|  |
| --- |
| **Nom :**  **Prénom :** |
| **Consignes :**   * **Travail en monôme**😉 * **Ecrivez votre réponse en dessous de chaque exercice. Si votre code est réparti sur plusieurs fichiers, écrivez le nom du fichier au-dessus de votre code.** * **Pensez à commenter votre code.** * **Une fois tous les exercices terminés, n’oubliez pas de remettre votre devoir.** * **Travail évalué. Des points faciles à prendre … Alors jouez le jeu de l’objet !** * **Pour ceux qui le souhaite, des rappels sur le Java seront donnés en cours.** * **Enjoy !** |

**Exercice 1 : « final »**

**Exercice 2 : Classe et méthode abstraite**

**Exercice 3 : « interface »**

**Exercice 4 : méthode statique**

**Exercice 5 : variable statique**

**Exercice 6 : static final**

**Exercice 1 : « final »**

* **Reprendre l’exercice sur le tableau polymorphe.**
* **Ajouter deux variables d’instance uniques à une classe.**
  + **« gravité » de type double (par défaut = 1.).**
  + **« astre » de type String (par défaut = « Terre »).**
* **Mettre à jour les méthodes « affichage » en ajoutant « gravité » et « astre ».**
* **Mettre à jour les constructeurs en ajoutant « gravité » et « astre ».**
* **Mettre à jour les méthodes « decollage »** 
  + **Calcul : position=((poussee\*duree\*duree\*100)/(masse+carburant+autre masse))/gravite.**
* **A l’appel du programme :**
  + **Tester la gravité de la Lune (~0,16) pour tous les types de fusée**
  + **Tester la gravité de Mars (~0,38) pour tous les types de fusée**
  + **Tester la gravité de Jupiter (~2,53) pour tous les types de fusée**
  + **Tester la gravité de la Terre (1) pour tous les types de fusée**

**Exercice 2 : Classe et méthode abstraite**

* **Reprendre le programme de l’exercice précédent.**
* **Faire de Fusee une classe « modèle » que l’on ne peut pas instancier.**
* **Obliger les développeurs à redéfinir la méthode « decollage » dans les classes dérivées.**

**Exercice 3 : « interface »**

* **Reprendre l’exercice précédent.**
* **Créer un type de Fusée pouvant emmener du fret et des passagers**

**Exercice 4 : méthode statique**

* **Créer une classe *MesMaths* contenant une méthode statique *maxInt* qui prend en paramètre deux entiers et qui retourne le plus grand des deux.**
* **Essayer cette méthode en affichant le résultat de maxInt(3,2)**

**Exercice 5 : variable statique**

* **Reprendre l’exercice précédent**
* **A l’aide d’une variable d’instance public de la classe MesMaths, compter le nombre d’utilisation de la méthode maxInt.**
* **Tester en affichant le résultat de maxInt(3,2), maxInt(1,2) et maxInt(5,2) et le nombre d’utilisation à chaque appel.**

**Exercice 6 : static final**

* **Reprendre l’exercice précédent**
* **Créer une « constante » de type double dans la classe MesMaths PI\_APPROX avec comme valeur une approximation du nombre**
* **Dans la fonction main, à l’aide de cette constante, calculer la circonférence d’un cercle de rayon 1,5 cm.**