%(ID)
Mark: 12.73/20 (total score: 14/22)

+24/1/60+

QCM TEST

Introduction à la programmation orientée objet
28/10/2021

Nom et prénom : LATAPIE Florian

Groupe : .....

Cochez les cases en mettant une ×. Le symbole ⊕ indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions :

- *cocher une bonne réponse apporte des points positifs*
- *ne pas cocher une mauvaise réponse apporte des points positifs*
- *cocher une mauvaise réponse peut apporter des points négatifs*
- *ne pas cocher une bonne réponse peut apporter des points négatifs*

Pour les question à une seule réponse :

- *cocher une bonne réponse apporte des points positifs*
- *cocher une mauvaise réponse peut apporter des points négatifs*
- *une réponse non-cochée apporte zéro points*

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

Toutes les questions sauf la dernière se basent sur les classes déclarées sur la feuille jointe. Sauf indication contraire, chaque question est indépendante des autres.

Question 1 Pour la classe `Auction`, écrivez la méthode `removeLots` qui satisfait

```
/**
 * Enleve tous les lots pour lesquels une personne donnee avait pose les encheres les plus elevees.
 * @param bidder La personne pour qui on cherche des encheres.
 * @return true si des lots ont ete enleves, sinon false.
 */
public boolean removeLots(Person bidder) {...}
```

Le corps de la méthode sera le plus court possible. ☐0 ☐1 ☒2 ☐3 ☐4 0.5/1

```
public boolean removeLots(Person bidder){
    for(int i=0; i < lots.size(); i++){
        if( lots.get(i).getHighestBid().getBidder().equals(bidder)){
            return lots.remove(i);
        }
    }
    return false; // non trouvé, non supprimé
}
// lots.removeIf( lot -> lot.getHighestBid().getBidder().equals(bidder))
// pas sur que ca marche mais c'est plus court et plus correct
```

enleve 1 lot seulement

Question 2 ⊕ Quels sont les niveaux d'accès par lesquels `P_PP` pourraient être remplacés dans la déclaration `P_PP class AuctionMain` :

1/1

- ☐ `private`
- ☒ `public`
- ☒ package-private (default)

Question 3 ⊕ Pour que la classe `AuctionMain` compile, certaines méthodes de la classe `Auction` doivent obligatoirement avoir un niveau d'accès élargi (`P_PP` ne peut être `private` par exemple). Lesquelles :

1/1

- ☐ `getLot`
- ☐ `removeLots`
- ☒ `makeABid`
- ☒ le constructeur `Auction`
- ☒ `enterLot`
- ☒ `showLots`

Question 4 Dans la question précédente, quel est le niveau d'accès le moins élevé possible :

1/1

- ☐ package-private (default)
- ☒ `public`

Question 5 ⊕ Pour que la classe `Lot` compile, certaines méthodes de la classe `Bid` doivent obligatoirement avoir un niveau d'accès élargi (`P_PP` ne peut être `private` par exemple). Lesquelles :

0/1

- ☒ `getBidder`
- ☒ le constructeur `Bid`
- ☒ `getValue`
- ☒ `getDate`

Question 6 Dans la question précédente, quel est le niveau d'accès le moins élevé possible :

1/1

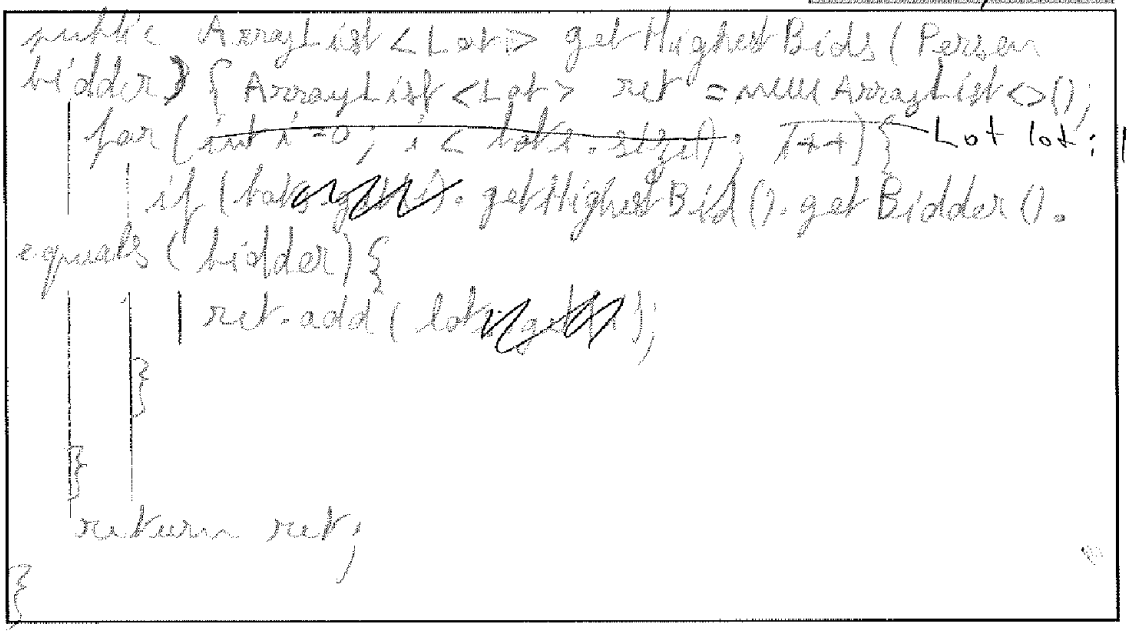
- ☒ package-private (default)
- ☐ `public`

Question 7 Pour la classe `Auction`, développez une nouvelle méthode `getHighestBids` qui satisfait la signature :

```
/**
 * Trouve tous les lots pour lesquels une personne donnee avait pose les encheres les plus elevees.
 * @param bidder La personne pour laquelle on cherche les encheres les plus eleves.
 * @return La liste des lots.
 */
public ArrayList<Lot> getHighestBids(Person bidder) {...}
```

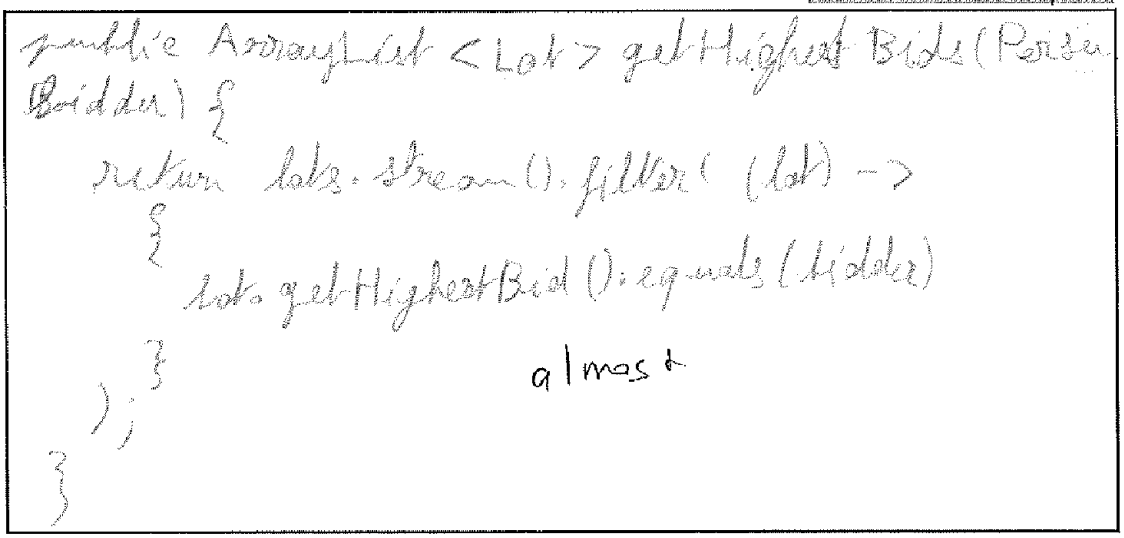
La méthode doit obligatoirement utiliser une boucle, c'est à dire en style procédural.

☐0 ☐1 ☐2 ☒3 ☐4 0.75/1



Question 8 Pour la classe `Auction`, écrivez la méthode `getHighestBids` qui satisfait la signature de la question précédente.
Interdiction d'utiliser aucune boucle, il faut utiliser le style fonctionnel.

☐0 ☐1 ☐2 ☐3 ☒4 1/1



Question 9 Il y a une méthode qui risque de présenter une faille de sécurité dans l'application. Cette méthode se trouve dans la classe :

0/1

- ☐ Lot
- ☐ Person
- ☒ Auction
- ☒ Bid

Question 10 On suppose que tous les problèmes éventuels auraient été résolus pour les classes `Auction`, `Bid`, `Lot`, et `Person`. Quel est le résultat de l'exécution de la classe `AuctionMain` :

```
class AuctionMain {
    private Auction auction = new Auction();

    private void demo() {
        Person fred = new Person("Fred");
        Person notFred = new Person("Not Fred");
        auction.enterLot("Atomic toaster");
        auction.makeABid(1, fred, 23);
        auction.enterLot("Ionic whoopie cushion");
        auction.makeABid(2, fred, 15);
        auction.enterLot("Pan galactic gargle blaster");
        auction.makeABid(3, notFred, 5);
        auction.removeLots(fred);
        auction.showLots();
    }

    public static void main(String... args) {
        demo();
    }
}
```

0/1

- ☐ Aucun de ces choix
- ☒ Rien. Il y a une erreur de compilation
- ☒ `3: Pan galactic gargle blaster Bid: 5`
- ☐ `2: Ionic whoopie cushion Bid: 15`
- ☐ `1: Atomic toaster Bid: 23`
`2: Ionic whoopie cushion Bid: 15`
- ☐ `1: Atomic toaster Bid: 23`

Question 11 ⊕ Dans la classe `Auction`, quelles classes doivent être explicitement importées :

1/1

- ☐ Lot
- ☒ ArrayList
- ☐ System
- ☐ Auction
- ☐ Bid
- ☒ Person
- ☐ String

Question 12 ⊕ Quels attributs pourraient être déclarés `final` :

1/1

- ☐ Auction#nextLotNumber
- ☒ Lot#number
- ☒ Person#name
- ☒ Bid#bidder
- ☒ Auction#lots
- ☐ Lot#highestBid
- ☒ Bid#date
- ☒ Lot#description
- ☒ Bid#value

Nous supposons pour la question suivante que les attributs sont déclarés correctement : `final` ou pas.

Question 13 ⊕ Quelles classes sont immuables (immutable) :

0.5/1

- ☐ Auction
- ☐ Lot
- ☒ Bid
- ☒ Person

Question 14 La classe `Bid` ne compile pas. Pourquoi ? ☒0 ☐1 ☐2 — 0/1

Bid est immuable, date doit être fourni dans le constructeur

P_PP Bid (Person bidder, double value, Date date){...}

Question 15 ⊕ Nous souhaitons compiler les classes. Le code se trouve à partir du dossier *auctionproject/src/*, et on aimerait que le bytecode soit généré dans le dossier *auctionproject/bin*. Tout en restant dans *auctionproject*, lesquelles des commandes feraient l'affaire :

- [x] `javac -d bin src/*/*.java`
- [ ] `javac -d bin *.java`
- [ ] `javac src/*/*.java`
- [x] `javac -d bin src/main/AuctionMain src/auction/Auction src/auction/Bid src/auction/Lot src/person/Person`
- [x] `javac -d bin src/main/*.java src/auction/*.java`

0.2/1

Question 16 ⊕ Pour exécuter le code de la question précédente, cela serait :

- [x] `java -cp bin main.AuctionMain`
- [ ] `java bin/AuctionMain.class`
- [ ] `java -cp bin AuctionMain`
- [ ] `java -cp bin AuctionMain.java`
- [ ] `java -cp bin AuctionMain.class`
- [ ] `java main.AuctionMain`

1/1

Question 17 ⊕ Dans la classe `AuctionMain` on décide de gérer plusieurs ventes avec la déclaration :

```
private final ArrayList<Auction> auctions = new ArrayList<>();
```

a est un `int`.
Alors, quelles expressions seraient valides dans `AuctionMain` pour le compilateur :

- [ ] `auctions[0] = new Auction();`
- [x] `auctions.add(new Auction());`
- [ ] `auctions = new ArrayList<>();`
- [ ] `if (a < auctions.length) {...}`
- [ ] `Auctions.auctions = new ArrayList<>();`
- [ ] `this.auctions = new ArrayList<>();`
- [x] `if (a < auctions.size()) {...}`

1/1

Question 18 ⊕ Dans la classe `AuctionMain` on décide de gérer plusieurs ventes avec la déclaration :

```
private final Auction[] auctions = new Auction[10];
```

a est un `int`.
Alors, quelles expressions seraient valides dans `AuctionMain` pour le compilateur :

0.7143/1

- ☐ `Auctions.auctions = new Auction[10];`
- ☒ `if (a < auctions.length) {...}`
- ☒ `auctions[0] = new Auction();`
- ☐ `if (a < auctions.size()) {...}`
- ☐ `auctions = new Auction[10];`
- ☐ `this.auctions = new Auction[10];`
- ☐ `auctions.add(new Auction());`

Question 19 Dans les deux questions précédentes, quel est le meilleur choix pour la déclaration de `auctions` :

1/1

- ☐ un tableau (*array*)
- ☒ une liste

Question 20 ⊕ a1, a2, a3 sont des objets de type `Auction`. Lesquelles des expressions déclarent, construisent et initialisent correctement un tableau ?

0.3333/1

- ☒ `Auction[] auctions = {a1, a2, a3};`
- ☒ `Auction[] auctions = new Auction[3];`
  `auctions[0] = a1;`
  `auctions[1] = a2;`
  `auctions[2] = a3;`
- ☒ `Auction[3] auctions = {a1, a2, a3};`
- ☐ `Auction[] auctions = (a1, a2, a3);`
- ☐ `Auction auctions = {a1, a2, a3};`
- ☐ `Auction[] auctions = new Auction();`

Question 21 Chaque objet de type `Lot` devrait comprendre un numéro d'identification unique, `number`. Cependant celui-ci dépend de la bienveillance de `AuctionMain` qui passe la valeur au constructeur. Ceci n'est pas satisfaisant.
Suggérez comment `Lot` pourrait générer ses propres identificateurs uniques : 1, 2, 3,...

0.5/1

☐0 ☐1 ☒2 ☐3 ☐4

il faut mettre l'attribut number en static et l'incrémenter dans le constructeur

```
P_PP Lot (String description){
    this.number ++;
    this.description = description;
}
```

pas possible si static

Question 22 ⊕ Indiquez les lignes de la classe `Whoops` qui provoquent une erreur de compilation :

- [x] `private class Whoops {`
- [ ] `    final private int input = 0;`
- [ ] `    private Integer[] inputs = {3, 1, 4, 1, 5};`
- [ ] `    Whoops() {`
- [ ] `        this(9);`
- [x] `        inputs = {3, 1, 4, 1, 5, 9};`
- [ ] `    }`
- [ ] `    private Whoops(int input) {`

0.5/1

- [x] `        input = input;`
- [ ] `    }`
- [x] `    public double aMethod(double output) {`
- [x] `        if (output != null) {`
- [x] `            inputs.forEach(i -> System.out.println(i));`
- [ ] `        }`
- [x] `    }`
- [ ] `}`

+24/8/53+