Užduotis 1: Klasė `Studentas`

Sukurkite klasę `Studentas`, kuri turi šiuos atributus:

- `vardas` (string),
- `pazymiai` (sąrašas su pažymiais nuo 1 iki 10).

Reikalavimai:

1. Naudokite getterius ir setterius, kad:
   - `vardas` visada būtų ne tuščias ir būtų string tipo.
   - `pazymiai` visada būtų sąrašas su skaičiais nuo 1 iki 10.
2. Pridėkite statinį metodą `ar_pazymys_teisingas`, kuris patikrina, ar paduotas pažymys yra tarp 1 ir 10.
3. Pridėkite metodą `vidurkis`, kuris apskaičiuoja studento pažymių vidurkį.

Pavyzdys:

```
studentas = Studentas("Jonas", [8, 9, 7])
print(studentas.vardas)  # Jonas
print(studentas.vidurkis())  # 8.0
studentas.pazymiai = [10, 9, 11]  # Klaida: Pažymys turi būti nuo 1 iki 10
```

Užduotis 2: Klasė `Preke`

Sukurkite klasę `Preke`, kuri turi šiuos atributus:

- `pavadinimas` (string),
- `kaina` (float),
- `kiekis` (int).

Reikalavimai:

1. Naudokite getterius ir setterius, kad:
   - `pavadinimas` negalėtų būti tuščias.
   - `kaina` negalėtų būti neigiama.
   - `kiekis` negalėtų būti neigiamas.
2. Pridėkite statinį metodą `ar_kaina_teisinga`, kuris patikrina, ar kaina yra teigiamas skaičius.
3. Pridėkite metodą `bendra_kaina`, kuris apskaičiuoja bendrą prekių kainą (kaina * kiekis).

Pavyzdys:

```
preke = Preke("Obuolys", 1.2, 10)
print(preke.bendra_kaina())  # 12.0
preke.kaina = -5  # Klaida: Kaina negali būti neigiama
```

Užduotis 3: Klasė `Automobilis`

Sukurkite klasę `Automobilis`, kuri turi šiuos atributus:

- `marke` (string),
- `metai` (int),
- `rida` (int).

Reikalavimai:

1. Naudokite getterius ir setterius, kad:
   - `marke` negalėtų būti tuščias string.
   - `metai` turi būti tarp 1900 ir 2023.
   - `rida` negali būti neigiama.
2. Pridėkite statinį metodą `ar_metai_teisingi`, kuris patikrina, ar metai yra tarp 1900 ir 2023.
3. Pridėkite metodą `amzius`, kuris apskaičiuoja automobilio amžių (2023 - metai).

Pavyzdys:

```
automobilis = Automobilis("Toyota", 2010, 150000)
print(automobilis.amzius())  # 13
automobilis.metai = 2025  # Klaida: Metai turi būti tarp 1900 ir 2023
```

Užduotis 4: Klasė `BankoSaskaita`

Sukurkite klasę `BankoSaskaita`, kuri turi šiuos atributus:

- `savininkas` (string),
- `balansas` (float).

Reikalavimai:

1. Naudokite getterius ir setterius, kad:
   - `savininkas` negalėtų būti tuščias string.
   - `balansas` negalėtų būti neigiamas.
2. Pridėkite statinį metodą `ar_suma_teigiama`, kuris patikrina, ar suma yra teigiama.
3. Pridėkite metodus `prideti_pinigus` ir `nuskaityti_pinigus`, kurie atnaujina balansą.

Pavyzdys:

```
saskaita = BankoSaskaita("Jonas", 1000)
saskaita.prideti_pinigus(500)
print(saskaita.balansas)  # 1500
saskaita.nuskaityti_pinigus(2000)  # Klaida: Balansas negali būti neigiamas
```

Užduotis 5: Klasė `Trikampis`

Sukurkite klasę `Trikampis`, kuri turi šiuos atributus:

- `a`, `b`, `c` (kraštinių ilgiai).

Reikalavimai:

1. Naudokite getterius ir setterius, kad:
   - Visos kraštinės turi būti teigiamos.
   - Trikampis turi egzistuoti (trikampio nelygybė: a + b > c, a + c > b, b + c > a).
2. Pridėkite statinį metodą `ar_trikampis_egzistuoja`, kuris patikrina, ar trikampis gali egzistuoti.
3. Pridėkite metodą `plotas`, kuris apskaičiuoja trikampio plotą pagal Herono formulę.

Pavyzdys:

```
trikampis = Trikampis(3, 4, 5)
print(trikampis.plotas())  # 6.0
trikampis.c = 10  # Klaida: Trikampis su tokiomis kraštinėmis negali egzistuoti
```

Užduotis 6: Klasė `Knyga`

Sukurkite klasę `Knyga`, kuri turi šiuos atributus:

- `pavadinimas` (string),
- `autorius` (string),
- `puslapiu_skaicius` (int).

Reikalavimai:

1. Naudokite getterius ir setterius, kad:
   - `pavadinimas` ir `autorius` negali būti tušti string.
   - `puslapiu_skaicius` turi būti teigiamas skaičius.
2. Pridėkite statinį metodą `ar_puslapiai_teisingi`, kuris patikrina, ar puslapių skaičius yra teigiamas.
3. Pridėkite metodą `vidutinis_teksto_ilgis`, kuris priima papildomą parametrą `zodziu_skaicius` ir apskaičiuoja vidutinį teksto ilgį viename puslapyje.

Pavyzdys:

```
knyga = Knyga("Haris Poteris", "J.K. Rowling", 300)
print(knyga.vidutinis_teksto_ilgis(90000))  # 300 žodžių per puslapį
knyga.puslapiu_skaicius = -10  # Klaida: Puslapių skaičius negali būti neigiamas
```

Užduotis 7: Klasė `Darbuotojas`

Sukurkite klasę `Darbuotojas`, kuri turi šiuos atributus:

- `vardas` (string),
- `pareigos` (string),
- `atlyginimas` (float).

Reikalavimai:

1. Naudokite getterius ir setterius, kad:
   - `vardas` ir `pareigos` negali būti tušti string.
   - `atlyginimas` negali būti neigiamas.
2. Pridėkite statinį metodą `ar_atlyginimas_teisingas`, kuris patikrina, ar atlyginimas yra teigiamas.
3. Pridėkite metodą `metinis_atlyginimas`, kuris apskaičiuoja metinį atlyginimą.

Pavyzdys:

```
darbuotojas = Darbuotojas("Jonas", "Programuotojas", 2000)
print(darbuotojas.metinis_atlyginimas())  # 24000
darbuotojas.atlyginimas = -1000  # Klaida: Atlyginimas negali būti neigiamas
```

Užduotis 8: Klasė `Biblioteka`

Sukurkite klasę `Biblioteka`, kuri valdo knygas. Klasė turi šiuos atributus:

- `knygos` (sąrašas `Knyga` objektų, kuriuos apibrėžėte ankstesnėje užduotyje).

Reikalavimai:

1. Pridėkite metodus:
   - `prideti_knyga`: prideda naują knygą prie sąrašo.
   - `pasalinti_knyga`: pašalina knygą pagal pavadinimą.
   - `rasti_knyga`: grąžina knygą pagal pavadinimą.
   - `vidutinis_puslapiu_skaicius`: apskaičiuoja vidutinį visų knygų puslapių skaičių.
2. Naudokite getterius ir setterius, kad užtikrintumėte, jog `knygos` sąraše yra tik `Knyga` objektai.
3. Pridėkite statinį metodą `ar_knyga_teisinga`, kuris patikrina, ar objektas yra `Knyga` tipo.

Pavyzdys:

```
biblioteka = Biblioteka()
biblioteka.prideti_knyga(Knyga("Haris Poteris", "J.K. Rowling", 300))
biblioteka.prideti_knyga(Knyga("LoTR", "J.R.R. Tolkien", 500))
print(biblioteka.vidutinis_puslapiu_skaicius())  # 400
biblioteka.pasalinti_knyga("LoTR")
print(biblioteka.rasti_knyga("Haris Poteris"))  # Knyga: Haris Poteris, Autorius: J.K. Rowling,
Puslapiai: 300
```

Užduotis 9: Klasė `Universitetas`

Sukurkite klasę `Universitetas`, kuri valdo studentus ir kursus. Klasė turi šiuos atributus:

- `studentai` (sąrašas `Studentas` objektų, kuriuos apibrėžėte ankstesnėje užduotyje),
- `kursai` (sąrašas stringų, kurie nurodo kursų pavadinimus).

Reikalavimai:

1. Pridėkite metodus:
   - `prideti_studenta`: prideda naują studentą prie sąrašo.
   - `prideti_kursa`: prideda naują kursą prie sąrašo.
   - `priskirti_studenta_kursui`: priskiria studentą prie kurso (naudokite papildomą atributą `Studentas.kursai`, kuris yra sąrašas).
   - `gauti_studentus_pagal_kursa`: grąžina visus studentus, kurie lanko tam tikrą kursą.
   - `gauti_geriausius_studentus`: grąžina studentus, kurių vidurkis yra didesnis nei 9.
2. Naudokite getterius ir setterius, kad užtikrintumėte, jog `studentai` sąraše yra tik `Studentas` objektai, o `kursai` yra stringų sąrašas.
3. Pridėkite statinį metodą `ar_studentas_teisingas`, kuris patikrina, ar objektas yra `Studentas` tipo.

Pavyzdys:

```
universitetas = Universitetas()
universitetas.prideti_studenta(Studentas("Jonas", [9, 10, 10]))
universitetas.prideti_studenta(Studentas("Petras", [8, 7, 9]))
universitetas.prideti_kursa("Matematika")
universitetas.priskirti_studenta_kursui("Jonas", "Matematika")
print(universitetas.gauti_studentus_pagal_kursa("Matematika"))  # [Jonas]
print(universitetas.gauti_geriausius_studentus())  # [Jonas]
```