Programozás I.

Kötelező program

Don't starve: Battle Royale

Általános követelmények, tudnivalók

- A feladat beadási határideje: 2023. április 30. 23:59:59. Ez szigorú határidő, a Bíró előre megadott időben zár. Pótlásra, javításra a határidő lejárta után nincs lehetőség.
- Az elkészült programot 2023 alkalommal lehet benyújtani, a megadott határidőig.
- A feltöltések közül a legmagasabb pontszámú számít a feladat értékelésekor.
- A nem forduló kódok nem kerülnek kiértékelésre. Az előre megírt osztályok, adattagok és metódusok szükségesek a teszteléshez, így azokat kitörölni nem szabad, különben a tesztelés egy része vagy egésze nem elvégezhető.
- Amennyiben egy feltöltött osztályban fordítási hiba van, akkor az az osztály egyáltalán nem tesztelhető.
- Ha végtelen ciklus van a programban, akkor a kiértékelést a Bíró 180 másodperc után megszakítja és az eredmény 0 pont lesz.
- A kész forráskódot tömörítve, zip formátumban kell feltölteni. Ügyelj rá, hogy a feltöltött zipnek a teljes package hierarchiát tükröznie kell. A zipet kitömörítve közvetlenül a default package-ben lévő package-et kell látnunk (prog1). A legtöbb fejlesztői környezet esetén ez azt jelenti, hogy az src mappa tartalmát kell betömörítenünk.
  - Zip készítése: Windowson és Linuxon is lehet a GUI-ban jobb klikkes módszerrel tömörített állományt létrehozni (Windowsban pl. a 7-Zip nevű ingyenes program használatával).
  - Linux terminálon belül például a "zip feladat.zip *.java" paranccsal is elkészíthető a megfelelő állomány.
- A bíró a programot Java 17-es verzióval fordítja és futtatja.
- Az értékelés a megvalósítandó interface-ek metódusain, illetve az egyéb, előre elkészített, de nem implementált függvényeken keresztül történik.
- A kiértékelés során a program egyes részei függnek egymástól, egyes részek a többi nélkül nem megvalósíthatók, nem tesztelhetők, így erre érdemes odafigyelnünk a megoldások elkészítésekor.
- Ebből adódóan, ha a program egy fontosabb részében van hiba, az a program más részeire is hatással lehet, tehát érdemes azt kijavítani először.

- A bíró jelzi, hogy az egyes funkcióknak mely funkciók az előfeltételei. Ha az ikon pirossal jelenik meg, akkor az előfeltételt nem sikerült tökéletesen teljesíteni. Ez nem jelenti azt, hogy ez a részfeladat is 0 pontot ér, csak annyit jelez, hogy lehet, hogy az előfeltétel nem teljesülése miatt nem kapunk rá pontot.
- A kiértékelés részletesen megtekinthető a Bíróban feltöltés után, ahol a rendszer kiírja, hogy mely funkcionalitások értek pontot és melyek nem. Ezen segítség alapján a hibák könnyen javíthatók.
- Ahol a feladat külön nem kéri, referenciákat használjunk és ne hozzunk létre feleslegesen új objektumokat.
- A program elkészítése során semmilyen külső függvénykönyvtár nem használható.
- A Bírós kiértékelő szándékos "kijátszása" esetén (az adott funkciók megvalósítása nélkül egy hibát kihasználva pontszerzés) a kapott pontszám a gyakorlatvezetők által később törölhető.

Kódminőség ellenőrzése

A Bíró a kód minőségét is ellenőrzi, amely rengeteg metrikát magában foglal. Ezeket részletesen a Bíróban lévő riportban megtekinthetjük, illetve mindegyikről rövid leírást olvashatunk, hogy pontosan mit mér és hogyan lehet javítani az adott pontszámon. Számos olyan szempont van, aminél több, mint 100%-ot lehet elérni. A plusz pontszám beleszámít a kódminőségre kapott teljes pontba, így az akár 100%-nál magasabb is lehet. A maximum pontszám szempontonként eltér (de minimum 100, maximum 125 százalék). A kód minősége az alábbi csoportok mentén történik:

- Objektumorientáltság: a programunk mennyire jól valósítja meg az objektumorientáltság elveit, a megoldás mennyire felel meg az objektumorientáltság szemléletmódjának.
- Kódbonyolultság: a programunk mennyire egyszerű és könnyen átlátható. Alapelvként annyi elmondható, hogyha egy osztály vagy metódus túl hosszú, akkor valószínűleg nem jó úton járunk.
- Duplikátumok: a programunk mennyi duplikátumot (klónt, azaz Ctrl+C, Ctrl+V-vel másolt és beillesztett) tartalmaz. A kódismétléseket érdemes elkerülni.
- Warningok: a programunk mennyi warningot tartalmaz
- Antipatternek: a programunk mennyi olyan részt tartalmaz, amely egyenesen ellentmondásba ütközik az objektumorientáltság 1-1 fontos alapelvével
- Dokumentáció: a programunk mennyire (jól) van dokumentálva
- Elnevezések: a programunkban a változók, adattagok, metódusok, osztályok a tanult, Java-ban használatos konvenciók szerint vannak-e elnevezve
- Kódolási stílus: a programunk külalakját ellenőrzi, hogy mennyire van szépen tagolva, a különböző operátorok, zárójelek, stb. mennyire vannak megfelelően használva

- Programkód szervezése: a programunkban vannak-e olyan elemek, amelyek bár a működésre nincsenek hatással, de a program átláthatóságát, olvashatóságát valamilyen szinten rontják.

Értékelés

A Bíróban az értékelésnek 3+1 szakasza van.

- Kötelező elemek: ezek az alapvető funkcionalitásokat tesztelik, amelyek nélkül a program többi része nem tesztelhető. A többi teszt futtatásához ennek a résznek tökéletesen kell lefutnia. Ezek a tesztek nem érnek pontot. A minta.zip-ben lévő megoldáskezdemény már teljesíti az összes feltételt, csak arra kell figyelni, hogy el ne rontsuk azokat.

- Funkcionalitás: a kért funkciókat ellenőrzi, hogy azok meg vannak-e valósítva és megfelelően működnek-e.

- Kódminőség: a kód minőségét teszteli a fent leírt, illetve a bíróban látható szempontok mentén.

- Statisztika: a programunkkal kapcsolatos néhány statisztika. A kiértékelés során semmilyen jelentősége nincs, de érdekességképpen megtekinthető.

A feladat végső pontszáma funkcionalitásra és kódminőségre kapott pontszámok összege. A kódminőségre kapott pontszám korrigálásra kerül a funkcionalitásra kapott pontszámmal. Például:

- Ha a funkcionalitás 100%-os, akkor a kódminőségre annyi pont jár, amennyit a bíró kiszámított.

- Ha a funkcionalitás 50%-os (félig van kész a feladat), akkor a kódminőségre kapott pontszám megszorzásra kerül 0.5-tel

- Ha a funkcionalitás 10%-os, akkor a kódminőségre kapott pontszám megszorzásra kerül 0.1-gyel

A feladatra kapott pontszám a riport.html fájlban látható. A bíró felületén az összesítésben (a bíró limitációi miatt) a kapható pontszám 100x-osa jelenik meg, erre érdemes odafigyelni. A bíró kijelzi, hogy a feladatra mennyi a minimum elérendő pont (ahol ebből adódóan szintén egy 100x-os érték látható).

Program elkészítése

A program elkészítéséhez egy elkezdett keretrendszert kell kibővíteni és a hiányzó részeit megvalósítani. A keretrendszer a Bíró-ról letölthető minta.zip néven.

Kicsomagolás után ezt beimportálhatjuk egy általunk kedvelt fejlesztői környezetbe (opcionális). A keretrendszer dokumentálva van: minden package, osztály, adattag és metódus tartalmaz JavaDoc kommenteket, amelyeket a forráskódon belül is megtekinthetünk, azonban célszerű legenerálni belőle a HTML dokumentumot (a kedvenc fejlesztői környezetünkben, vagy parancssorból a javadoc paranccsal) a jobb átláthatóság kedvéért.

Ebben láthatjuk minden osztálynak a leírását, illetve a megvalósítandó függvényeknél minden azzal kapcsolatos tudnivalót. Az esetleges "hiányzó" információkat ezen pdf tartalmazza.

A kiadott osztályok alapján UML diagramot készíthetünk, amely segít megérteni a projekt szerkezetét. Ehhez ha IntelliJ IDEA-ban dolgozunk, kattintsunk jobb klikkel az src mappára, majd ott diagrams -> show diagram. Ha itt egyes helyeken csak package-eket látunk, akkor jobb klikk, majd expand nodes, és akkor az összes package-et is ki fogja fejteni. Ezt addig ismételjük, amíg nem látjuk az összes osztályt. A diagramon alapvetően nem jelennek meg, csak az osztályok. Ha szeretnénk látni az adattagokat, metódusokat és kapcsolatokat, akkor ezt a bal felső sarokban tudjuk szabályozni (fields, constructors, methods és show dependencies) a kis ikonok segítségével.

Ha szeretnénk csak egy adott package-ben lévő osztályokat látni, azt is megtehetjük, ezesetben az adott package-re kell jobb klikkelni az src helyett.

Másik fejlesztői környezetben is nagy valószínűséggel lehet UML diagramot generálni, ennek pontos menetéről az interneten érdemes utánanézni.

Megvalósítással kapcsolatos információk

- Új package-ek szabadon létrehozhatóak, de minden package a meglévő solution package-en belül kell, hogy legyen
- Új osztályok szabadon létrehozhatóak, de minden osztály a meglévő solution package-en belül kell, hogy legyen
- A meglévő osztályokban új adattagok, új metódusok szabadon létrehozhatóak
- A meglévő osztályokban már meglévő adattagokat és metódusok törzsét nem célszerű módosítani, mert az a tesztelés során problémákhoz vezethet és pontveszteséggel járhat.
- A meglévő osztályokban már meglévő metódusok, illetve az interface-ek metódusainak fejlécét megváltoztatni tilos, mert az a tesztelés egy pontján garantáltan fordítási hibához vezet és fordítás hiányában a programot nem lehet értékelni.

Kiegészítendő osztályok

A keretrendszer tartalmaz számos olyan osztályt, amelyben vannak nem implementált metódusok. Ezeket úgy tudjuk felismerni, hogy a törzse egy utasítást tartalmaz:
throw new NotImplementedException();
Az ilyen megvalósítandó metódusok:

- prog1.kotprog.dontstarve.solution.GameManager.joinCharacter
- prog1.kotprog.dontstarve.solution.GameManager.getCharacter
- prog1.kotprog.dontstarve.solution.GameManager.remainingCharacters
- prog1.kotprog.dontstarve.solution.GameManager.loadLevel
- prog1.kotprog.dontstarve.solution.GameManager.getField
- prog1.kotprog.dontstarve.solution.GameManager.startGame
- prog1.kotprog.dontstarve.solution.GameManager.tick
- prog1.kotprog.dontstarve.solution.GameManager.time
- prog1.kotprog.dontstarve.solution.GameManager.getWinner
- prog1.kotprog.dontstarve.solution.GameManager.isGameStarted
- prog1.kotprog.dontstarve.solution.GameManager.isGameEnded
- prog1.kotprog.dontstarve.solution.GameManager.setTutorial
- prog1.kotprog.dontstarve.solution.inventory.items.EquippableItem.percentage
- prog1.kotprog.dontstarve.solution.utility.Position.getNearestWholePosition

Megvalósítandó interface-ek

A keretrendszer tartalmaz számos interface-t, amelyet jelenleg egy osztály sem implementál. A feladat ezen interface-ekhez olyan osztályt készíteni, amelyek azokat implementálják, és természetesen az összes metódusnak értelmes implementációt adni.

Amennyiben egy interface-t csak egy osztály implementálja, akkor a Bíró automatikusan megtalálja azt. Amennyiben több implementáció is tartozik hozzá, akkor használjuk a @Testable annotációt azon az osztályon, amelyiket a bíróval teszteltetni szeretnénk. Ha nem használjuk (vagy több osztályra is rátesszük), akkor a tesztelésre kerülő osztály véletlenszerűen kerül kiválasztásra.

Az ilyen interface-ek:

- prog1.kotprog.dontstarve.solution.character.BaseCharacter
- prog1.kotprog.dontstarve.solution.inventory.BaseInventory
- prog1.kotprog.dontstarve.solution.level.BaseField

Kötelező elemek

Az alábbi szempontok kötelezőek, ezek nélkül a program nem tesztelhető.

- A prog1.kotprog.dontstarve.solution.inventory.BaseInventory interface-t implementáló osztály(ok)-nak kötelezően lennie kell default konstruktorának.
- A prog1.kotprog.dontstarve.solution.GameManager osztály ne tartalmazzon (a meglévőkön túl) statikus változókat és metódusokat és ne támaszkodjon más osztályok statikus adattagjaira és metódusaira.

Feladatleírás

A játékról

A feladat megvalósítani Java nyelven a Don't Starve egy olyan változatát, amely egyben egy Battle Royale is. A játék lényege, hogy mi legyünk az utolsó életben maradó játékos a játékban. A játékban 1 emberi játékos és tetszőlegesen sok gépi játékos vesz részt.

A pálya

A pálya kialakítása

A játék egy kellően nagy területen játszódik (nem feltétlen négyzet alakú). A pálya alapvetően szárazföldből áll, de lehetnek rajta tavak is. A karakterek a tóra nem tudnak rálépni (az összes többi mezőre igen), és a tavon nem lehet semmilyen tárgy. A tó nem választja szét a szárazföldet több különálló részre.

A pályán található tárgyak

A pályán 5 féle "tereptárgy" található.

Név	Szükséges eszköz	Idő	Nyersanyag
Fa	fejsze	4 időegység	2 farönk
Kő	csákány	5 időegység	3 kő
Gally	-	2 időegység	1 gally
Bogyó	-	1 időegység	1 bogyó
Répa	-	1 időegység	1 répa

Az első oszlop a tereptárgy neve. A második oszlop jelzi, hogy milyen eszköz szükséges a nyersanyag "begyűjtéséhez" (az eszköznek a karakter kezében kell lennie). A harmadik oszlop a begyűjtési időt, míg a negyedik a kapott nyersanyagot és annak mennyiségét jelzi.

Tereptárgyak begyűjtése

- A fa kivágásához fejszére, míg a kő kibányászásához csákányra van szükség. A fejszét, illetve a csákányt a a karakternek a kezében kell tartania.
- A több időegység alatt kitermelhető tereptárgyak kitermelése tovább tart. A kitermelés elvégezhető a megadott számú egymás utáni körben, de olyan is lehet, hogy megkezdjük és csak jóval később folytatjuk.
- A fa és kő esetében a kitermelés végeztével a tereptárgy eltűnik a pályáról, és helyette a mezőre megadott mennyiségben farönk vagy kő kerül, amelyeket a játékos 1 időegység alatt (tehát egyszerre) begyűjthet.
- A gally, bogyó és répa esetében a kitermelt nyersanyag azonnal a játékos inventory-jába kerül. Ha ott nincs hely, akkor ugyanúgy az adott mezőre kerül, ahonnan később fel lehet venni.

A játék menete

Pálya beolvasása

A játék kezdetekor (még a játékosok csatlakozása előtt) a pályát be kell töltenünk. A pálya elkészítése egy kép alapján történik, ahol a kép pixelei jelölik az egyes mezőket, az alábbi táblázat szerint:

Mező	Szín	RGB
Üres	#32C832	(50, 200, 50)
Víz	#3264C8	(50, 100, 200)
Fa	#C86432	(200, 100, 50)
Kő	#C8C8C8	(200, 200, 200)
Gally	#F0B478	(240, 180, 120)
Bogyó	#FF0000	(255, 0, 0)
Répa	#FAC800	(250, 200, 0)

Játékosok csatlakozása

A pálya betöltése után a karakterek csatlakoznak a játékhoz. A játék kezdete előtt tetszőleges mennyiségű karakter csatlakozhat. A játék megkezdéséhez legalább 2 karakternek lennie kell a pályán és pontosan egynek kell ember által irányítottnak lennie. A csatlakozási sorrend tetszőleges. Csatlakozás után a csatlakozó karakter kap 4 darab véletlenszerű alap nyersanyagot (fa, kő, gally, bogyó, répa). Egy fajta nyersanyagból többet is kaphat. Ez nyilván azt jelenti, hogy nem mindegyik nyersanyagból kap.

A csatlakozott karaktert úgy kell elhelyezni a pályán, hogy az eddig csatlakozott karakterektől legalább 50 távolságra legyen (légvonalban). Amennyiben ez nem megoldható, akkor az 50-es thresholdot 5-ösével csökkentve próbáljuk meg elhelyezni a karaktert (tehát ha nincs olyan mező, ami legalább 50 távolságra lenne, akkor keressünk olyan mezőt, ami legalább 45 távolságra található. Ha ilyen sincs, akkor 40, 35...).

A csatlakozott karakter csak üres mezőre kerülhet, és a pozíciójának egész koordinátának kell lennie. Egy új karakter csatlakozásának hatására a már pályán lévő karakterek pozíciói nem módosulhatnak.

A játék megkezdése után már nem lehet csatlakozni.

Játékmód

A játék két módban indítható el: normál és tutorial módban. A játékmódot a játékosok csatlakozása előtt kell kiválasztani. Az alapértelmezett a normál mód. Tutorial módban a gépi játékosok ugyanúgy csatlakoznak a játékhoz és ugyanúgy el kell helyezni őket, és a játék teljes részét képzik, viszont nem csinálnak semmit, tehát a játék során végig (boldogan) egy helyben állnak, amíg meg nem halnak. Normál módban a gépi játékosok is végeznek cselekvéseket.

Játék vége

A játék akkor ér véget, ha már csak 1 karakter marad életben vagy ha az emberi játékos meghal. Ezután már semmilyen cselekvés nem végezhető, az idő sem telik tovább.

A karakterek

Karakterek tulajdonságai

A karaktereknek 2 fő tulajdonsága van.

- **Életerő (HP)**: a karakterünk életereje (max: 100). Ha elérjük a 0-t, akkor a karakter meghal.
- **Éhség (hunger)**: a karakterünk jóllakottsága (max: 100). Ha 100, azt jelenti, hogy nem éhes. Ha 0 (0 alá nem mehet), akkor éhezik. Ha egy karakter éhsége 0, akkor minden időegység eltelte után 5 életerőt veszít.

Inventory

A karakterek az összegyűjtött tárgyakat (itemeket) az inventory-jukban gyűjthetik. Az intentory-ban 10 hely (slot) van. Minden sloton csak egyféle item lehet. Bizonyos itemek stackelhetőek, azaz egy sloton több is lehet belőlük. Ha egy slot betelik, és a karakter begyűjt egy olyan itemet, akkor azzal új slotot kell kezdeni. Tehát rendelkezhetünk több itemmel egy adott típusból, de akkor több slotot is fognak elfoglalni.

A inventory-hoz ha hozzáadunk egy itemet, akkor a már megkezdett slothoz fogjuk hozzáadni. Ha oda nem fér (vagy nincs még ilyen itemünk vagy nem stackelhető), akkor a legelső üres slotra kell betenni. Ebből adódóan az itemek sorrendje is fontos. Az inventory-ból továbbá eldobhatjuk az itemeket, megcserélhetjük őket, átmozgathatjuk őket, vagy akár több stacket kombinálhatunk egymással.

Kraftoláskor egy új tárgy keletkezik, amelynek ugye költségei is vannak. Először az előállításhoz szükséges nyersanyagok tűnnek el az inventory-ból, utána adódik hozzá a lekraftolt item. Az előállítási költségként felmerülő nyersanyagok mindig az inventory elejéről vonódnak le (ha több stack van egy adott nyersanyagból). Ha a lekraftolt item nem fér el az inventory-ban, akkor a pályán arra a pozícióra kerül, ahol a karakter éppen áll. A tábortűz lekraftolás után nem kerül az inventory-ba, hanem rögtön az aktuális mezőre kerül. Tábortűz csak olyan mezőre kerülhet, ahol nincs egyéb tereptárgy (pl. fa, kő, stb.). Ha egy olyan mezőn állunk kraftoláskor, ahol már van tereptárgy, akkor ne történjen semmi.

Az egyes nyersanyagokhoz tartozó maximális stack méret:

Név	Stack méret
Fejsze	1 (nem stackelhető)
Csákány	1 (nem stackelhető)
Fáklya	1 (nem stackelhető)
Lándzsa	1 (nem stackelhető)
Fa	15
Kő	10
Gally	20
Nyers répa	10
Főtt répa	10
Nyers bogyó	10
Főtt bogyó	10

Idő telése

Az idő folyamatosan telik, ilyenkor minden játékos egymás után végrehajt egy cselekvést. Először az emberi játékos cselekszik, utána a gépi ellenfelek csatlakozási sorrendben, majd ezután mozdul 1-et az óra (tehát eltelik egy időegység).

Minden időegység eltelésekor minden karakter jóllakottsága csökken 0.4-del. Ha a jóllakottság egy adott körben 0-ra csökken, akkor az életerő már abban a körben csökken.

Cselekvések

A karakterek a játék során az alábbi cselekvéseket hajthatják végre:

Cselekvés	Leírás
Várakozás	A karakter nem csinál semmit
Lépés	A karakter lép a pályán, megadott irányban (jobbra, balra, fel, le)
Lépés és támadás	A karakter lép a pályán, megadott irányban, majd támad
Favágás	A karakter fát vág
Kőbányászás	A karakter követ bányászik
Támadás	A karakter megtámadja a hozzá legközelebb álló ellenfelet
Kraftolás	A karakter egy megadott tárgyat lekraftol
Begyűjtés	A karakter begyűjt egy tárgyat az aktuális mezőről
Eldobás	A karakter eldob egy kiválasztott tárgyat az aktuális mezőre
Tárgyhasználat	A karakter egy adott tárgyat használ (vagy a kezébe veszi, vagy megeszi)
Főzés	A karakter egy répát vagy bogyót megfőz
Tárgymozgatás	A karakter az inventory-ban egy tárgyat másik slotra helyez
Tárgycsere	A karakter az inventory-ban két tárgyat megcserél egymással
Tárgykombinálás	A karakter az inventory 2 slotján lévő (azonos típusú) tárgyat egyesíti

A karakterek a pályán lépésekkel tudnak mozogni. Alapvetően minden karakter sebessége 1, azaz lépésenként 1 mező megtételére képes. A sebesség azonban csökkenhet, amennyiben a karakter életere és/vagy éhsége alacsony.

A karakterek nem léphetnek olyan mezőre, amin víz van. Ha egy adott karakter a lépés után egy olyan mezőre kerülne (az a mező lenne a legközelebb hozzá), ahol víz van, akkor a cselekvésnek nem lesz semmilyen hatása, a karakter marad az eredeti pozíción.

Az életerő az alábbi módon befolyásolja a karakter mozgási sebességét:

Életerő	Szorzó
[50%;100%]	1
[30%;50%)	0.9
[10%;30%)	0.75
(0%;10%)	0.6

Az éhség az alábbi módon befolyásolja a karakter mozgási sebességét:

Éhség	Szorzó
[50%;100%]	1
[20%;50%)	0.9
(0%;20%)	0.8
0%	0.5

Például ha a karakternek 38%-os életereje és 15%-os éhsége van, akkor ez egy 0.9-es és egy 0.8-as szorzót jelent, tehát a sebessége 1 * 0.9 * 0.8, azaz 0.72 lesz.

A karakterek egy adott pillanatban lehetnek két mező között is (tehát a koordinátákat valós értékként számoljuk), azonban mindig "tartoznak" egy adott mezőhöz, mégpedig ahhoz, amelyik hozzájuk a legközelebb esik. A távolságot a teljes játék során valós távolságként számoljuk, tehát két egymás melletti átlós mező távolság $\sqrt{2}$, azaz $\sim$1.41 egység.

Ételek megfőzése

A nyers répát és nyers bogyót meg lehet főzni. A főtt répát és bogyót már nem lehet még egyszer megfőzni. Ha egy répát megfőzünk, akkor főtt répát, míg ha egy bogyót megfőzünk, akkor főtt bogyót kapunk.

Az inventory-nkban lévő répát és bogyót úgy tudjuk megfőzni, ha egy olyan mezőn állunk, ahol egy tábortűz található (és van az inventory-nkban répa vagy bogyó). A főzés ideje 1 időegység. A főzés hatására az inventory-ba egy főtt répa vagy bogyó kerül (és a nyers nyilván eltűnik).

Ételek elfogyasztása

Az ételeket el tudjuk fogyasztani. Az elfogyasztott ételek hatását az alábbi táblázat mutatja be:

Név	Jóllakottság	Élet
Nyers bogyó	+20	-5
Nyers répa	+12	+1
Főtt bogyó	+10	+1
Főtt répa	+10	+3

Kraftolás

A játékosok által kraftolható tárgyak listája:

Név	Költség	Élettartam
Fejsze	3 gally	40 használat
Csákány	2 farönk, 2 gally	30 használat
Lándzsa	2 farönk, 2 kő	10 használat
Fáklya	1 farönk, 3 gally	20 időegység
Tábortűz	2 gally, 2 farönk, 4 kő	60 időegység

Támadás

A karakterek meg tudják támadni a hozzájuk legközelebb eső (de legfeljebb 2 egység távolságra lévő) ellenséges karaktert. A támadás hatására az ellenfél karaktere megadott mennyiséget sebződik. A sebzés mértéke a támadó kezében tartott eszközétől függ:

Eszköz	Sebzés	Mellékhatás
Lándzsa	19	lándzsa élettartama 1-gyel csökken
Fejsze	8	fejsze élettartama 1-gyel csökken
Csákány	8	csákány élettartama 1-gyel csökken
Fáklya	6	-
-	4	-

Ha egy karakter meghal (bármilyen oknál fogva), akkor először elejti a kezében lévő tárgyat, majd az inventory-jának teljes tartalma az aktuális mezőre kerül (az inventory elejétől kezdve, sorrendben).

Segítség, linkek

- Játék linkje
- Wiki
- Videó

Egyéb megjegyzések

- A jóllakottság egy körön belül mehet 100 fölé, de a kör végével már maximum 100 lehet. Pl. ha a jóllakottságunk 81 és megeszünk egy nyers bogyót, akkor a jóllakottság ezzel 101 lesz, viszont a kör végén ez csökken 0.4-del, tehát a kör végén 100 lesz (nem pedig 99.6).
- A karakterek által végrehajtott cselekvés (getLastAction()) csak olyan action lehet, ami már eredetileg is szerepelt a programban.
- A feladat megoldása során mindenhol a GameManager osztályban lévő Random objektumot kell használni (ha szükségünk van véletlen szám generálásra). Új random objektumot létrehozni, vagy más módon generálni véletlenszerű számot nem megengedett.
- Az AI tesztelésénél az alábbi szempontok alapján számolódik a pont: sikerült-e legyőzni az emberi játékost, hány AI karakter maradt életben és mennyi idő alatt lett vége a játéknak. Elsődleges cél a győzelem. Másodlagos cél, hogy minél több AI életben maradjon. Harmadlagos cél, hogy minél gyorsabban sikerüljön legyőzni a játékost.