

Exercices POO en Java pour Débutants

Niveau Débutant

Ce document contient trois exercices de programmation orientée objet (POO) en Java.

Exercice 1: Classe Voiture

Consigne:

1. Crée une classe Voiture avec les attributs suivants : marque (String), modèle (String), et année (int).
2. Ajoute un constructeur pour initialiser ces attributs.
3. Ajoute une méthode afficherDetails() qui affiche les détails de la voiture.
4. Dans une classe Main, crée deux objets Voiture et appelle la méthode afficherDetails() pour chacun.

Exemple d'exécution :

Voiture 1 : Peugeot 308, Année : 2020

Voiture 2 : Renault Clio, Année : 2018

Espace pour ton code:

Exercice 2: Classe CompteBancaire

Consigne:

1. Crée une classe CompteBancaire avec les attributs titulaire (String) et solde (double).
2. Ajoute un constructeur pour initialiser ces attributs.
3. Ajoute les méthodes suivantes :
 - déposer(double montant) : ajoute le montant au solde.
 - retirer(double montant) : soustrait le montant du solde (vérifie que le solde reste positif).
 - afficherSolde() : affiche le solde actuel.
4. Dans une classe Main, crée un objet CompteBancaire, dépose 1000 euros, retire 500 euros, et affiche le solde.

Exemple d'exécution :

Solde après dépôt : 1000.0

Solde après retrait : 500.0

Espace pour ton code:

Exercice 3: Classe Etudiant

Consigne:

1. Crée une classe Etudiant avec les attributs nom (String), âge (int), et notes (un tableau de double).
2. Ajoute un constructeur pour initialiser ces attributs.
3. Ajoute les méthodes suivantes :
 - calculerMoyenne() : calcule et retourne la moyenne des notes.
 - afficherDetails() : affiche le nom, l'âge et la moyenne des notes de l'étudiant.
4. Dans une classe Main, crée un objet Etudiant avec trois notes, calcule et affiche sa moyenne.

Exemple d'exécution :

Nom : Jean Dupont, Âge : 20 ans, Moyenne : 14.67

Espace pour ton code:

Exercices POO en Java pour Débutants Avancés

Niveau Débutant Avancé

Ce document contient cinq exercices de programmation orientée objet (POO) en Java avec une difficulté progressive.

Exercice 1: Héritage et Surcharge de Méthodes

Consigne:

1. Crée une classe de base `Animal` avec les attributs `nom` (`String`) et `age` (`int`), et une méthode `faireDuBruit()` qui affiche un message générique.
2. Crée une classe `Chien` qui hérite de `Animal` et surcharge la méthode `faireDuBruit()` pour afficher "Woof! Woof!"
3. Crée une classe `Chat` qui hérite de `Animal` et surcharge la méthode `faireDuBruit()` pour afficher "Miaou! Miaou!"
4. Dans une classe `Main`, crée des objets `Chien` et `Chat`, et appelle la méthode `faireDuBruit()` pour chacun.

Espace pour ton code:

Exercice 2: Classes Abstraites

Consigne:

1. Crée une classe abstraite `Forme` avec une méthode abstraite `calculerAire()`.
2. Crée des classes `Carre` et `Cercle` qui héritent de `Forme` et implémentent la méthode `calculerAire()`.
3. Dans une classe `Main`, crée des objets `Carre` et `Cercle`, et appelle la méthode `calculerAire()` pour chacun.

Espace pour ton code:

Exercice 3: Interfaces

Consigne:

1. Crée une interface `Volant` avec une méthode `voler()`.

2. Crée une classe Oiseau qui implémente l'interface Volant.
3. Dans une classe Main, crée un objet Oiseau et appelle la méthode voler().

Espace pour ton code:

Exercice 4: Polymorphisme et Collections

Consigne:

1. Utilise les classes Animal, Chien, et Chat de l'Exercice 1.
2. Crée une liste ArrayList d'objets Animal et ajoute des objets Chien et Chat.
3. Parcoure la liste et appelle la méthode faireDuBruit() pour chaque objet.

Espace pour ton code:

Exercice 5: Gestion des Exceptions dans les Classes

Consigne:

1. Crée une classe CompteBancaire avec un attribut solde (double) et une méthode retirer(double montant) qui lève une exception si le montant est supérieur au solde.
2. Dans une classe Main, utilise un bloc try-catch pour gérer l'exception levée par la méthode retirer().

Espace pour ton code:

Exercices POO en Java

Niveau Débutant à Intermédiaire

Ce document contient une série d'exercices POO en Java adaptés à votre niveau, avec une difficulté croissante.

Temps estimé et consignes sont fournis pour chaque exercice.

Exercice 1: Création de Classes et d'Objets

Consigne: Crée une classe `Personne` avec les attributs `nom (String)` et `age (int)`.

Ajoute une méthode `sePresenter()` qui affiche le nom et l'âge de la personne.

Dans une classe `Main`, crée deux objets `Personne` et appelle la méthode `sePresenter()` pour chacun.

Temps estimé: 15 minutes

Espace pour ton code:

Exercice 2: Constructeurs et Méthodes

Consigne: Crée une classe `Voiture` avec les attributs `marque (String)`, `modele (String)`, et `annee (int)`.

Ajoute un constructeur pour initialiser ces attributs et une méthode `afficherDetails()` pour afficher les détails de la voiture.

Dans une classe `Main`, crée deux objets `Voiture` et appelle la méthode `afficherDetails()` pour chacun.

Temps estimé: 20 minutes

Espace pour ton code:

Exercice 3: Héritage

Consigne: Crée une classe de base `Animal` avec les attributs `nom (String)` et `age (int)`, et une méthode `faireDuBruit()` qui affiche un message générique.

Crée une classe `Chien` qui hérite de `Animal` et surcharge la méthode `faireDuBruit()` pour afficher "Woof! Woof!"

Dans une classe `Main`, crée un objet `Chien` et appelle la méthode `faireDuBruit()`.

Temps estimé: 25 minutes

Espace pour ton code:

Exercice 4: Classes Abstraites et Interfaces

Consigne: Crée une classe abstraite `Forme` avec une méthode abstraite `calculerAire()`.

Crée une classe `Carre` qui hérite de `Forme` et implémente la méthode `calculerAire()`.

Dans une classe `Main`, crée un objet `Carre` et appelle la méthode `calculerAire()`.

Temps estimé: 30 minutes

Espace pour ton code:

Exercice 5: Polymorphisme et Collections

Consigne: Utilise les classes `Animal` et `Chien` de l'Exercice 3.

Crée une liste `ArrayList` d'objets `Animal` et ajoute des objets `Chien`.

Parcours la liste et appelle la méthode `faireDuBruit()` pour chaque objet.

Temps estimé: 35 minutes

Espace pour ton code:

Exercice 6: Gestion des Exceptions dans les Classes

Consigne: Crée une classe `CompteBancaire` avec un attribut `solde` (double) et une méthode `retirer(double montant)` qui lève une exception si le montant est supérieur au solde.

Dans une classe `Main`, utilise un bloc `try-catch` pour gérer l'exception levée par la méthode `retirer()`.

Temps estimé: 30 minutes

Espace pour ton code: