



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 ?

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

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



Question 3 La classe Robe comprend deux variables d'instances entières, l'une caractérisant le motif du textile (int motif;), et l'autre, la teinte (int teinte;). Deux robes sont identiques si et seulement si leur motif et leur teinte sont identiques. Quelle est la signature de la méthode isEqual ?

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

1/1

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

1/2

```
... public boolean ressemble (MixaCarte c) {  
    ... if (this.chapeau.isEqual(c.chapeau)) return true;  
    ... if (this.pommette.isEqual(c.pommette)) return true;  
    ... if (this.robe.isEqual(c.robe)) return true;  
    ... return false;  
}
```

+1/2

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

☒ 0 ☐ 1

0/1

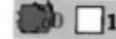
```
private static String[] FORME = new String[3];  
FORME[0] = "haut de forme", FORME[1] = "melon", FORME[2] = "pointu";
```

+1/2



+80/3/34+

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

```

public String toString() {
    String s = "";
    for (int i = 0; i < FORME.length; i++)
        for (int j = 0; j < COULEUR.length; j++)
            s += FORME[i] + " de couleur " + COULEUR[j];
    return s;
}

```

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 ?

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

1/1

Question 8 Dans la classe Joueur, écrire la méthode possedeUneCarteQuiRessembleA prenant en entrée une MixaCarte c et retournant le nombre de cartes en possession du joueur courant qui ressemble à c. Il conviendra d'utiliser la méthode ressemble décrite dans une autre question.



2/2

```

public int possedeUneCarteQuiRessembleA (MixaCarte m) {
    int nbr;
    for (MixaCarte c : sesCartes)
        if (c.ressemble(m))
            nbr++;
    return nbr;
}

```



+80/4/33+

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

2/2

```
public Jeu (int n) {  
    lesJoueurs = new Joueur[n];  
    for (int i = 0; i < n; i++)  
        lesJoueurs[i] = new Joueur();  
}
```

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 remetUneCarte qui ajoute à la pioche la MixaCarte donnée en entrée.

☐ 0 ☒ 1 ☐ 2

1/2

```
public void remetUneCarte (MixaCarte m) {  
    size = pioche.size();  
    pioche.add(m, size);  
    pioche.add(m);  
    joueur.delete(1);  
    non demande  
}
```

Question 11 ♣ Quelles sont les caractéristiques recherchées dans une application logicielle?

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

2/2



+80/5/32+

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,

```
combinaisons.get(2) → (1,1)
combinaisons.get(3) → (1,2),(2,1)
combinaisons.get(4) → (1,3),(2,2),(3,1)
...
combinaisons.get(12) → (6,6)
```

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

- Justifiez votre choix de structure de données.

0 1 2 3

0/3

```
public class De {
    ? tab[3][combinaisons] = new tab[3][36]; 36? 2 ≤ sum ≤ 12
    for(int i=0; i<6; i++)
        for(int j=0; j<6; j++)
            int tab[3] res = new tab[3];
            res[0] = i;
            res[1] = j;
            res[2] = i+j;
            combinaisons[i+j] = res;

    public tab[2] get(int n) { for(int i=0; i<36; i++)
        if (combinaisons[i][2] == n) {
            tab[0] = combinaison[i][0];
            tab[1] = combinaison[i][1];
            return tab;
        }
    }
}
```

returns a pair
not a list of pairs



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

0/3

