PPJ Zadanie dodatkowe

Drone for every occasion

Zadanie:

-Należy zaimplementować opisanie poniżej klasy oraz metody. W przypadku brakujących typów pól/metod należy użyć typów najlepiej odpowiadającym w danym miejscu. Tworzenie dodatkowych pól oraz metod w celu usprawnienia działania programu oraz zwiększenia czytelności kodu jest dozwolone a nawet zachęcane.

Klasa Dron

-Pola:

-uniqueId
-name
-weight
-enginePower
-batteryLevel

-Metody:

-checkFlyParameters() – ma za zadanie zwrócić true jeżeli enginePower jest większy od weight oraz batteryLevel jest większy od 0, false wpp.
-fly(distance) – ma za zadanie symulować lot drona. Należy odjąć distance od batteryLevel. Jeżeli batteryLevel jest mniejszy, należy wyświetlić odpowiedni komunikat.
-revEngine() – ma za zadanie wyświetlić „Vroom” tyle razy, ile wynosi enginePower/weight.

Klasa RacingDrone

-Pola:

-racingTeam
-positionInRanking

-Metody:

-race(Drone[] racers) – ma za zadanie zwrócić drona o największej wartości enginePower.
-revEngine() – ma za zadanie wyświetlić „Vroom” tyle razy, ile wynosi enginePower/weight a następnie „ZOOOOOM’.
-sortByPosition(RacingDrone[]) – metoda ma za zadanie posortować tablicę dronów po positionInRanking, a następnie ją zwrócić. Jeżeli 2 drony mają tą samą pozycję, należy ułożyć je według enginePower.

Klasa VampireDrone

-Pola:

-construcktor (zawsze równy „Bram Stoker”)
-isDoneBat (domyślnie równy false)

-Metody:

-drainEnergy(Drone) – jeżeli isTransfomred jest równe false, ma za zadanie pobrać połowę batteryLevel o drona dostarczonego jako argument i dodać ją do batterLevel naszego drona. W przeciwnym wypadku należy wyświetlić odpowiedni komentarz.
-turnIntoBatDrone() – ma za zadanie zmienić wartość pola isTransformed na true, a także zmniejszyć batteryLevel oraz weight o połowę.

Klasa ChristmasDrone

-Pola:

-Gift Gift

-Metody:

-deliverGift() – ma za zadanie wyświetlić „dostarczono <tu wstaw informacje o paczce> paczkę. A następnie zmienić wartość pola Gift na null. Jeżeli waga paczki + waga drona przekraczają enginePower lub Gift == null lub pole isReadyToBeDelivered paczki jest równe false, należy wyświetlić stosowny komunikat

Klasa Gift

-Pola:

-nameOfContent
-weight
-isReadyToBeDelivered

-Metody:

-prepare() – zmienia wartość isReadyToBeDelivered na true
-unpack() – zmienia wartość isReadyToBeDelivered na false, wypisując przy tym na konsolę zawartość prezentu.

Klasa DroneControleRoom

-Pola:

-Drone[] allDrones
-windPowerOutside

-Metody:

-countDronesThatCanFly – wypisz na konsolę liczbę dronów które są w stanie lecieć (enginePower>weight raz betteryLevel>0)

-chargeAllDrones – dodaj 20 do pola batteryLevel każdego drona w tablicy
-addNewDrone(Drone) – dodaj nowego drona do allDrones
-sortAllDrones – sortuje wszystkie drony po wadze (rosnąco) a następnie wypisuje je w posortowanej kolejności
-findMostPowerful – znajduje i wypisuje drona o największej wartości enginePower. Metoda powinna być rekurencyjna

Klasa ???

-Stwórz własną klasę dziedziczącą po klasie Drone. Dodaj do niej:

-przynajmniej 3 pola
-przynajmniej 3 metody (w tym jedną rekurencyjną.
-czym ciekawsze i bardziej złożone metody, tym lepiej.

Ogólne uwagi:

-Każda klasa powinna posiadać zaimplemenowaną metodę toString.
-To czy pole/metoda powinny być public/private czy static/non-static zostawiam do Państwa decyzji.
-Każda klasa powinna posiadać przynajmniej jeden konstruktor
-Klasa Main powinna zawierać metodę main wraz z tworzeniem obiektów i prezentacją zaimplementowanych metod.
-Wszystkie tablice powinny być dynamicznie powiększane/zmniejszane w razie potrzeby
-Pamiętaj o używaniu getterów i seterów (jeżeli są potrzebne)
-W razie pytań proszę o kontakt 😊