EPITA ING1 2019 S2 MOB2

Nom et prénom *lisibles*:

....

....

....

....

Identifiant (de haut en bas):

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

Ne rien écrire sur les bords de la feuille, ni dans les éventuels cadres grisés ☠. Noircir les cases plutôt que cocher. Renseigner les champs d'identité. Si votre `uid` ne fait que 5 chiffres, commencez par un 0 (première ligne). Les questions marquées par ♣ peuvent avoir plusieurs réponses justes. Toutes les autres n'en ont qu'une. Il n'est pas possible de corriger une erreur, mais vous pouvez utiliser un crayon. Les réponses justes créditent; les incorrectes pénalisent; et les blanches et réponses multiples valent 0.

■ J'ai lu les instructions et mon sujet est complet: les 4 entêtes sont `+361/1/nn+`···`+361/4/nn+`.

---

1. ♣ Par rapport aux méthodes classiques, quelles sont les caractéristiques de celles de CLOS :

■ On n'exécute jamais une méthode directement
☐ On peut créer des méthodes par défaut
■ On peut spécialiser sur plusieurs arguments (multi-méthodes)
■ Les méthodes ne font pas partie des classes

2. En quoi le mécanisme d'instanciation de CLOS est-il différent de celui des systèmes classiques :

☐ Il n'existe jamais de constructeur par défaut
■ Il n'y a pas de constructeurs mais une fonction centralisée d'instanciation
☐ Il est impossible de garantir l'initialisation des slots
☐ Il n'y a pas de constructeurs car Lisp dispose d'un ramasse-miettes

3. Que ce passe-t-il si l'on défini une méthode sans que la fonction générique correspondante existe :

☐ La méthode est ignorée
■ La fonction générique est créée automatiquement
☐ CLOS déclenche un warning
☐ CLOS déclenche une erreur

4. Qu'est-ce que l'option `:allocation` d'un slot :
- ☐ Cette option permet d'allouer des slots sur la pile ou sur le tas
- ■ C'est l'option spécifiant si un slot est local à une instance, ou partagé par toutes les instances d'une classe
- ☐ Pour les classes ayant de nombreux slots, c'est l'option permettant de toucher plus d'allocations familiales
- ☐ Cette option permet de spécifier si un slot est initialisé par défaut, ou différemment à chaque instanciation

5. Comment s'appelle la class racine de toutes les classes dans CLOS :
- ■ `t`
- ☐ `standard-objet`
- ☐ Il n'y en a pas
- ☐ `standard-class`

6. ♣ Quelles formes de polymorphisme existent dans CLOS :
- ☐ Le masquage
- ☐ La surcharge
- ■ La ré-écriture
- ■ La généricité due au typage dynamique

7. De quelle couche logicielle de CLOS `make-instance` fait-elle partie :
- ☐ 3
- ■ 2
- ☐ 4
- ☐ 1

8. Qu'est-ce qu'une combinaison de méthodes :
- ☐ Le chaînage produit par des appels consécutifs à `call-next-method`
- ☐ La tenue vestimentaire officielle que toute méthode doit enfiler avant de se mettre au travail
- ■ Une manière d'utiliser une ou plusieurs méthodes applicables pour produire le résultat final d'un appel générique
- ☐ L'ensemble des *around*, *before*, *after* et méthodes primaires

9. Qu'est-ce qu'un `eql specializer` :
- ☐ La combinaison built-in qui renvoie `t` si toutes les méthodes applicables retournent la même valeur
- ☐ L'un des nombreux prédicats d'égalité de Lisp
- ☐ La spécialisation d'un slot à un type statique particulier
- ■ La spécialisation d'un argument de fonction générique sur une valeur particulière plutôt que sur une classe

10. Comment créer une méta-classe dans CLOS :
- ☐ En utilisant l'option `:metaclass` de `defclass`
- ■ En sous-classant `standard-class`
- ☐ En utilisant la macro `defmeta`
- ☐ En sous-classant `standard-object`

11. ♣ Si une classe utilisateur ne spécifie aucune super-classe :
- [x] Elle hérite quand même de `t`
- [x] Elle hérite quand même de `standard-object`
- [ ] Elle hérite quand même de `standard-class`
- [ ] Elle est à la racine de sa hiérarchie

12. Quel mécanisme Lispien remplace la surcharge des langages plus classiques :
- [x] Une sémantique plus expressive d'appels de fonction
- [ ] Le fait que les méthodes soient externes aux classes
- [ ] Aucun, car on peut très bien surcharger en Lisp
- [ ] Les multi-méthodes

13. Qu'est-ce qu'un `reader` dans CLOS :
- [ ] Une macro servant juste de wrapper à `slot-value`
- [ ] Une fonction permettant d'introspecter le contenu d'une classe
- [x] Une fonction générique d'accès en lecture à un slot
- [ ] La fonction de parsing du code Lisp (le R de REPL)

14. De combien de couches logicielles CLOS est-il constitué :
- [ ] 1
- [ ] 4
- [ ] 2
- [x] 3

15. De quelle couche logicielle de CLOS `defclass` fait-elle partie :
- [ ] 4
- [x] 1
- [ ] 2
- [ ] 3

16. Qu'est-ce qu'un `initarg` dans CLOS :
- [ ] Le nom d'une classe servant à initialiser un objet à travers `make-instance`
- [x] Un paramètre optionnel passé à `make-instance`, permettant d'initialiser les slots d'un objet au moment de sa création
- [ ] Une fonction d'initialisation d'objet autre que celle fournie par CLOS par défaut
- [ ] Un argument d'initialisation de slot par défaut, utilisé quand le programmeur ne fourni par de valeur explicite

17. Pourquoi `slot-value` est-il considéré comme un mécanisme de bas niveau :
- [ ] Cette fonction ne permet pas de distinguer les accès en lecture ou en écriture
- [ ] Cette fonction ne permet pas de distinguer les slots publics ou privés
- [x] Il est nécessaire de connaître le nom du slot afin d'y accéder, donc on manque d'abstraction
- [ ] Cette fonction ne permet pas de distinguer les slots partagés ou classiques

18. ♣ Dans la combinaison de méthodes standard, quelles catégories de méthodes sont sujettes au chaînage automatique (sans besoin de `call-next-method`) :

- ☐ *around methods*
- ☐ Méthodes primaires
- ■ *after methods*
- ■ *before methods*

19. ♣ Parmi les systèmes suivants, quels sont les prédécesseurs de CLOS :

- ■ Flavors
- ☐ Common Objects
- ☐ MOP
- ■ Common Loops

20. Que fait la fonction `call-next-method` :

- ☐ Elle permet de passer des *before methods* aux méthodes primaires, ou des méthodes primaires aux *after methods*
- ☐ Elle appelle la prochaine fonction générique par ordre de définition dans la classe de l'objet concerné
- ☐ Si elle existe, elle appelle la méthode appropriée de la fonction générique `next`
- ■ Si elle existe, elle appelle la prochaine méthode dans ordre des méthodes applicables

21. Qu'est-ce qu'une `initform` dans CLOS :

- ☐ Une forme initialisée par un `initarg`
- ■ Une expression dont la valeur sert à initialiser un slot par défaut
- ☐ Une expression attribuant une valeur par défaut à un `initarg`
- ☐ Une expression optionnelle passé à `make-instance`, permettant d'initialiser les slots d'un objet au moment de sa création

22. Quel est le résultat d'un appel à `slot-value` sur un slot non-initialisé :

- ☐ `nil`
- ☐ `<unbound>`
- ☐ L'affectation automatique du slot à une valeur par défaut
- ■ Une erreur de type `unbound-slot`

23. Quel danger y a-t-il à changer dynamiquement la classe d'une instance :

- ■ Certaines méthodes valides deviennent inapplicables sur la nouvelle classe de l'instance
- ☐ Toute référence à l'instance devient aussitôt invalide
- ☐ Changer une instance de classe ?! Ça va pas la tête ?!
- ☐ Certains slots risquent de disparaître

24. Que signifie MOP :

- ☐ Modular Object Prototypes
- ■ Meta-Object Protocol
- ☐ Multiple Object Polymorphism
- ☐ Maximally Obfuscated Program

Fin de l'épreuve.