HUMAN booster

VOTRE SOLUTION COMPETENCE

Programmation orientée objet

Exercices

Table des matières

PARTIE 1 : UML

Dans les exercices suivant vous devez dessiner les diagrammes de classes correspondant aux situations énoncées.

Exercice 1.1

Un utilisateur à un nom, prénom et une date de naissance. Il peut manger, dormir et travailler.

Exercice 1.2

Une maison est composée d'un nombre de pièces, de portes et de fenêtres.
Elle se nettoie.

Exercice 1.3

Un ventilateur à une marque et est composé de pales. Il peut ventiler une pièce si on lui précise une vitesse.

Exercice 1.4

Une chaise à une couleur et un nombre de pieds. On peut savoir si elle est occupée et dans le cas contraire on peut l'occuper.

Exercice 1.5 (Interface)

Une voiture et une moto ont le même fonctionnement lié à la notion de véhicule, ils peuvent rouler à une vitesse donnée, freiner jusqu'à une vitesse donnée, avoir un accident.
Une moto peut avoir une selle pour le passager ou non.
Une voiture à une capacité de coffre.

Exercice 1.6 (Héritage)

Un élève et un formateur ont tous les deux des choses en commun, des connaissances en POO ou non, un prénom, un nom et un âge, ils rentrent le soir chez eux.

Exercice 1.7 (Composition)

Un avion à un nombre de hublot, il peut voler. Un réacteur peut être en panne ou non, et on peut savoir s'il est démarré ou non. Un avion à plusieurs réacteurs.

Exercice 1.8 (Attribut protégé)

Une ville à un nom un nombre d'habitant. Une capitale est une ville, elle peut définir son nom et à un monument.

PARTIE 2 : Classes et objets

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 2.1 à 2.4

Vous reprendrez les diagrammes UML **1.1** à **1.4** pour les réaliser en Java (l'implémentation des méthodes n'est pas demandée, seulement les déclarations).

PARTIE 3 : Introduction au JAVA

Dans les exercices suivant vous devez écrire les fonctions correspondantes aux situations énoncées.

Aide : Histoire de vous simplifier un peu la vie, voici comment transformer l'entrée utilisateur en int :
String monEntree = « 123 »;
Int monNombre = Integer.parseInt(monEntree) ;

Je le précise tout de même … Cela ne fonctionne que si la chaîne de caractère est entièrement numérique.

Exercice 3.1

Écrire une fonction permettant d'échanger les valeurs de deux entiers A et B, et ce quel que soit leur contenu préalable.

Exercice 3.2

On dispose de trois String A, B et C. Écrire une fonction transférant à B la valeur de A, à C la valeur de B et à A la valeur de C.

Exercice 3.3

Écrire un programme qui demande un nombre à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

Exercice 3.4

Écrire un programme qui demande son prénom à l'utilisateur, et qui lui réponde par un charmant « Bonjour » suivi du prénom.

On aura ainsi le dialogue suivant :

- *Machine* : Quel est votre prénom ?
- *Utilisateur* : Marie-Cunégonde
- *Machine* : Bonjour, Marie Cunégonde !

Exercice 3.5

Écrire une fonction qui lit le libellé d'un article, le prix HT d'un article, le nombre d'articles et le taux de TVA, et qui affichera le prix total TTC correspondant.

On aura ainsi le dialogue suivant :

- *Machine* : Quel est le libellé de l'article ?
- *Utilisateur* : Clé USB
- *Machine* : Quel est le prix HT de l'article ?
- *Utilisateur* : 10
- *Machine* : Combien avez-vous d'article ?
- *Utilisateur* : 3
- *Machine* : Quel est le taux de TVA applicable ?
- *Utilisateur* : 20
- *Machine* : 3 x Clé USB = 30 € HT soit 36 € TTC

Exercice 3.6

Écrire une fonction qui affiche si un lycéen est admis, recalé ou au rattrapage en fonction de la note saisis par l'utilisateur.

Exercice 3.7

Écrire une fonction demandant une durée en secondes à l'utilisateur, et qui la convertit en heures, minutes, secondes. On affichera ensuite le résultat.

Exercice 3.8

Un commerçant accorde une remise de 5 % pour tout achat d'un montant compris entre 100 et 500 € et 8 % au-delà. Écrire une fonction de calcul du montant de la remise sur un achat donné et qui retourne cette valeur.

Exercice 3.9

Écrire une fonction qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : « Plus petit ! », et inversement, « Plus grand ! » si le nombre est inférieur à 10.

Exercice 3.10

Écrire une fonction demandant à l'utilisateur de saisir un entier strictement positif et réalisant l'affichage ci-dessous :

```
>>>
Saisissez un nombre de lignes : 8
********
*******
******
*****
****
***
**
*
```

Exercice 3.11

Écrire une fonction demandant à l'utilisateur de saisir un entier strictement positif et réalisant l'affichage ci-dessous :

Exercice 3.12

Écrire une fonction qui déclare un tableau de 9 notes, dont on fait ensuite saisir les valeurs par l'utilisateur. Elle retournera ensuite ce tableau.

Exercice 3.13

Écrire une fonction qui demandera à l'utilisateur combien de phrase il veut saisir. On fera ensuite saisir ces phrases à l'utilisateur et on les stockera.

Exercice 3.14

Écrire une fonction qui demandera à l'utilisateur deux chiffre entre 1 et 10. On utilisera les chiffres saisis comme taille d'un tableau de chaine de caractère à deux dimensions.
On initialisera toutes les cases de ce tableau avec la concaténation suivante : « Cette cellule correspond à la première dimension **i** et à la deuxième dimension **j**. » (on remplacera i et j par les index du tableau).
Ce tableau sera ensuite retourné.

Exercice 3.15

Écrire de deux façons (itérative et récursive) une fonction ayant pour paramètres un entier x et un entier n, et retournant la valeur de x puissance n.

Exemple : x = 2 et n = 3 → 2*2*2 = 8

PARTIE 4 : Interface et Héritage simple JAVA

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 4.1 & 4.2

Vous reprendrez les diagrammes UML **1.5** et **1.6** pour les réaliser en Java (l'implémentation des méthodes n'est pas demandée, seulement les déclarations).

PARTIE 5 : Visibilité protégé JAVA

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 5.1

Vous reprendrez le diagramme UML **1.8** pour le réaliser en Java (l'implémentation des méthodes n'est pas demandée, seulement les déclarations).

PARTIE 6 : Constructeur lors d'un héritage JAVA

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 6.1

Une personne à un nom et prénom, elle peut respirer et dormir. Un étudiant est une personne qui a en plus une adresse mail. Un étudiant a au minimum un nom et un prénom quand il est instancié. On pourra aussi avoir besoin de l'instancier avec un nom, prénom et une adresse mail.

PARTIE 7 : Classe abstraite JAVA

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 7.1

Un meuble à un nombre de planche, un nombre de vis et un nom. Une table est un meuble qui a un nombre de place et sur laquelle on peut manger. Un tabouret est un meuble qui a un nombre de pieds et que l'on peut occuper. Un meuble n'est pas assez spécialisé pour être utilisé.

PARTIE 8 : Polymorphisme JAVA

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 8.1

Un élève et un formateur ont tous les deux des choses en commun, ce sont des personnes. Ils des connaissances en POO ou non, un prénom, un nom et un âge et ils se déplacent.
Cependant, une personne pour se déplacer démarre sa Clio, le formateur démarre sa Ferrari et un élève fait ses lacets (On affichera en chaine de caractère l'action).

PARTIE 9 : Composition JAVA

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 9.1

L'avion à un nombre de hublot, il peut voler. Un réacteur peut être en panne ou non, et on peut savoir s'il est démarré ou non. Un avion à plusieurs réacteurs.

PARTIE 10 : Délégation JAVA

Dans les exercices suivant vous devez créer les classes correspondantes aux situations énoncées.

Exercice 10.1

Un cercle est défini par un objet Point qui est son centre et par un rayon qui est un entier.
Nous pouvons déplacer ce cercle un donnant de nouvelles coordonnées x, y pour définir la position du centre du cercle. Le déplacement du centre sera délégué au déplacement du Point qui définit le centre.

PARTIE 11 : Mini-jeu JAVA

Exercice 11.1

Le jeu consiste à découvrir par essais successifs un nombre sélectionner par hasard par le programme. Pour chaque essai, le joueur reçoit un message : "Le chiffre recherché est plus grand !", "Le chiffre recherché est plus petit !" ou "Bien joué ! le nombre à trouver était : 18, trouvé en 4 coups !".

La partie est finie quand le joueur a trouvé le nombre.

A la fin d'une partie, l'utilisateur doit indiquer s'il veut faire une nouvelle partie ou non.

Pour choisir un nombre au hasard on utilisera la méthode random de la classe Math qui retourne un réel (double) tiré au hasard et de manière uniforme dans l'intervalle [0 1].

Pour choisir un nombre au hasard, on utilisera la Random de la sorte :
Random rand = new Random();
rand.nextInt(100); → obtenir un nombre entre 0 et 100

Exemple d'une partie :

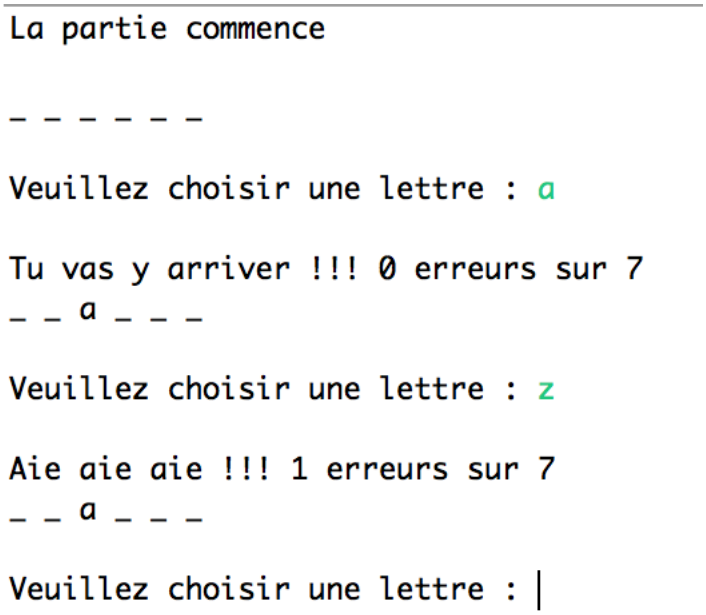
```
Donne moi un nombre : 10
Le chiffre recherché est plus grand !
Donne moi un nombre : 30
Le chiffre recherché est plus petit !
Donne moi un nombre : 20
Le chiffre recherché est plus petit !
Donne moi un nombre : 17
Le chiffre recherché est plus grand !
Donne moi un nombre : 18
Bien joué ! le nombre à trouver était : 18
tu l'as trouvé en 4 coups !
Voulez-vous faire une nouvelle partie ? (OUI/NON)
```

Exercice 11.2 (Le pendu)

Le Pendu est un jeu consistant à trouver un mot en devinant quelles sont les lettres qui le composent.

Nous définirons un tableau de mot directement dans le code avec une sélection aléatoire à chaque nouveau jeu.

Le joueur peut faire 7 erreurs avant de perdre.



Exemple d'un début de partie.

a) Définir le diagramme de classe du jeu
b) Réaliser le pseudo code du jeu
c) Exécution en Java
d) Bonus : Affichage du pendu

Exercice 11.3 (Morpion)

Le tic-tac-toe, aussi appelé « morpion », est un jeu de réflexion se pratiquant à deux joueurs au tour par tour dont le but est de créer le premier un alignement.

Deux joueurs s'affrontent. Ils doivent remplir chacun à leur tour une case de la grille avec le symbole qui leur est attribué ? Le gagnant est celui qui arrive à aligner trois symboles identiques, horizontalement, verticalement ou en diagonale.

Vous réaliserez un jeu de morpion opposant 2 joueurs qui pourront définir leur pseudo et les symboles qu'ils souhaitent utiliser.

Le jeu se déroulera en plusieurs manches définie par l'utilisateur. L'affichage se limitera à une sortie console.

```
Joueur 1
Veuillez entrer le nom du joueur : Gaspar
Entrer le symbol que voulez vous utiliser : X
Joueur 2
Veuillez entrer le nom du joueur : Baltazar
Entrer le symbol que voulez vous utiliser : O

Combien de manche voulez vous faire : 3

Le jeu peut commencer :
Gaspar vs Baltazar

1ère manche :

     |     |
_____|_____|_____
     |     |
_____|_____|_____
     |     |
     |     |


Joueur : Gaspar
Veuillez entrer la ligne visée : 0
Veuillez entrer la colonne visée : 1
     |  X  |
_____|_____|_____
     |     |
_____|_____|_____
     |     |
     |     |


Joueur : Baltazar
Veuillez entrer la ligne visée :
```

Exemple d'un début de partie.

a) Définir le diagramme de classe du jeu
b) Réaliser le pseudo code du jeu
c) Exécution en Java