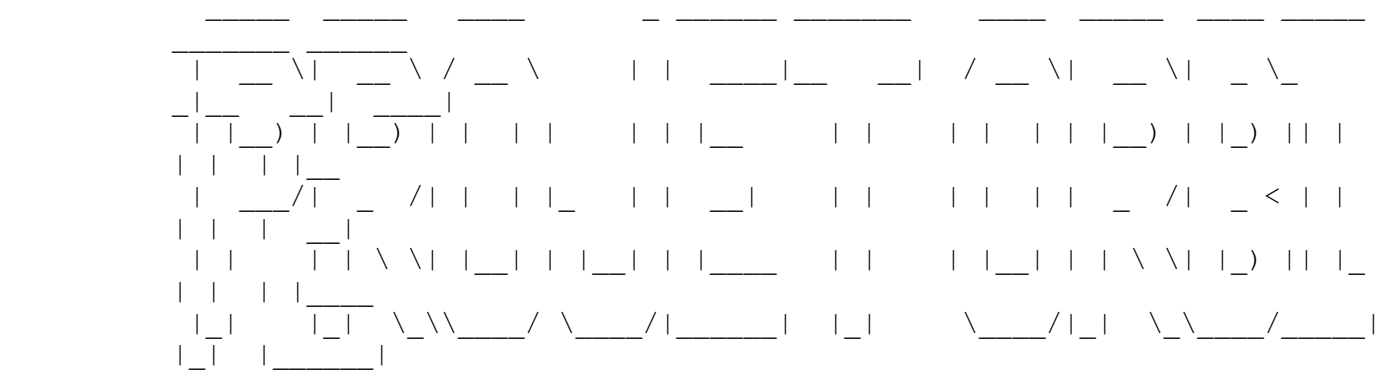


Ce projet consiste a simuler le deplacement d'un objet soumis a une gravitÃ© dans un univers en
2d.le sujet sera rÃ©partie en 3 parties:
-crÃ©ation des classes du projet
-simulation du dÃ©placement de l'objet
-appliquÃ© la gravitÃ© d'une planete dans l'equation de dÃ©placement

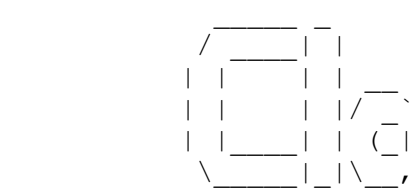
chaqu'une de ces parties ont des consignes et contraintes vis a vis de leur gain de point.
points de chaque partie:
-partie 1 _classe_: 6 point
--detail:
-respect des regles: 4 points
-crÃ©ation des tables objet, planete, position, simulation, constante: 2 points
-partie 2 _deplacement_: 7 points
--detail:
-implÃ©mentation de la boucle de simulation: 2 points
-crÃ©ation du vecteur direction: 2 points
-modification de la position en relation avec le vecteur: 3 points
-partie 3 _gravitÃ©_: 7 points
--detail:
-calcul de la distance entre l'objet et la planete: 2 points
-application de la formule de la gravitÃ©: 3 points
-deplacment en fonction de la gravitÃ©: 2 points
-malus
--detail:
-noms des variables pas claires: -1.5 points
-non respect des demandes de rendu: -2 points
-code non commentÃ©: -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 commentÃ©, reussi ou non, comporte des ressemblances que nous considÃ©ront
comme flagrante avec des ressources en ligne, ou avec des exercices rendu par d'autres
Ã©tudiants, alors il sera considÃ©rÃ© comme de la triche.

dernier commit autorisÃ© le 15 janvier a 23h59

Toute triche resultera en a 0 pour l'etudiant. Non nÃ©gociable.

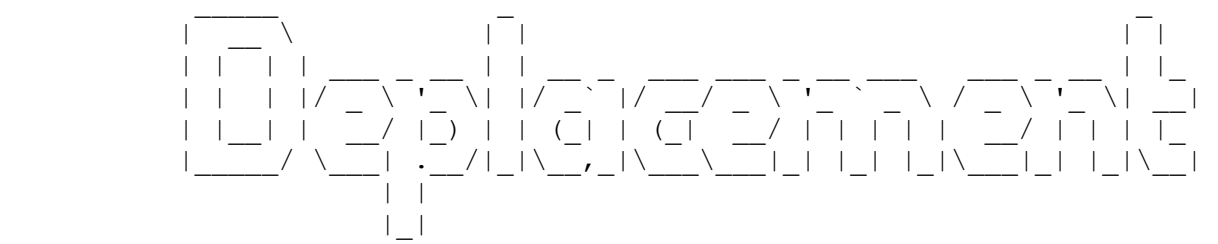


crÃ©er les classes objet, planete, position, simulation, constante.

regles:
tout attribut doit etre privÃ©e.
tout attribut doit avoir un geteur si il a vocation a etre rÃ©cupÃ©rÃ© dans une autre classe.
tout attribut doit avoir un seteur.
toutes les classes non-abstract doivent avoir un constructeur au complet.
toutes mÃ©thodes doivent etre ecris en camelCase.
toutes mÃ©thodes doivent etre privÃ©e par defaut, et public si elle a vocation a etre
rÃ©cupÃ©rÃ© 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 donnÃ©es 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 diamÃ¨tre.
-hÃ©rite d'un objet.
-ne peut pas etre dÃ©placÃ© apres avoir Ã©tÃ© instanciÃ© (sa position)
constante:
-n'a rien pour l'instant.
-ne peut pas etre instanciÃ©.
-tout son contenue devras etre accessible a tout moment.
simulation:
-a une planete

-a un objet
-a un angle de lancer (en Â°)
-a une vitesse (m/s)

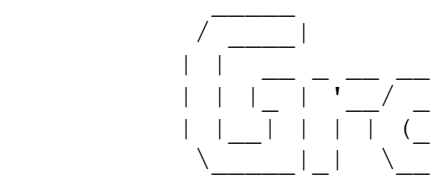


dans simulation, crÃ©er la mÃ©thode startSimulation.
-elle ne prend aucun paramÃ¨tre
-sera lancer par le main pour lancer une simulation

dans constante, crÃ©er la mÃ©thode pour calculer un vecteur direction grace a une position,
un angle et a une vitesse.
-l'angle correspond a l'angle crÃ©e par la droite dirigeant le vecteur direction initiale et
l'axe des abscysse.

dans simulation, simuler le dÃ©placement dans une boucle
-nous prendrons en compte aucune friction
-afficher les donnÃ©es 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 utilisÃ©es pour tester votre simulation.



dans constante, crÃ©er un attribut stockant la constante de gravitÃ©.
dans constante, crÃ©er la mÃ©thode pour calculÃ© la distance entre 2 coordonÃ©es.
dans constante, crÃ©er la mÃ©thode pour calculÃ© la force gravitationelle grace a la distance
et le poids des 2 objets.

dans simulation, implÃ©menter la modification du vecteur direction en fonction de la gravitÃ©
dans la boucle de simulation.
-si l'objet en deplacement touche la planete, arreter la simulation.
-afficher les donnÃ©es 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 paramÃ¨tre deja prÃ©-remplis, avec
comme contraite:
-simuler un orbite, ou la ditance avec la planete oscille entre -10%|+10% de la distance
initiale.

bonus:
implÃ©menter dans l'affichage la distance total parcouris