Általános tudnivalók

Ebben az ismertetésben az osztályok, valamint a minimálisan szükséges metódusok leírásai fognak szerepelni. A feladatmegoldás során fontos betartani az elnevezésekre és típusokra vonatkozó megszorításokat, illetve a szövegek formázási szabályait. Segédfüggvények is létrehozhatók, a feladatban nem megkötött adattagok és elnevezéseik is a feladat megoldójára vannak bízva. Törekedjünk arra, hogy az osztályok belső reprezentációját a lehető legjobban védjük, tehát csak akkor engedjünk meg, és csak olyan hozzáférést, amelyre a feladat felszólít, vagy amit az osztályt használó kódrészlet megkíván!

A beadott megoldásnak működnie kell a mellékelt tesztprogrammal, de ez nem elégséges feltétele az elfogadásnak. Törekedjünk arra, hogy a megírt forráskód kellően általános és újrafelhasználható legyen!

Használható segédanyagok: Java dokumentáció, legfeljebb egy üres lap és toll. Ha bármilyen kérdés, észrevétel felmerül, azt a felügyelőknek kell jelezni, *NEM* a diáktársaknak!

A feladat összefoglaló leírása

A feladat a Jégvarázs (Frozen) nevű kalandjáték megvalósítása.

Arendelle királyságban két kis hercegnő lakik: Elza és Anna. Az idősebbik hercegnő, Elza különleges képességgel rendelkezik: parancsolni tud a hónak és a jégnek. Mikor az emberek megismerik ezt a nem mindennapi képességét, boszorkánynak tartják. Emiatt Elza úgy dönt, hogy elszökik, s ezzel Arendelle királyságát az örök tél országává változtatja. Anna a nővére keresésére indul. Útközben misztikus trollokkal (félelmetes lényekkel) találkozik, és a hóviharral is meg kell küzdenie. Sikerül-e eljutnia Elza jégpalotájába, vagy a szigorú tél és a trollok keresztülhúzzák a számítását?

A programhoz tartozik egy tesztelő (Test.java, DummyPlayer.java és DummyKingdom.java), amely az egyes osztályok funkcionalitását teszteli, illetve a várható pontszámot mutatja.

A feladat részletes ismertetése

Colour (2 pont)

A `game.utils.Colour` felsorolási típus segítségével a játékban szereplő színeket reprezentáljuk.

- Vegyük fel a játékban szereplő színeket (`COLOURLESS`, `YELLOW`, `BROWN`, `GREY`, `BLUE`, `RED`, `PINK`, `GREEN`, `PURPLE`) tartalmazó felsorolási típust! *(1 pont)*
- Definiáljuk felül az `Object`-től örökölt `toString` metódust! A felüldefiniált `toString` metódus az egyes színeket szöveges formában adja vissza. Pl. A COLOURLESS szín esetén a "colourless" szöveget. *(1 pont)*

Gem *(7 pont)*

A `game.items.Gem` osztály segítségével a feladatban szereplő drágaköveket reprezentáljuk.

- A `Gem` osztálynak három privát adattagja van, a drágakő nevének jelölésére szolgáló `String` típusú `name`, a színének leírására szolgáló `Colour` felsorolási típusú `colour`és az értékének dukátban való kifejezésére szolgáló egész típusú `value` változó. Az osztálynak meg kell valósítania a `Comparable<Gem>` interfészt, hogy a drágakövek rendezhetőek legyenek. A `Gem` osztály konstruktora publikus, amely három kapott paraméter alapján beállítja a `name`, a `colour` és a `value` adattagokat a megadott értékekre.
- Az osztálynak három lekérdező metódusa van, a `getName`, a `getColour` és a `getValue`, amelyek segítségével a drágakő nevét, színét és dukátban kifejezett értékét tudjuk lekérdezni.
- A `Gem` osztályban felüldefiniáljuk az `Object`-től örökölt `toString` metódust. A felüldefiniált `toString` metódus az alábbi szöveget adja vissza: "The [colour] [name] is worth [value] ducats.", ahol [colour] a drágakő színe, [name] a neve, [value] pedig az értéke. Pl. "The blue sapphire is worth 15 ducats." *(1 pont)*
- Az `Object`-től örökölt `equals` metódust is felüldefiniáljuk. Két `Gem` típusú objektum akkor egyenlő, ha a `name`, `colour` és `value` értékeik rendre megegyeznek. *(2 pont)*
- Az `Object`-től örökölt `hashCode` metódust is felüldefiniáljuk. Megköveteljük, hogy azonos drágakövekre a `hashCode` azonos értéket adjon vissza. *(2 pont)*
- A `Gem` osztálynak meg kell valósítania a `Comparable<Gem>` interfészt, hogy a drágakövek rendezhetőek legyenek. Elsődlegesen a drágakő neve, másodlagosan a színe, harmadlagosan pedig az értéke szerint rendezünk. Pl. ("diamond", COLOURLESS, 50) = ("diamond", COLOURLESS, 50); ("sapphire", BLUE, 20) > ("ruby", RED, 10); ("sapphire", BLUE, 20) > ("sapphire", BLUE, 15); ("ruby", RED, 10) < ("ruby", PINK, 20). *(2 pont)*

Position *(8 pont)*

A `game.utils.Position` osztály segítségével a feladatban szereplő helyek pozícióját tudjuk ábrázolni.

- A `Position` osztálynak két privát, egész értékű adattagja van, amelyek a feladatban szereplő helyek pozícióját tárolják el. Arendelle királysága, ahol Anna bolyong, egy téglalap alapú tábla. A hely pozícióját annak alapján határozzuk meg, hogy a hely a téglalap hányadik sorának (`y`) hányadik oszlopában (`x`) található. Feltesszük, hogy mindegyik hely egységnyi pozíción helyezkedik el.
- A `Position` osztálynak hat nyilvános, osztályszintű (static), konstans (final) adattagja is van. Az egész értékűek a következők: `MIN_X` az `x` koordináta minimumát (értéke `1`),`MIN_Y` az `y` koordináta minimumát (értéke `1`), `MAX_X` az `x` koordináta maximumát (értéke `8`), `MAX_Y` pedig az `y` koordináta maximumát (értéke `8`) adja meg. A `Position`típusú adattagok az alábbiak: `INITIAL_POSITION` a kezdeti pozíciót, vagyis az `(1,1)` pozíciót (a kezdeti pozíció a téglalap alapú tábla bal alsó sarkában van; a kezdeti pozíciótól közvetlenül jobbra lévő pozíció, ugyanabban a sorban az `(1,2)`, a kezdeti pozíció felette lévő pozíció, ugyanabban az oszlopban a `(2,1)`), `WINNING_POSITION`pedig a célpozíciót rögzíti, amelyet véletlenszerűen inicializálunk.
- A `Position` osztálynak két konstruktora van. Az első konstruktor publikus, amely két kapott paraméter alapján beállítja az `x` és az `y` adattagokat a megadott értékekre. A második konstruktor privát, paraméter nélküli, és véletlenszerűen inicializálja az `x` és az `y` adattagokat a minimum- (`MIN_X`,`MIN_Y`) és maximumértékek (`MAX_X`,`MAX_Y`) közötti

egész számra. A véletlenszerűen inicializált pozíció `x` és `y` koordinátája nem egyezhet meg a kezdeti pozíció `x` és `y` koordinátájával (vagyis egyik sem lehet 1). *(1 pont)*
- A `Position` osztálynak van egy `getX` és egy `getY` nevű getter metódusa, amelyek az adott pozíció `x` és `y` koordinátáit adják vissza. Van továbbá egy `setX` és egy `setY`nevű setter metódusa is, amelyek a paraméterként kapott egész érték alapján beállítják az adott pozíció `x` és `y` koordinátáit.
- A `Position` osztálynak van egy paraméter nélküli `isInside` nevű metódusa, amely eldönti, hogy a pozíció az Arendelle királyságát reprezentáló téglalapon belül van-e. *(1 pont)*
- Legyen egy `neighbours` függvénye, amely visszaadja a szomszédos pozíciók listáját. Az átlós pontok nem számítanak szomszédosnak. Tehát a `(2,2)` pozíció szomszédai:`(1,2), (2,1), (2,3), (3,2)`. Ügyeljünk arra, hogy a listában csak azok a pozíciók szerepeljenek, amelyek az Arendelle királyságát reprezentáló téglalapon belül vannak! *(2 pont)*
- A `Position` osztályban felüldefiniáljuk az `Object`-től örökölt `toString` metódust. Az eredmény legyen `(x,y)` formátumú. Pl. `(1,1)`. *(1 pont)*
- Az `Object`-től örökölt `equals` metódust is felüldefiniáljuk. Két `Position` típusú objektum akkor egyenlő, ha megfelelő koordinátáik rendre megegyeznek. *(1 pont)*
- Az `Object`-től örökölt `hashCode` metódust is felüldefiniáljuk. Megköveteljük, hogy azonos pozíciókra a `hashCode` azonos értéket adjon vissza. *(1 pont)*
- Az osztálynak meg kell valósítania a `Comparable<Position>` interfészt, hogy a pozíciók rendezhetőek legyenek. Először a pozíciók `x` koordinátáját hasonlítjuk össze. Ha ezek megegyeznek, akkor az `y` koordinátáját. Pl. `(1,2)` nagyobb, mint `(1,1)`, viszont kisebb, mint `(2,1)`. *(1 pont)*

Kingdom *(1 pont)*

Valósítsuk meg a `game.arendelle.Kingdom` absztrakt osztályt, amely Arendelle királyságában szereplő helyszíneknek a közös ősosztálya.

- A `Kingdom` osztályban felüldefiniáljuk az `Object`-től örökölt `toString` metódust. A felüldefiniált `toString` metódus az alábbi szöveget adja vissza: "Welcome...".

SnowStorm *(1 pont)*

Valósítsuk meg a `game.arendelle.SnowStorm` osztályt, amely a `Kingdom` osztály leszármazottja, és olyan helyszínt reprezentál, ahol hóvihar van kilátásban.

- A `SnowStorm` osztályban definiáljuk felül a `Kingdom` osztálytól örökölt `toString` metódust. A felüldefiniált `toString` metódus az alábbi szöveget adja vissza: "A winter storm will move across the region. You had better avoid it.".

TrollCave *(1 pont)*

Valósítsuk meg a `game.arendelle.TrollCave` osztályt, amely a `Kingdom` osztály leszármazottja, és olyan helyszínt reprezentál, ahol trollbarlang van.

- Az `TrollCave` osztályban definiáljuk felül a `Kingdom` osztálytól örökölt `toString` metódust. A felüldefiniált `toString` metódus az alábbi szöveget adja vissza: "Trolls are giant wealth suckers. Steer clear of them.".

Chalet *(2 pont)*

Valósítsuk meg a `game.arendelle.Chalet` osztályt, amely a `Kingdom` osztály leszármazottja, és olyan helyszínt reprezentál, ahol hegyi kunyhó van.

- A `Chalet` osztálynak egy privát adattagja van, a kunyhóban lévő drágakövek jelölésére szolgáló ArrayList típusú `gemList` változó. Az osztály konstruktora publikus, amely a kapott paraméter alapján beállítja a `gemList` változót a megadott értékre.
- Az osztálynak van egy `getGemList` nevű lekérdező metódusa, amelynek segítségével a kunyhóban lévő drágaköveket tudjuk lekérdezni. Van továbbá egy `setGemList` nevű setter metódusa is, amely a paraméterként kapott drágakőlista alapján beállítja a kunyhóban lévő drágakövek listáját.
- A `Chalet` osztálynak van egy nyilvános `collectItems` nevű metódusa, amely paraméterként egy egész típusú változót kap, amely megadja, hogy mennyivel kell csökkenteni a kunyhóban lévő drágakövek számát. A `collectItems` az összegyűjtött drágaköveket egy `ArrayList<Gem>` listában adja vissza. A drágakövek számát úgy csökkentjük, hogy a drágakőlista elejéről töröljük a megadott számú drágakövet, amennyiben ez lehetséges (ne akarjunk több drágakövet összegyűjteni, mint amennyi van a kunyhóban - ha többet akarnánk, akkor ne történjen semmi). *(1 pont)*
- A `Chalet` osztályban definiáljuk felül a `Kingdom` osztálytól örökölt `toString` metódust. A felüldefiniált `toString` metódus az alábbi szöveget adja vissza: "Welcome...If you are lucky, you will come across some gems here for Elsa.". *(1 pont)*

SnowPalace *(1 pont)*

Valósítsuk meg a `game.arendelle.SnowPalace` osztályt, amely a `Kingdom` osztály leszármazottja, és a jégpalotát reprezentálja.

- A `SnowPalace` osztályban definiáljuk felül a `Kingdom` osztálytól örökölt `toString` metódust. A felüldefiniált `toString` metódus az alábbi szöveget adja vissza: "Welcome...to the Palace. Elsa is waiting for you. You have won the game.".

Reachable *(2 pont)*

Hozzuk létre a `game.arendelle.Reachable` interfészt, amely azt jelzi, hogy egy adott helyszín elérhető-e.

- Az interfésznek legyen egy paraméter nélküli, logikai visszatérési értékű `isReachable` nevű metódusa, amely azt adja vissza, hogy az adott helyszín elérhető-e.
- A `Kingdom` osztály valósítsa meg a `Reachable` interfészt! *(1 pont)*
- A `SnowStorm` osztály `isReachable` metódusa véletlenszerűen ad vissza igaz vagy hamis logikai értéket, attól függően, hogy elérhető-e a helyszín vagy sem a hóvihar közeledte miatt. *(1 pont)*
- A `TrollCave` osztály `isReachable` metódusa igaz logikai értékkel tér vissza, jelezve, hogy Annának egy trollal kell megküzdenie (bár veszélyes a helyszín, de elérhető).
- A `Chalet` osztály `isReachable` metódusa igaz logikai értékkel tér vissza, jelezve, hogy Anna hamarosan megpihenhet egy hegyi kunyhóban (a kunyhó elérhető).

- A `SnowPalace` osztály `isReachable` metódusa igaz logikai értékkel tér vissza, jelezve, hogy Anna elérte a jégpalotát.

Player *(2 pont)*

Valósítsuk meg a `game.players.Player` absztrakt osztályt, amely a játékosoknak a közös ősosztálya.

- A `Player` osztálynak két védett (protected) adattagja van, a játékos birtokában lévő drágakövek jelölésére szolgáló ArrayList típusú `gemList`, valamint a játékos pozíciójának jelölésére szolgáló `Position` típusú `pos` változó. A `Player` osztály konstruktora publikus, amely két kapott paraméter alapján beállítja a `gemList` és a `pos` változókat a megadott értékekre.
- Az osztálynak két lekérdező metódusa van, a `getGemList` és a `getPosition`, amelyek segítségével a játékos birtokában lévő drágaköveket és a játékos pozícióját tudjuk lekérdezni. Van továbbá egy `setGemList` és egy `setPosition` nevű setter metódusa, amelyek a paraméterként kapott drágakőlistája és pozíció alapján beállítják a játékos birtokában lévő drágakövek listáját és a játékos pozícióját.
- Az osztálynak van egy védett, absztrakt, paraméter nélküli és String visszatérési értékű `display` metódusa is.

Troll *(2 pont)*

Valósítsuk meg a `game.players.Troll` osztályt, amely a `Player` osztály leszármazottja, és a trollokat reprezentálja.

- A `Troll` osztálynak van egy privát, konstans adattagja, a troll nevének jelölésére szolgáló String típusú `name`.
- A `Troll` osztály konstruktora publikus, amelynek három paramétere van: az ArrayList típusú `gemList`, a Position típusú `pos` és az egész típusú `num`.
  A `gemList`segítségével a troll drágakőlistáját, a `pos` segítségével a pozícióját tudjuk beállítani, a `num` pedig a troll számát jelöli. A troll nevének (`name`) beállítása úgy történik, hogy a "Troll" szóhoz hozzáfűzzük a `num` értékét. Tehát ha a `num` értéke 3, akkor a troll neve "Troll 3" lesz. *(1 pont)*
- A `Troll` osztálynak egy lekérdező metódusa is van, a `getName`, amely a troll nevét adja vissza.
- A `Troll` osztályban definiáljuk felül a `Player` osztálytól örökölt `display` metódust. A láthatóság legyen nyilvános. A felüldefiniált `display` metódus a következő szöveget adja vissza: "Hi. My name is [name]. Nice to meet you, Anna.", ahol [name] a troll nevét jelöli. Pl. "Hi. My name is Troll 3. Nice to meet you, Anna." *(1 pont)*

Anna *(7 pont)*

Valósítsuk meg a `game.players.Anna` osztályt, amely a `Player` osztály leszármazottja, és Annát reprezentálja.

- Az `Anna` osztálynak van egy privát, konstans String típusú `name` nevű adattagja, amely Anna nevét reprezentálja, értéke "Princess Anna".

- Az `Anna` osztály konstruktora publikus, amelynek két paramétere van: az `ArrayList<Gem>` típusú `gemList` és a `Position` típusú `pos`. A `gemList` segítségével Anna drágakőlistáját, a `pos` segítségével pedig a pozícióját tudjuk beállítani.
- Az osztálynak van egy nyilvános `getName` nevű metódusa, mellyel Anna nevét kérdezhetjük le.
- Az osztálynak van egy nyilvános `fight` nevű, visszatérési érték nélküli metódusa (Anna harcol a trollal), amely paraméterként kap egy Troll típusú változót. A metódus csökkenti Anna drágaköveinek számát, ha vannak drágakövei (a troll drágaköveinek száma ugyanennyivel nő). Az, hogy mennyi drágakövet veszünk el Annától véletlenszerű (de ne akarjunk tőle többet elvenni, mint amennyi van neki) és az is, hogy melyik drágaköveket vesszük el tőle. *(2 pont)*
- Az osztálynak van egy nyilvános `fight` nevű, visszatérési érték és paraméter nélküli metódusa (Anna harcol a téli viharral), amelyben Anna drágaköveit a természetes rendezés szerint sorba rendezzük, és drágakőlistájának első elemét vesszük el - ezt veszíti el Anna a viharban. *(2 pont)*
- Az osztálynak van egy nyilvános `move` nevű, visszatérési érték és paraméter nélküli metódusa, amelynek segítségével Annát léptetjük. Véletlenszerűen választunk egy pozíciót azon pozíció szomszédai közül, amelyen Anna áll. *(2 pont)*
- Az `Anna` osztályban definiáljuk felül a `Player` osztálytól örökölt `display` metódust. A láthatóság legyen nyilvános. A felüldefiniált `display` metódus a következő szöveget adja vissza: "I have won the game.", ha Anna pozíciója nyerő pozíció, egyébként pedig az "I have lost the game." szöveget. *(1 pont)*

AdventureGame *(8 pont)*

Végül írjuk meg az `AdventureGame` osztályt, amely magát a játékot valósítja meg.

- Az `AdventureGame` osztálynak négy nyilvános, egész típusú, konstans osztályszintű adattagja van, a `NUM_OF_TROLL_CAVES`, a `NUM_OF_SNOW_STORMS`, a `NUM_OF_CHALETS` és a`MAX_NUM_OF_STEPS`, amelyek trollbarlangok számát, azon helyek számát, ahol hóvihar van, illetve ahol kunyhó található, valamint a lépések számának maximumát jelölik. A`NUM_OF_TROLL_CAVES` és `NUM_OF_SNOW_STORMS` adattagok kezdeti értéke egy-egy 15 és 20 közötti véletlen szám A `NUM_OF_CHALETS` adattagot úgy számolhatjuk, hogy a játékban szereplő összes helyek számából levonjuk a trollbarlangok, a hóviharos helyek és a hópalotát tartalmazó helyek számát; a játékban 1 hópalota van, a nyerőpozíción. Végül 50 a `MAX_NUM_OF_STEPS` adattag értéke.
- Az osztálynak négy privát, osztályszintű adattagja van, az ArrayList típusú `trollList`, az ArrayList típusú `chalets`, a SnowPalace típusú `sp` és az ArrayList típusú `pool`, amelyek rendre a trollok listáját, a kunyhók listáját, a hópalotát és a játékban szereplő különböző fajta drágaköveket jelölik. Feltesszük, hogy a kunyhót tartalmazó mezőkön egy kunyhó van, valamint azt, hogy a trollbarlangokban is csak egy troll lakik.
- Az osztálynak van egy privát `Map` típusú adattagja is, a `board`, amely a játéktáblát reprezentálja, és a `Position` típusú objektumokhoz `Kingdom` típusú objektumokat (a királyságban szereplő helyszíneket) rendel (egy helyszínhez egy pozíciót rendelünk hozzá).
- Az osztálynak van egy nyilvános, osztályszintű, `Anna` típusú, `princess` adattagja is, amely Anna hercegnőt reprezentálja.
- Az osztálynak van egy nyilvános, osztályszintű, egész típusú `numOfSteps` adattagja is, a hercegnő által megtett lépések számát adja meg.

- Az `AdventureGame` osztály konstruktora publikus. A konstruktorban inicializáljuk a 8*8-as (Position.MAX_X* Position.MAX_Y) játéktáblát és felépítjük a királyságot (l. a`buildKingdom` metódust alább). A királyság felépítése előtt ellenőrizzük, hogy a kunyhót, trollbarlangot, viharos helyet és hópalotát tartalmazó helyek számának összege megegyezik-e a játéktábla mezőinek számával. Ha ez nem teljesül, akkor dobjunk `InappropriateBoardSizeException` kivételt (l. alább). *(2 pont)*.
- Az `AdventureGame` osztálynak négy lekérdező metódusa is van: a `getKingdom`, a `getPool`, a `getChalets` és a `getTrolls`, amelyek rendre a játéktáblát, valamint a játékban szereplő különböző fajta drágakövek listáját, a kunyhók listáját és a trollok listáját adják vissza.
- A játékban szereplő drágakőfajtákat a privát `initGemPool` metódus inicializálja, amely `ArrayList<Gem>` típusú listát ad vissza. A játékban a következő drágakőfajták szerepelnek: ("diamond", COLOURLESS, 50), ("diamond", YELLOW, 45), ("diamond", BROWN, 40), ("diamond", GREY, 35), ("sapphire", BLUE, 30), ("ruby", RED, 10), ("ruby", PINK, 20), ("emerald", GREEN, 40), ("amethyst", PURPLE, 30).
- Az osztálynak van egy privát `initChalets` nevű metódusa, amely a játékban szereplő kunyhókat inicializálja és egy `ArrayList<Chalet>` típusú listát ad vissza. A kunyhókban található drágaköveket a játékban szereplő drágakőfajták közül választjuk ki. A drágakövek száma egy kunyhóban egy 1 és 9 közötti véletlen szám. Két tökéletesen ugyanolyan (azaz ugyanolyan fajtájú, színű és értékű) drágakő nem lehet egy kunyhóban. Pl. két kunyhó esetén a következő egy helyes inicializálás: 1. kunyhó drágakövei: ("diamond", COLOURLESS, 50), ("diamond", YELLOW, 45),("diamond", BROWN, 40),("diamond", GREY, 35), 2. kunyhó drágakövei: ("ruby", PINK, 20), ("diamond", COLOURLESS, 50), ("emerald", GREEN, 40), ("amethyst", PURPLE, 30),("diamond", YELLOW, 45). Ha az 1. kunyhó drágakövei a következők lennének: ("diamond", COLOURLESS, 50), ("diamond", COLOURLESS, 50), ("diamond", YELLOW, 45),("diamond", BROWN, 40),("diamond", GREY, 35), akkor az inicializálás nem helyes, mert a ("diamond", COLOURLESS, 50) kétszer fordul elő ugyanabban a kunyhóban.
- Az `AdventureGame` osztálynak van egy privát, `buildKingdom` nevű, visszatérési érték és paraméter nélküli metódusa, amely inicializálja a játékban szereplő drágakőfajtákat (l. az `initGemPool` metódust), a kunyhók listáját (l. az `initChalets` metódust), a játéktábla pozícióihoz véletlenszerűen rendeli hozzá a helyeket (a játéktábla kezdeti pozícióján kunyhót tartalmazó hely áll, a nyerő pozíción a jégpalota, a többi helyet véletlenszerűen inicializáljuk, a véletlenszerű hozzárendeléshez használhatunk pl. egy listát, amelyben a pozíciókat véletlenszerűen tudjuk keverni) és inicializálja a játékban szereplő trollokat is (l. az `initTrolls` metódust alább). *(3 pont)*
- Az `AdventureGame` osztálynak van egy privát, `initTrolls` nevű, visszatérési érték és paraméter nélküli metódusa, amely a trollok listáját inicializálja. A trollokat a játéktábla trollbarlangot tartalmazó helyeihez rendeljük hozzá, minden barlanghoz egyet. A trollok drágakő listája kezdetben üres, pozíciója megegyezik a barlangját tartalmazó hely pozíciójával, neve pedig tartalmazza a troll sorszámát. A trollokat 1-től, nem pedig 0-tól kezdjük el számozni.
- Az `AdventureGame` osztálynak van egy nyilvános `simulation` nevű, visszatérési érték és paraméter nélküli metódusa, amellyel a kalandjátékot szimuláljuk. Ebben először Anna hercegnőt inicializáljuk: drágakőlistáját üresre állítjuk, pozíciója pedig a kezdeti pozíció lesz. A szimuláció addig tart, amíg Anna el nem éri a nyerőpozíciót, és lépéseinek száma nem haladja meg a maximálisan megengedett lépések számát. Ha a hercegnő kunyhót tartalmazó helyen van, akkor a kunyhóban található drágakövek közül valamennyit zsákmányol (a zsákmányolt drágakövek számát véletlenszerűen határozzuk meg), majd továbblép. Ha trollbarlangot tartalmazó helyre lép, akkor megküzd a trollal, majd továbblép. Mindig írassuk ki, hogy melyik trollal harcol Anna (l. a `display` metódust). Se a kunyhót, se a trollbarlangot tartalmazó hely esetében ne felejtsük el ellenőrizni, hogy a

hely elérhető-e! Ha a helyen, amelyre lépni szeretne, épp hóvihar tombol, akkor ellenőrizze azt, hogy a hely elérhető-e. Ha elérhető, akkor megküzd a hóviharral, majd továbblép. Ha nem érhető el, akkor csak továbblép. Ha a hópalotát tartalmazó helyre ér, de az eddig megtett lépéseinek száma nem haladta meg a megengedett lépésszámot, akkor az azt jelenti, hogy megnyerte a játékot. Ha a szimuláció véget ér, írassuk ki, hogy Anna megnyerte-e a játékot, vagy sem (l. a `display`metódust) *(3 pont)*

InappropriateBoardSizeException *(1 pont)*

Hozzunk létre egy saját kivételosztályt, amellyel azt fogjuk jelölni, hogy a tábla dimenzióit nem megfelelően adtuk meg. Ez származzon az `IllegalArgumentException`kivételosztályból. Legyen egy paramétert nem váró konstruktora és hívja meg az ősének üzenetszöveget váró konstruktorát. A megadott üzenet legyen "Inappropriate board size dimensions.".

Pontozás

A tesztelő által adott pontszám csak becslésnek tekinthető, a gyakorlatvezető levonhat pontokat, vagy adhat részpontokat.

0 - 13: *elégtelen (1)* 14 - 21: *elégséges (2)* 22 - 29: *közepes (3)* 30 - 37: *jó (4)* 38 - 45: *jeles (5)*

Jó munkát! :-)