A feladat összefoglaló leírása

Egy MultiPlayer (többszereplős) játékban játékosok (`Player`) és járművek (`Vehicle`, `Car`, `Train`) vannak. A járműveknek van aktuális sebességük, mely a játék folyamán megváltozhat (`accelerate`, gyorsítás). A játékosoknak van virtuális pénzük, amellyel autókat vásárolhatnak. Egy játékosnak tetszőleges számú autója lehet. A járművek közül az autók (`Car`) rendelkeznek *természetes összehasonlítással* (`compareTo`), a játékosokon pedig értelmezünk egy speciális egyenlőségfogalmat (`equals`).

Alapfeladat (19 pont)

Készítsen `game.Player` néven osztályt, mellyel egy játékost reprezentálunk. Egy játékosnak van neve (`String`), IP-címe (`String`), virtuális pénze (`int`), valamint tartozik hozzá járművek (`Car`) egy listája (`ArrayList<>`). A `Player` osztálynak legyen egy (paraméteres) konstruktora, amely átveszi és eltárolja a játékos nevét, IP-címét és virtuális pénzét, ha az argumentumok megfelelőek. Ha az argumentumok nem megfelelőek, a konstruktor dobjon `IllegalArgumentException` kivételt.

- A név akkor megfelelő, ha nem `null` referencia.
- Az IP-cím akkor megfelelő, ha nem `null` referencia és a hossza pozitív, valamint nem tartalmaz fehér szóközt (whitespace: szóköz, tabulátor, sorvége).
- A virtuális pénz akkor megfelelő, ha nemnegatív.

Írja meg a járművek absztrakt `game.vehicles.Vehicle` ősosztályát. Minden megkonstruált jármű objektumnak van egyedi, létrehozás után már módosíthatatlan azonosítója (`id`, egész szám, a leszármazott osztályok számára is elérhető). Az első jármű azonosítója 0, a következőé 1, és így tovább. A járműveknek van aktuális sebessége (`currentSpeed`, `double` típusú rejtett adattag). A sebességhez írjon getter metódust. Írjon visszatérési érték nélküli `accelerateCurrentSpeed()` nevű metódust, amely paraméterként a sebességváltozás értékét kapja (lehet negatív is), és a metódus a jármű sebességét ellenőrzötten megváltoztatja: amennyiben a jármű sebessége 0 alá csökkenne, a metódus `game.utils.VehicleException` kivételt dob. A negatív paraméter jelenti a jármű lassítását. A metódust rejtse el a külvilág elől, a leszármazott osztályoknak viszont legyen elérhető, de nem felüldefiniálható. A `Vehicle` osztálynak legyen végül egy implementáció nélküli, nyilvános `accelerate()` metódusa is, amely a sebességváltoztatás mértékét kapja paraméterként, és nincs visszatérési értéke. Az `accelerate()` metódust a leszármazott osztályok fogják implementálni, amikor is különböző módon hívják meg az ősosztály `accelerateCurrentSpeed()` metódusát.

A `game.utils.VehicleException` kivétel egy ellenőrzött kivétel legyen, amelyet lehet paraméter nélkül és paraméteresen is konstruálni. Paraméteres konstruáláskor egy sztringet fogad paraméterként, és meghívja az ősosztály paraméteres konstruktorát.

A `game.vehicles.Train` és `game.vehicles.Car` osztályok származzanak a `game.vehicles.Vehicle` osztályból. Definiálják felül (implementálják) az ősosztálytól örökölt `accelerate()` metódust! A `Train` járművet csak lassan lehet fékezni:

amennyiben az `accelerate()` paramétere, `amount` negatív, akkor az `amount` 1/10-ével, máskülönben `amount`-tal hívja meg az ősosztály `accelerateCurrentSpeed()` metódusát.

A `game.vehicles.Car` osztály egy autót reprezentál. Egy autónak van maximális sebessége (`int`) és ára (`int`). Ezen jellemzők legyenek létrehozás után már módosíthatatlan adattagjai a `Car`-nak. A konstruktor vegye át és tárolja el ezen jellemzőket. Az árhoz tartozzon getter metódus. Írjon `toString()` metódust, amely tartalmazza a jármű egyedi azonosítóját (`id`), maximális sebességét és árát. Definiálja felül (implementálja) az ősosztálytól örökölt `accelerate()` metódust. Az autó sebessége csak akkor változzon meg, ha az autó új sebessége nem lépné át az autó maximális sebességét, egyébként ne történjen semmi.

Írjon `main.Main` névvel főprogramot. A főprogram osztálya tartalmaz egy `loadPlayerFromFile()` osztályszintű metódust, amely paraméterként egy játékosnevet vár, és beolvassa a "users/játékosnév.txt" szöveges állományból a játékos adatait. A metódus kódját alább *hibásan* közöljük. A hibákat (4 darab) meg kell keresni, és ki kell javítani!

```
public static Player loadPlayerFromFile(String playerName){
    File input = new File("users/" + playerName + ".txt");

    String data = null;
    try (BufferedReader bf = new BufferedReader(new FileReader(input))){
        String line = bf.readLine();
        data = line.split(" ");

        return new Player(playerName, data[1], data[2]);
    } catch (IOException e) {
        System.out.println("IO error occured: " + e.getMessage());
    }

    return null;
}
```

A `loadPlayerFromFile()` metódus beolvassa a "users/playerName.txt" fájlt (feltehető, hogy a fájl egyetlen egy sort tartalmaz). A fájl egyetlen sorában egy IP-cím (sztring) és egy virtuális pénzösszeg (`int`) áll szóközzel elválasztva (feltehető, hogy az IP-cím szóközt nem tartalmaz). A metódus a játékosnévvel és a beolvasott adatokkal megkonstruál egy játékos objektumot. Amennyiben a fájlban a pénzösszeg nem konvertálható egész számmá, akkor a metódus 0 kezdőpénzzel konstruál egy játékost.

A főprogramban **nem** kell olvasni a billentyűzetről vagy ciklusokat írni, a már megírt függvények működését szeretnénk kipróbálni beégetett értékekkel. A főprogramban konstruálja meg a következő
játékosokat: `Daniel`, `Peter`, `Richard`, `Tamas`, `Zorror` (használja a `users/` könyvtárban lévő txt fájlokat). Példányosítson legalább 5 autót (`Car`), amelyek között legalább 2-nek azonos a maximális sebessége és eltérő árúak.

Írjon unit teszteket (`tests.Tests` osztály) a következő elvárások tesztelésére.

- A `Player` konstruktora `null` név esetén `IllegalArgumentException` kivételt vált ki.
- A `Player` konstruktora negatív virtuális pénz esetén `IllegalArgumentException` kivételt vált ki.
- A `Player` konstruktora szóközt tartalmazó IP-cím esetén `IllegalArgumentException` kivételt vált ki.
- A `Player` konstruktor sikeres lefutása után a konstruált játékos virtuális pénze jól állítódott be.
- A `Car` osztály `accelerate()` metódusa pozitív és negatív paraméter esetén is jól működik.
- A `Car` osztály `accelerate()` metódusa a maximális sebesség átlépésénél nem csinál semmit.
- A `Car` osztály `accelerate()` metódusa `VehicleException` kivételt vált ki, ha a sebesség 0 alá csökkenne.

Emlékeztető a JUnit használatához:

- Az `org.junit.Assert` osztályt (és/vagy annak statikus metódusait, pl. `assertEquals`) kell importálni, valamint az `org.junit.Test` annotációt.
- A JUnit futtatása, ha a tesztesetek osztálya a névtelen csomagba tartozó `SimpleTest` osztály, a következő parancsokkal történik.

Windows:

```
javac -cp .;junit-4.12.jar;hamcrest-core-1.3.jar SimpleTest.java

java -cp .;junit-4.12.jar;hamcrest-core-1.3.jar org.junit.runner.JUnitCore SimpleT
est
```

Linux:

```
javac -cp .:junit-4.12.jar:hamcrest-core-1.3.jar SimpleTest.java

java -cp .:junit-4.12.jar:hamcrest-core-1.3.jar org.junit.runner.JUnitCore SimpleT
est
```

Egyenlőség (5 pont)

Írjon a `Player` osztályhoz `equals()` metódust úgy, hogy az megfeleljen az `equals()` metódussal szembeni elvárásoknak, valamint írjon

alkalmas `hashCode()` metódust. Két játékos objektumot akkor tekintünk egyenlőnek, ha megegyezik a nevük, ugyanannyi pénzük van, és ugyanazokat az autókat vásárolták meg (használja az `ArrayList` `equals()`-át). Az IP-cím egyezőségét tehát **nem** követeljük meg! Írjon unit teszteket azokra az elvárásokra, hogy különböző IP-című játékosokat tényleg azonosnak tekint az `equals()`, valamint, hogy egyenlő játékosok `hashCode()`-ja is egyenlő.

Összehasonlítás, rendezés (6 pont)

A `Car` osztály valósítsa meg a `Comparable` `interface`-t. Két `Car` objektum közül az a kisebb, amelyiknek maximális sebessége kisebb. Egyenlő maximális sebesség esetén az a kisebb, amelyik olcsóbb. Írjon unit tesztet arra, hogy tranzitív-e a `compareTo()`.

Egy játékos megvásárolhat tetszőleges számú autót, ha van elég virtuális pénze. Írjon `buyCar()` metódust a `Player` osztályhoz, amely `Car` referenciát fogad paraméterként, és nincs visszatérési értéke. Amennyiben a játékosnál elegendő pénz van, akkor a `Car` referenciát adja hozzá a játékos által birtokolt autók listájához, és csökkentse a pénzét az autó árával. A metódus ne engedje meg, hogy ugyanazt az autót több játékos is megvásárolhassa (ennek megoldásához egészítse ki a kódot esetlegesen szükséges adatagokkal/getterekkel). Sikertelen vásárlás esetén a metódus váltson ki `VehicleException` kivételt valamilyen informatív üzenettel. Írjon unit teszteket amikor a metódus `VehicleException` kivételelt dob.

Egy játékos által birtokolt autók listáját lehessen lekérdezni a `getSortedCars()` metódussal. A metódus térjen vissza az autók rendezett listájával, ahol a rendezési szempont az autók előbb megírt `compareTo()` metódusa. Vigyázzon arra, hogy a metódus ne szivárogtassa ki a `Player` osztály belső állapotát!

Példaként a főprogramban a Daniel játékos vásároljon legalább 3 autót, amelyekben legyen legalább 2 olyan, amelynek azonos a maximális sebessége. Írassa ki a képernyőre a Daniel játékos által birtokolt autók rendezett listáját.