```
 _____  _____   ____       _ ______ _______    ____  _____  ____ _____ _______ ______
|  __ \|  __ \ / __ \     | |  ____|__   __|  / __ \|  __ \|  _ \_   _|__   __|  ____|
| |__) | |__) | |  | |    | | |__     | |    | |  | | |__) | |_) || |    | |  | |__
|  ___/|  _  /| |  | |_   | |  __|    | |    | |  | |  _  /|  _ < | |    | |  |  __|
| |    | | \ \| |__| | |__| | |____   | |    | |__| | | \ \| |_) || |_   | |  | |____
|_|    |_|  \_\\____/ \____/|______|  |_|     \____/|_|  \_\____/_____|  |_|  |______|


 ____              __
|  _ \            \_\
| |_) | __ _ _ __ ___ _ __ ___    ___
|  _ < / _` | '__/ _ \ '_ ` _ \ / _ \
| |_) | (_| | | |  __/ | | | | |  __/
|____/ \__,_|_|  \___|_| |_| |_|\___|
```

Ce projet consiste a simuler le deplacement d'un objet soumis a une gravité dans un univers en

2d.le sujet sera répartie en 3 parties:

-création des classes du projet

-simulation du déplacement de l'objet

-appliqué la gravité d'une planete dans l'equation de déplacement

chaqu'une de ces parties ont des consignes et contraintes vis a vis de leur gain de point.

points de chaque partie:

-partie 1 _classe_: 6 point

--detail:

-respect des regles: 4 points

-création des tables objet, planete, position, simulation, constante: 2 points

-partie 2 _deplacement_: 7 points

--detail:

-implémentation de la boucle de simulation: 2 points

-création du vecteur direction: 2 points

-modification de la position en relation avec le vecteur: 3 points

-partie 3 _gravité_: 7 points

--detail:

-calcul de la distance entre l'objet et la planete: 2 points

-application de la formule de la gravité: 3 points

-deplacment en fonction de la gravité: 2 points

-malus

--detail:

-noms des variables pas claires: -1.5 points

-non respect des demandes de rendu: -2 points

-code non commenté: -1.5 points

-Bonus

--detail:

-visualisation en dehors du terminal: 3 points

le rendu devra etre fais sur gitea, dans un nouveau repertoire. Le namespace et nom du repertoire

de rendu devront etre votre "%pseudo_ytrack%_orbite" (exemple: pchesneau_orbite).

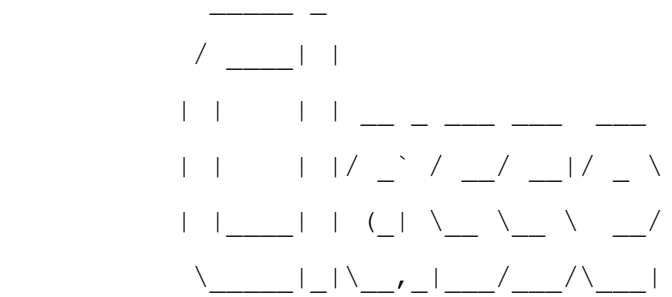
Si un exercice rendu non commenté, reussi ou non, comporte des ressemblances que nous considéront

comme flagrante avec des ressources en ligne, ou avec des exercices rendu par d'autres

étudiants, alors il sera considéré comme de la triche.

dernier commit autorisé le 15 janvier a 23h59

Toute triche resultera en a 0 pour l'etudiant. Non négociable.



créer les classes objet, planete, position, simulation, constante.

regles:

tout attribut doit etre privée.

tout attribut doit avoir un geteur si il a vocation a etre récupéré dans une autre classe.

tout attribut doit avoir un seteur.

toutes les classes non-abstract doivent avoir un constructeur au complet.

toutes méthodes doivent etre ecris en camelCase.

toutes méthodes doivent etre privée par defaut, et public si elle a vocation a etre

récupéré dans une autre classe.

ces regles sont valable pour tout le projet.

position:

-a une valeur x et y correspondant a la position dans l'espace.

-a une fonction affichePosition, qui return les données x et y dans une string sous cette

forme:

position: x:%x%km

y:%y%km

objet:

-a une position.

-a un poid.

planete:

-a un diamètre.

-hérite d'un objet.

-ne peut pas etre déplacé apres avoir été instancié (sa position)

constante:

-n'a rien pour l'instant.

-ne peut pas etre instancié.

-tout son contenue devras etre accessible a tout moment.

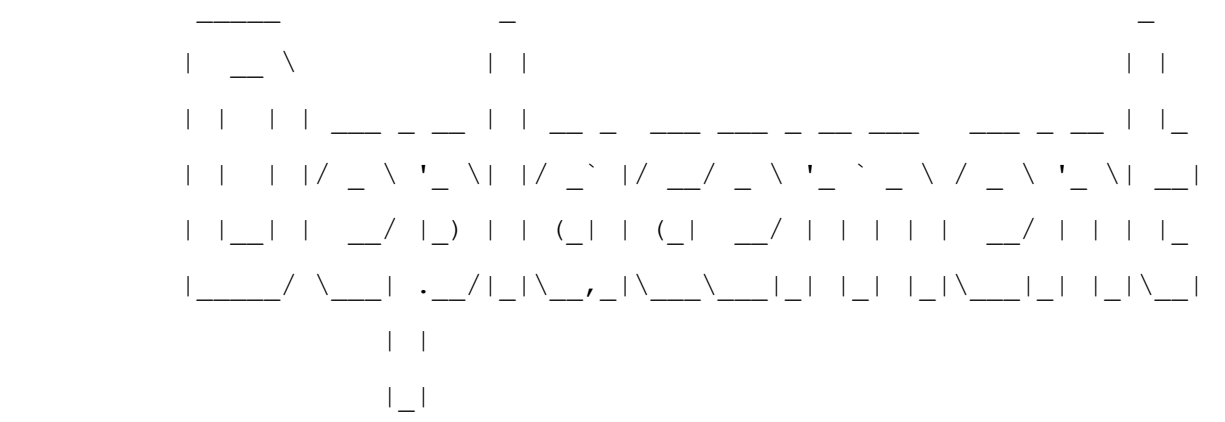
simulation:

-a une planete

-a un objet

-a un angle de lancer (en °)

-a une vitesse (m/s)



dans simulation, créer la méthode startSimulation.

-elle ne prend aucun paramètre

-sera lancer par le main pour lancer une simulation

dans constante, créer la méthode pour calculer un vecteur direction grace a une position,

un angle et a une vitesse.

-l'angle correspond a l'angle crée par la droite dirigeant le vecteur direction initiale et

l'axe des abscysse.

dans simulation, simuler le déplacement dans une boucle

-nous prendrons en compte aucune friction

-afficher les données sous cette forme:

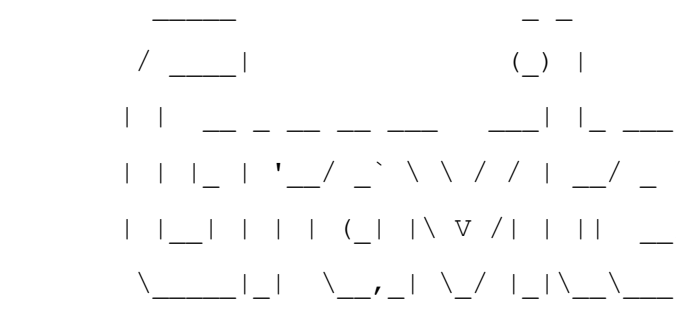
position: x:%x%km

y:%y%km

vitesse: %vitesse%ms

-l'affichage dois etre en temps reel.

dans le main, laisser les valeurs utilisées pour tester votre simulation.



dans constante, créer un attribut stockant la constante de gravité.

dans constante, créer la méthode pour calculé la distance entre 2 coordonées.

dans constante, créer la méthode pour calculé la force gravitationelle grace a la distance

et le poids des 2 objets.

dans simulation, implémenter la modification du vecteur direction en fonction de la gravité

dans la boucle de simulation.

-si l'objet en deplacement touche la planete, arreter la simulation.

-afficher les données sous cette forme:

position: x:%x%km

y:%y%km

vitesse: %vitesse%ms

hauteur: %hauteur%km

-l'affichage dois etre en temps reel, et coherent on fonction de la vitesse de

raffraichisement.

dans votre rendu, rendre un exemple de simulation avec des paramètre deja pré-remplis, avec

comme contraite:

-simuler un orbite, ou la ditance avec la planete oscille entre -10%|+10% de la distance

initiale.

bonus:

implémenter dans l'affichage la distance total parcouris