Programavimo kursinis darbas

Minesweeper

Arno Mėlinausko EEf-23

1.

a. What is your application?

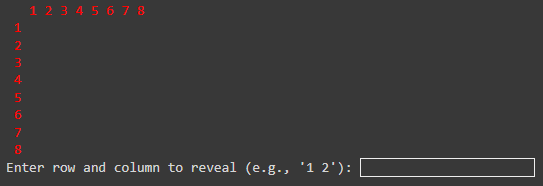
Aš parašiau minesweeper koda kuris leistu ji žaisti

b. How to run the program?

Programa galima zaisti per bet koki python copileri.

c. How to use the program?

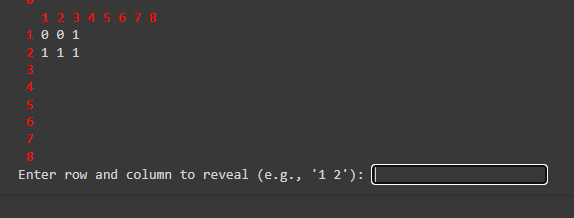
Programa naudoti yra paprasta. Paspaudus vykdyti koda iskris jum tokia lentele



Pasirinkus kurio norime skaiciaus eilute ir stulpeli irasome tokia tvarka:

Rasome pasirinkta eilutes skaiciu, spaudziame tarpa ir tada rasome pasirinkta stulpelio skaiciu.

Paspaudus 1 1 lentele dabar atrodo taip:



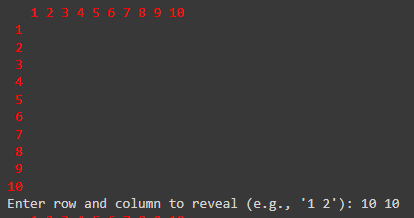
Taip galite toliau zaisti si zaidima.

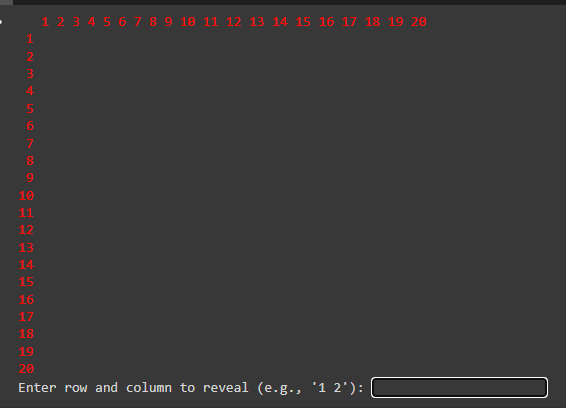
2.

Polymorphism :

Norėdami parodyti Polymorphism Minesweeper žaidimo kode, galime sukurti įvairių tipų lentas, kurios praplečia abstrakčią lentų klasę. Polymorphism leidžia mums naudoti vieną sąsają (šiuo atveju plokštės klasę) skirtingų tipų plokštėms reprezentuoti, o konkretus elgesys priklausys nuo poklasio įgyvendinimo.

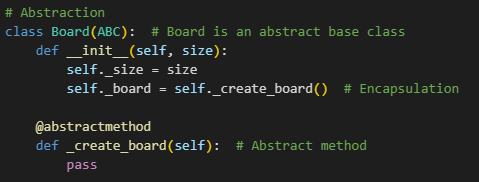
Naudodamas Polymorphism mano zaidimas gali atlaikyti skirtingu dydziu lentas:





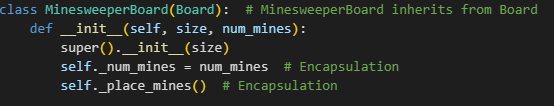
Abstraction: is one of the important principles of object-oriented programming. It refers to a programming approach by which only the relevant data about an object is exposed, hiding all the other details. This approach helps in reducing the complexity and increasing the efficiency of application development.

Šio kodo Abstraction demonstruoja Board Abstraction klasė, apibrėžianti įvairių tipų plokščių sąsają ir struktūrą, neįgyvendinant konkrečių detalių. Specifinį elgesį suteikia poklasiai (MinesweeperBoard ir SpecialMinesweeperBoard). Tai leidžia MinesweeperGame klasėje pakaitomis naudoti skirtingus lentų tipus, laikantis Abstraction principo.



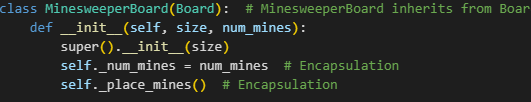
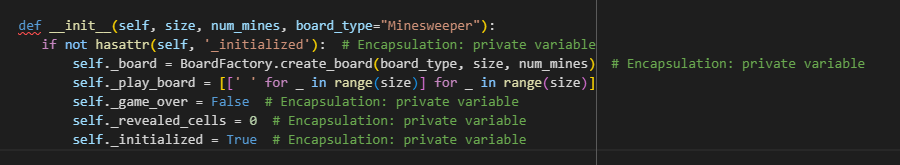
Inheritance: allows us to define a class that inherits all the methods and properties from another class.

Naudodami Inheritance, galime apibrėžti bendrą Board klasės struktūrą ir elgesį ir tada sukurti konkrečius plokščių tipus, kurie praplečia šią funkciją. Dėl to kodas yra labiau modulinis, daugkartinis ir lengviau prižiūrimas.



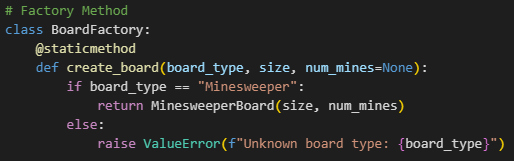
Encapsulation: is one of the main pillars on which the object-oriented programming paradigm is based.

Encapsulation šiame kode įgyvendinamas naudojant privačius kintamuosius ir metodus, kurie riboja prieigą prie objektų vidinės būsenos ir funkcionalumo. Viešieji metodai suteikia kontroliuojamą sąveikos su objektais būdą, užtikrinant, kad vidinė būsena būtų modifikuojama tik kontroliuojamu būdu. Tai padeda išlaikyti objektų vientisumą ir daro kodą patikimesnį ir lengviau prižiūrimą.



FactoryMethod: is a creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.

Naudojant gamyklinio metodo modelį, kodas pasiekia didesnį lankstumą ir išplečiamumą. Naujų tipų lentas galima lengvai pridėti nekeičiant esamos žaidimo logikos, laikantis atviro/uždarymo principo.

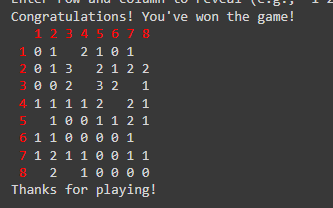


Singleton: is a design pattern that ensures a class has only one instance and provides a global point of access to it.

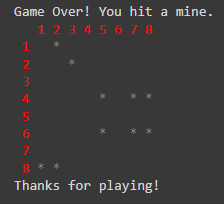
Singleton modelis šiame kode užtikrina, kad tik vienas MinesweeperGame klasės egzempliorius būtų sukurtas ir naudojamas visoje programoje. Tai pasiekiama naudojant privačios klasės kintamąjį \_instance, nepaisytą \_\_new\_\_ metodą ir \_\_init\_\_ metodą su \_inicializuota vėliava. Šis modelis yra naudingas valdant bendrus išteklius arba koordinuojant veiksmus visoje sistemoje.



Rezultatai:



Tinkamai apejus visas minas zaidimas sekmingai baigias ir restartavus koda galima pradeti nauja partija, bet pataikius ant minos



Gausite toki lentele kur parodo kur minos slėpėsi ir galite bandyti is naujo.

Conclusion:

Pamaciau kaip sunku yra programuoti, bet su draugu ir interneto pagalba viska galima padaryti. Manau si darba uzbaigiau sekmingai bent pagal savo programavimo pajegumus, o ateityje manau bus tikrai minimalus programavimas kokiem maziem darbelem