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 Classe `Citizen`

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

```
private final static int VOTING_AGE = 18;
private static int citizenCount = 0;

private final String firstName;
private final String lastName;
private int age;
private final int 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) ?*

L'attribut `voterId` étant `final` et le type `int` étant primitif, il est impossible de modifier le `voterId` d'un `Citizen` et il n'est donc pas possible de définir une méthode `setVoterId` dans la classe `Citizen`.

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

```
public void incrementAge(){
  age++;
}
```

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

```
public boolean canVote(){
  return age >= VOTING_AGE;
}
```

**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`).

```
private String getUpperCaseLastName(){
  return lastName.toUpperCase();
}
```

**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`).

```
private String getCapitalizedFirstName(){
  return firstName.substring(0,1).toUpperCase() + firstName.substring(1).toLowerCase();
}
```

**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`).

```
public String getName() {
  return  getCapitalizedFirstName() + " " + getUpperCaseLastName();
}
```

**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.*

```
public Citizen(String firstName, String lastName, int age) {
  this.lastName = lastName;
  this.firstName = firstName;
  this.age = age;
  this.voterId = citizenCount ++;
}
```

**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.*

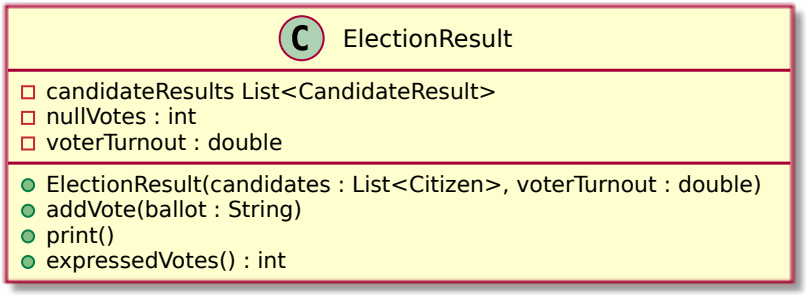
```
public boolean equals(Object o) {
  if (!(o instanceof Citizen))
```

```
    return false;
  Citizen citizen = (Citizen) o;
  return this.voterId == citizen.voterId;
}
```

**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).*

```
public void addVote(){
  voteCount++;
}
```

**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).*

```
public void addVote(String ballot){
  for(CandidateResult result : candidateResults)
    if(result.getCandidate().getName().equals(ballot)) {
      result.addVote();
      return;
    }
  this.nullVotes++;
}
```

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

```
private final List<Citizen> registeredVoters;
private final List<Citizen> participatingVoters;
private final List<String> ballots;
```

**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.*

```
public PollingPlace(List<Citizen> possibleVoters) {
  this.registeredVoters = new ArrayList<>();
  for(Citizen citizen : possibleVoters)
    if(citizen.canVote())
      registeredVoters.add(citizen);
  this.participatingVoters = new ArrayList<>();
  this.ballots = new ArrayList<>();
}
```

**Question 15 (1 point) :** *Compléter la méthode* `acceptVoteFrom` *de la classe* `PollingPlace`*. Cette méthode renvoie* `true` *si le seulement le citoyen passé en argument 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.*

```
private boolean acceptVoteFrom(Citizen citizen) {
  return registeredVoters.contains(citizen)
          && !participatingVoters.contains(citizen);
}
```

**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`*.*

```
public boolean castBallot(Citizen citizen, String ballot) {
  if (!acceptVoteFrom(citizen))
    return false;
  participatingVoters.add(citizen);
  ballots.add(ballot);
  return true;
}
```

**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).*

```
public double voterTurnout(){
  return participatingVoters.size() / (double) registeredVoters.size();
}
```

**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.*

```
  public ElectionResult countTheVotes(List<Citizen> candidates){
    ElectionResult result = new ElectionResult(candidates, voterTurnout());
    for(String ballot : ballots){
      result.addVote(ballot);
    }
    return result;
  }
```

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

```
public interface CandidateSelector {
  boolean acceptCandidate(CandidateResult result, int expressedVotes);
}
```

**Question 20 (1 point) :** *Écrivez la classe* `AbsoluteMajoritySelector` *(répondre sur votre copie).*

```
public class AbsoluteMajoritySelector implements CandidateSelector{
  public boolean acceptCandidate(CandidateResult result, int expressedVotes) {
    return result.getVoteCount() > expressedVotes/2;
  }
}
```

**Question 21 (1 point) :** *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`.

```
  List<Citizen> selectedCandidates(CandidateSelector selector){
    List<Citizen> selectedCandidates = new ArrayList<>();
    for (CandidateResult result : candidateResults){
      if(selector.acceptCandidate(result, expressedVotes()))
        selectedCandidates.add(result.getCandidate());
    }
    return selectedCandidates;
  }
```