Disclaimer: Hogyha egy absztrakt függvényt (vagy bármilyen függvényt) egy leszármazott override-ol (jelen esetben mindenhol xd), akkor azt be kell fel kell tüntetni a leszármazottnál is. Pl, a Character move() függvényét fel kell tüntetni az Instructor-nál, és a Student-nél is (Rendszertervező diplomás mérnökember approved)  
  
Character

* - currentRoom : Room
* + move(Room, Room) : void
  + módosítás
* + newRound(int): void
  + végül picit átdolgoztam, pontosabban átneveztem az action függvényt, hogy a game controller ezt hívja meg, és majd ebben a függvényben lesz meghívva az action függvény
  + a ‘NewRound’ és ‘Student Start Round’ szekvencia diagramokból ki tudjátok venni, mire gondoltam

Student

* - transistor: Transistor
  + ez csak a pár egyik elemének eltárolására van, miután megvan a másik pár, bekerül mindkettő az inventoryba
* - inventorySize: int
  + inventory max mérete, ergo amennyi item belefér
* + getTransistor(): Transistor
* + dropItem(Item): void
* + rollDice(): int
  + a dobott értékkel tér vissza
* + transistorJump() : void

Room

* + isAccessible(Room) : boolean
* + addCharacter(Character): void
  + módosítás addStudent helyett
* +getCharacters(): list <Characters>
* + removeCharacter(Character): void
* + removeitem(Item): void
* + addItem(Item): void
* + addItems(list<Item>): void
* + getIems(): list<Item>
* + getNeighbours(): list<Room>
* + addNeighbours(list<Room>): void
* + removeNeighbours(list<Room>): void
* - sortItems(); ////////////////////////////////////////////////////////
* + onEntered(c: character): void
  + módosítás a paraméterben
* + getTransistor() : Transistor
* Meg szerintem bővíthetőség miatt ez legyen egy **absztrakt ősosztály**, és ennek **legyen 2 leszármazottja**:
  + Regular Room
  + Gas Room
* Ugyanitt, ne legyen isGasRoom változó, mert az típuslekérdezés, és az iit megbasz. Meg, ha az újításként új szobákat akarnának behozni, akkor nem lehet bővíteni, csak plusz változókkal, ami oop elveket sért (Open/Closed principle)

GameController

* - rooms: List<Room>
* - studentCount: int
  + am jelenleg nem nagyon tároljuk el a szobákat xd
* GameController(szobák listája, playerek listája)
  + átadni a konstruktorban függőségeket
    - am ezt dependency injection-nek hívják, tanultunk róla
* + splitRoom(Room): void
* + mergeRooms(Room, Room): void
* + addRoom(Room): void
* + removeRoom(Room): void
* + newRound(): void
* + endGame(): void
* + removePlayer(Student): void
* + play(): void
* + rollDice() : int

Item

* nem kell action() és use() függvény egyszerre
  + azok nem ugyanazt csinálják?
  + ha igen, akkor szerintem a use() maradjon, hogy ne keverjük össze a játékosok ‘action’-jével
  + ja és legyen egy ‘Student’ paramétere, hogy elérje a hallgatót
* canInstructorPickUp() : bool
  + függvény, hogy a logarlécet ne tudja már felvenni a tanár xd
    - alapjáraton default megvalósítása lehet csak annyi, hogy ‘return true;’ csak a logarlécnél legyen felülírva

Tranzisztor

* getPairsRoom(): Room
* pairing(Transistor): void

OnEvent

* onPickedUp(Student): void
  + módosítás
* onRoundStart(Student): void
  + módosítás
* onAttacked(Student): void
  + módosítás

**Jeges rant:**  Szóval, Szerintem a gamecontroller csak a köröket indítsa és kezelje, viszont azt hogy egy karakter mennyit dob az maga a Characteren belül legyen egy fgv.

Így nem kell mindig átadni a karakter osztálynak a gamekontroller intjét. **Javítsatok ki ha tévedek**, én így fogok doplgozni. És akkor így van a Gamecontrollerben egy newRound() fgv. ami annyit csinál, hogy végigmegy az összes charackteren és meghívja a rollDice() fgv.t ami a többit lekezeli.

Conclusion: nem így adtuk le a faszom dokumentációt so megcsinálom az alapján sorry.

Így viszont kell egy + setter ami elkerülhető lett volna.

**Gábor rant:** Egy jó kis Balage (.Y.) hiányzik a napjaimból. 🙁