VILNIUS TECH

Gustas Pakenis Ekf-24

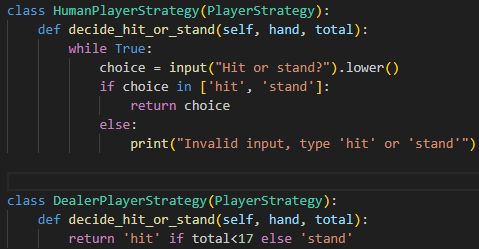
Objektinio programavimo kursinis darbas

Įvadas:

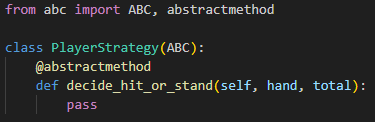
Mano programa - tai žaidimas Blackjack su netikrų pinigų lošimo funkcija. Kad pradėti žaidimą, reikia paleisti main.py programą ir žaidimas prasidės automatiškai (tai galima padaryti terminale, parašius “python main.py”). Tada bus paleista programa ir žaidėjo bus paprašyta įvesti netikrų pinigų statymo skaičių. Kiekvieną kartą paleidus programą žaidėjui duodami 100 pinigų. Tada prasideda standartinis Blackjack žaidimas. Žaidėjui pasakomos jo ir dealerio kortos, žaidėjui duodamas pasirinkimas arba paiimti dar vieną kortą (“Hit”) arba neiimti kortos (“Stand”). Jei žaidėjas perkopia 21 jis pralaimi. Žaidėjas gali imti kortas kol nusprendžia nebeimti kortos (“Stand”), prasideda kortų dalintojo (“dealerio”) eilė. Tada, kuris žaidėjas ar kortų dalintojas yra arčiau 21 laimi. Programos naudotojui reikia sekti instrukcijas parašytas ekrane, jeigu naudotojas praranda visus pinigus, programa yra sustabdoma ir uždaroma.

Reikalavimų įvykdymas:

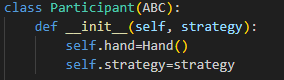
Polimorfismas: - kai metodas yra vykdomas skirtingai priklausomai nuo konkrečios klasės (ar duomenų tipo realizacijos)i. Pvz. priklausant nuo, kurios klasės buvo sukurtas objektas (HumanPlayerStrategy ar DealerPlayerStrategy) vykdomas metodas skiriasi nors metodo pavadinimas yra toks pat.

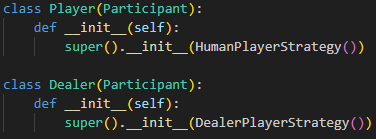


Abstrakcija - paslepiama sudėtinga logika po paprastomis sąsajomis, pvz. PlayerStrategy - abstrakti bazinė klasė, kurioje yra metodas decide\_hit\_or\_stand(...), nereikia žinoti, kaip tas sprendimas vyksta

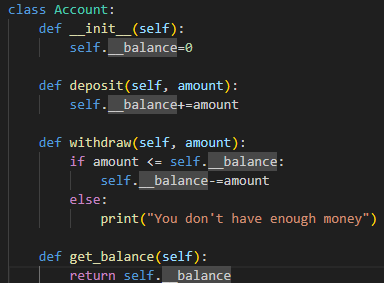


Pavaldėjimas - tai, kai naujos klasės paveldi senesnių metodus. Pvz. Klasę Participant paveldi klasė Player ir Dealer. Šios klasės naudoja Participant klasės konstruktorių \_\_init\_\_.



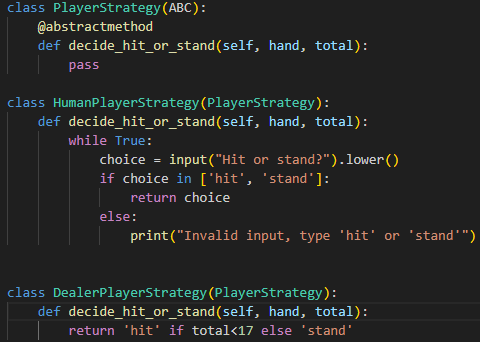


Enkapsuliacija - tai, kai paslepiami vidiniai klasės parametrai ir tam, kad juos pasiekt reikia naudoti metodus, pvz. klasėje Account yra privatus kintamasis \_\_balance, kad pasiekt \_\_balance naudojami metodai withdraw(), deposit(), get\_balance().



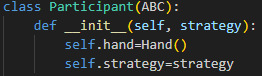
Dizaino šablonas (“Design pattern”):

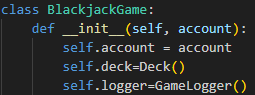
Pasirinkau strategijos dizaino šabloną. Strategijos šablonas apibrėžia algoritmų šeimą, kiekvieną jų apibendrina ir padaro juos pakeičiamus. Tai leidžia algoritmui skirtis nepriklausomai nuo jį naudojančių klientų. Patogiausia naudoti strategijos šabloną, kai yra keli algoritmai ar elgesiai, kurie gali būti pakeisti, tai yra jeigu klasė turi atlikti kažkokią užduotį skirtingais būdais, strategijos šablonas leidžia lengvai inkapsiulioti kiekvieną variaciją ir leidžia lengvą elgesio pakeitimą.



Kompozicija:

Kompozija - tai, kai objektas yra sukuriamas kitame objekte.

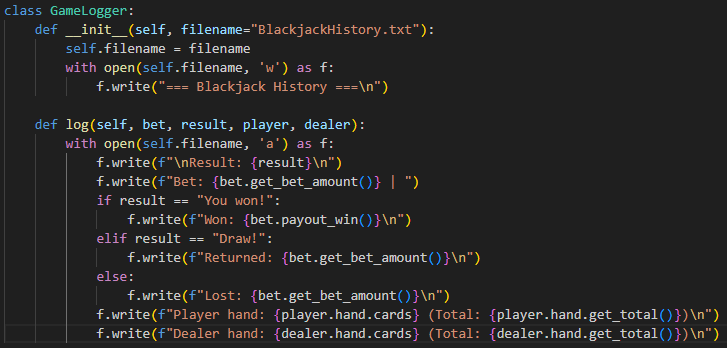




Pvz. Klasėje Participant yra sukuriamas objektas ‘hand’ iš klasės Hand(), Klasėje BlackjackGame yra sukuriami objektai deck ir logger. Šie objektai egzistuoja tik kituose objektuose, kur jie yra sukuriami.

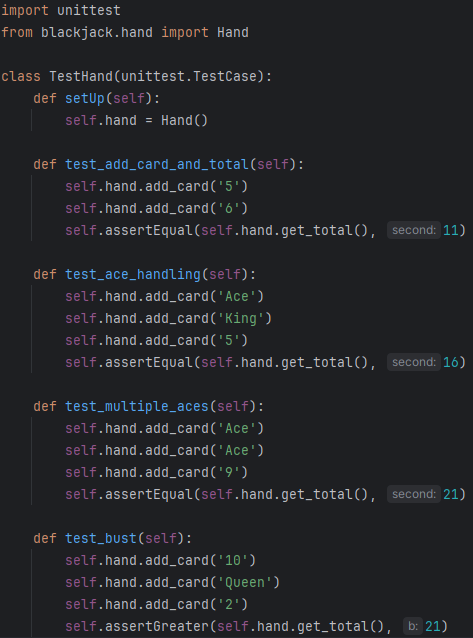
Skaitymas ir rašymas į failą:

Kode yra klasė GameLogger, kuri įrašo paskutinio žaidimo istoriją, tai yra kiekvieno mačo rezultatą (Laimėjimas ar pralaimėjimas), laimėtą/pralaimėtą pinigų sumą, žaidėjo ir kortų dalintojo (dealerio) turėtas kortas. Paleidžiant žaidimą iš naujo, failo turinys yra ištrinamas ir leidžiamas naujam žaidimui.



Testavimas:

Ištestuotos pagrindinės klasių funkcijos, naudojant unittest. Pvz. ištestuota klasė Hand pasirūpinant, kad esant skirtingoms kortų kombinacijoms, taip pat pratestuojant kraštutinius atvejus, tokius kaip 2 tūzai, gaunama tinkama totali kortų reikšmė ir įsitikinama, kad kortų reikšmių skaičiavimas yra teisingas.



Rezultatai ir apibendrinimas:

Rezultatai

* Buvo įgyvendinti visi 4 OOP principai. Tai leido kodui būti suprantamesniam ir švaresniam.
* Panaudotas Strategijos dizaino šablonas leido daugiau kintamumo tarp žaidėjo ir stalo (dealerio) logikos, kas ateity leistų ir naujų veikėjų logikos pridėjimą, pvz. Sukurti žaidėją, kuris žaistų už tave.
* Kodo suskirstymas į skirtingus modulius buvo vienas didesnių iššūkių, nes jis reikalavo daug dėmesio žiūrint į santykius tarp skirtingų klasių.
* Pagrindinės kodo funkcijos buvo patikrintos naudojant vienetų testus (unit tests)

Apibendrinimas:

Per kursinį darbą buvo sukurtas paprastas, pilnai funkcionuojantis “Blackjack” žaidimas naudojant pagrindinius objektinio programavimo principus. Gautas žaidimas yra žaidžiamas terminale, kuris leidžia žaidėjui priimti tinkamus sprendimus, turi sąskaitos valdymą ir žaidimo istorijos įrašymo galimybes. Ateityje programai galima suteikti grafinę sąsają, kelių žaidėjų galimybes ir modifikuoti roboto sprendimus. Taip pat įmanoma sukurti kitus kortų žaidimus panaudojant esamas klases.