+1/1/60+

QCM **TEST**

Introduction à la programmation orientée objet
21/10/2015

Nom et prénom :

.....

Groupe : .....

Vous apporterez un très grand soin à la présentation car elle interviendra dans la notation. Par exemple, les réponses très peu lisibles ou contenant du code non indenté ***seront considérées comme fausses****. Par ailleurs, la qualité du code proposé et la complexité des solutions interviendront dans la notation.*

Question 1 ♣ On souhaite écrire un jeu en Java dans un package `mixamatou`. Qu'est-il obligatoire ?

- ☐ compiler les fichiers avec la commande `javac -package mixamatou *.java`
- ☐ mettre tous les fichiers dans un répertoire `mixamatou`.
- ☐ mettre tous les fichiers sources dans un même `.jar`
- ☐ écrire `import mixamatou.*;` en début de tous les fichiers source
- ☐ écrire `package mixamatou;` en début de tous les fichiers source

Question 2 La classe `MixaCarte` représente un chat portant un `Chapeau`, une `Robe` et ayant un type de `Pommette`. Un constructeur par défaut de ces trois objets existe. La classe `MixaCarte` possède déjà un constructeur:

```
public class MixaCarte {

    private Chapeau hat;
    private Pommette cheek;
    private Robe dress;

    public MixaCarte(Chapeau c, Pommette p, Robe r) {
        hat = c;
        cheek = p;
        dress = r;
    }
 ...
```

Comment écrire le constructeur par défaut de `MixaCarte`?

- ☐ `public MixaCarte() { new this(); }`
- ☐ `public MixaCarte() { this(new Chapeau(),new Pommette(),new Robe()); }`
- ☐ `public MixaCarte() { return this(); }`
- ☐ `public void MixaCarte() { this(); }`
- ☐ `public MixaCarte() { this( Chapeau.this(), Pommette.this(), Robe.this()); }`

Question 3 La classe `Chapeau` comprend deux variables d'instances entières, l'une caractérisant la forme du chapeau (`int forme;`), et l'autre, la couleur (`int couleur;`). Deux chapeaux sont identiques si et seulement si leur forme et leur couleur sont identiques. Quelle est la signature de la méthode `isEqual` ?

- ☐ public boolean isEqual()
- ☐ public static boolean isEqual(Chapeau c1, Chapeau c2)
- ☐ public boolean isEqual(Chapeau c)
- ☐ public void isEqual()

Question 4 Deux cartes se ressemblent si elles possèdent un élément (`Chapeau`, `Pommette` ou `Robe`) en commun. Sachant que ces trois classes possèdent une méthode `isEqual`, écrire la méthode `ressemble` qui prend en entrée une `MixaCarte` et qui renvoie `true` si elle ressemble à la `MixaCarte` courante, `false` sinon. ☐0 ☐1 ☐2

```
.....

.....

.....

.....

.....
```

Question 5 Afin d'améliorer l'affichage d'un objet de la classe `Chapeau`, on souhaite définir un tableau constant `FORME` (le même pour tous les objets de la classe `Chapeau` contenant les chaines de caractères "`haut de forme`", "`melon`" et "`pointu`". Ecrire la ligne de code correspondante: ☐0 ☐1

```
.....

.....
```

Question 6 On suppose que de la même façon que pour le tableau `FORME`, un tableau `COULEUR` est créé. Ecrire la méthode `toString` de telle sorte qu'un affichage possible soit: **chapeau melon de couleur rose**.

☐0 ☐1

.....

.....

.....

Question 7 La classe `Joueur` possède pour variable d'instance une `ArrayList` de `MixaCarte`, appelée `sesCartes`. Cette variable représente les cartes que le joueur courant possède. Quel est le code correct ?

- ☐ `private MixaCarte<ArrayList> sesCartes;`
- ☐ `private ArrayList<MixaCarte> sesCartes;`
- ☐ `public ArrayList sesCartes = new MixaCarte();`
- ☐ `private ArrayList(new MixaCarte()) sesCartes;`

Question 8 Dans la classe `Joueur`, écrire la méthode `possedeUneCarteQuiRessembleA` prenant en entrée une `MixaCarte c` et retournant `true` si le joueur possède une carte qui ressemble à `c`, `false` sinon. Il conviendra d'utiliser la méthode `ressemble` décrite dans une autre question.

☐0 ☐1 ☐2

.....

.....

.....

.....

.....

.....

.....

Question 9 La classe `Jeu` possède pour variable d'instance un tableau de `Joueur` appelé `lesJoueurs`. Ecrire le constructeur de la classe `Jeu` qui prend en entrée le nombre de joueurs sous forme d'un entier et qui remplit le tableau de joueurs en utilisant le constructeur par défaut de la classe `Joueur`. ☐0 ☐1 ☐2

.....

.....

.....

.....

.....

Question 10 La classe `Pioche` contient initialement l'ensemble des `MixaCarte` du Jeu sous forme d'une `ArrayList` qui est remplie dans le constructeur. Quand un joueur pioche une carte, il la prend sur le dessus du paquet (indice minimal dans la liste). Lorsqu'il remet une carte, il la remet sous le paquet (indice maximal dans la liste). Ecrire la méthode `piocheUneCarte` qui retourne la carte piochée (elle ne fera évidemment plus partie de la `Pioche`). ☐0 ☐1 ☐2

.....

.....

.....

.....

.....

Question 11 ♣ Quelles sont les caractéristiques *à éviter* dans une application logicielle?

- ☐ High coupling
- ☐ Low cohesion
- ☐ High cohesion
- ☐ Low coupling

Question 12 Il s'agit de jouer aux dés. Pour deux dés il y a 36 combinaisons possibles. Ce qui nous intéresse, c'est la *somme* de deux dés.

- Vous allez créer une classe `De` qui aura une structure de données référencée par la variable `combinaisons` qui va stocker les combinaisons qui donnent lieu à une somme spécifique, et qui sera indexée par la somme. Par exemple,

$$\begin{aligned}
\texttt{combinaisons.get(2)} &\longrightarrow (1,1)\\
\texttt{combinaisons.get(3)} &\longrightarrow (1,2),(2,1)\\
\texttt{combinaisons.get(4)} &\longrightarrow (1,3),(2,2),(3,1)\\
&\dots\\
\texttt{combinaisons.get(12)} &\longrightarrow (6,6)
\end{aligned}$$

  Interdit de tout faire "à la main" ; il faut obligatoirement utiliser une boucle.

- Justifiez votre choix de structure de données.

☐0 ☐1 ☐2 ☐3

Question 13 Pour le jeu de cartes (`cards`), il avait été demandé la méthode `Jeu#battre` qui était censée mélanger les cartes.
Comme vous ne faites pas confiance à la personne qui aurait développé cette méthode (avec raison), vous allez écrire une méthode pour vérifier si les cartes sont bien mélangées :

```
/**
 * @return true si les cartes sont bien melangees ; sinon false.
 */
public boolean testBattre(Jeu jeu) {
    // code a fournir pour verifier si le jeu est bien melange
}
```

☐0 ☐1 ☐2 ☐3