.
 * @return     the specified substring.
 */
public String substring(int index){/* code */}
/**
 * Returns a string that is a substring of this string. The
 * substring begins at the specified {@code beginIndex} and
 * extends to the character at index {@code endIndex - 1}.
 * Thus the length of the substring is {@code endIndex-beginIndex}.
 * Examples:
 * "hamburger".substring(4, 8) returns "urge"
 * "smiles".substring(1, 5) returns "mile"
 *
 * @param      beginIndex   the beginning index, inclusive.
 * @param      endIndex     the ending index, exclusive.
 * @return     the specified substring.
 */
public String substring(int beginIndex, int endIndex){/* code */}
/**
 * Converts all of the characters in this {@code String} to lower
 * case using the rules of the default locale.
 * @return  the {@code String}, converted to lowercase.
 */
public String toLowerCase(){/* code */}
/**
 * Converts all of the characters in this {@code String} to upper
 * case using the rules of the default locale.
 * @return  the {@code String}, converted to uppercase.
 */
public String toUpperCase(){/* code */}
```

9 À compléter et à rendre

9.1 Classe `Citizen` (à rendre)

```
public class Citizen {
    [                                                    ] int VOTING_AGE = 18;

    [                                                    ] citizenCount = 0;

    [                                                    ] final String firstName;

    [                                                    ] final String lastName;

    [                                                    ] int age;

    [                                                    ] final int voterId;

    public void incrementAge() {
        [                                                                    ]
    }

    public [                                ] canVote(){
        [                                                                    ]
    }

    private [                               ] getUpperCaseLastName(){
        [                                                                    ]
    }

    private [                               ] getCapitalizedFirstName(){
        [                                                                    ]
    }

    public [                                ] getName() {
        [                                                                    ]
    }
```

```
  public Citizen(String firstName, String lastName, int age) {
    
  }

  public boolean equals(Object o) {
    if (!(o instanceof Citizen))
      return false;
    Citizen citizen = (Citizen) o;
    
  }
}
```

9.2 Classe PollingPace (à rendre)

```
public class PollingPlace {

    [                    ] registeredVoters;

    [                    ] participatingVoters;

    [                    ] ballots;


    public PollingPlace([          ] possibleVoters) {
        [                                          ]
    }

    private [          ] acceptVoteFrom(Citizen citizen) {
        [                                          ]
    }
```

```
  public                          castBallot(Citizen citizen, String ballot) {
                                  
  }

  public                          voterTurnout(){
                                  
  }

  public                          countTheVotes(                          candidates){
                                  
  }
}
```