Programozás I. Gyakorló feladatsor

SZTE Szoftverfejlesztés Tanszék

2023. tavasz

Általános követelmények, tudnivalók

- A feladat elkészítési határideje: **vasárnap 23:59:59**. Ez szigorú határidő, a Bíró előre megadott időben zár, pótlásra nincs lehetőség.
- A feladatokat számítógép előtt kell megoldani, tetszőleges fejlesztői környezetben, tetszőleges operációs rendszer segítségével.
- Az elkészült programot **20** alkalommal lehet benyújtani, a megadott határidőig.
- Csak a leírásban szereplő osztályokat, metódusokat és adattagokat kell megvalósítani, egyéb dolgokért nem jár plusz ont.
- A feladat megoldása során minden megadott előírást pontosan követni kell! Tehát, ha a feladat leírása szerint egy adattag neve a "elsoFoku", akkor az alábbi elnevezések nem megfelelőek: "elsőFokú", "elsofoku", "elso_foku", "elsőFoq". Ugyanez igaz a metódusok, osztályok elnevezésére is!
- A metódusok esetében a visszatérési típus, a név, módosítók és a paraméterek típusai (és azok sorrendje) kerülnek ellenőrzésre, azonban a paraméterek nevei tetszőlegesek lehetnek.
- Az órán tanult konvenciókat követni kell (getter/setter elnevezés, toString, indentálás, stb). Abban az esetben is, ha ezt a feladat külön nem emeli ki, az ellenőrzés során erre is építünk.
- A nem forduló kódok nem kerülnek kiértékelésre, ezt utólagosan a gyakorlatvezető sem bírálhatja felül. (Hiszen mindenki rendelkezésére áll a saját környezete, ahol fordítani, futtatni tudja a forráskódot, így feltöltés előtt ezt mindenképpen érdemes megnézni!)
- Az adattagok és konstruktorok hiányában garantáltan 0 pontos lesz a kiértékelés, mert ezek minden teszt alapját képezik.
- Ha végtelen ciklus van a programban, akkor ezt a Bíró ki fogja dobni 3 másodperc után (ha többször is meghívásra kerül ilyen metódus, akkor ez többszöri 3 másodperc, összesen akár 2 perc is lehet). Ilyenkor NE kattints még egyszer a *Feltöltés* gombra, mert akkor kifagyhat a Bíró, csak a böngésző újraindításával lehet megoldani a problémát (emellett elveszik 1 feltöltési lehetőség is).
- Kérdés/probléma esetén a gyakorlatvezetők tudnak segítséget nyújtani.
- **A feladat megoldása során a default csomagba dolgozz, majd a kész forrásfájlokat tömörítve, zip formátumban töltsd fel, azonban a zip fájlt tetszőlegesen elnevezheted!**

  - 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 feladatokban az alábbi dolgok az alapértelmezettek (**kivéve**, ha a feladat szövege mást mond)
  - az osztályok láthatósága publikus
  - az egész érték 32 bites
  - a lebegőpontos számok dupla pontosságúak
  - az olyan metódusok void visszatéréssel rendelkeznek, amelyeknél nincs specifikálva visszatérési típus.
  - a metódusok mindenki számára láthatóak
  - az adattagok csak az adott osztályban legyenek elérhetőek

- A *riport.txt* és a fordítási log fájlok megtekinthetőek az alábbi módon:
  1. Az *Eredmények megtekintése* felületen a vizsgálandó próba új lapon való megnyitása
  2. A kapott url formátuma: `https://biro2.inf.u-szeged.hu/Hallg/IB204L/FELADAT/hXXXXXX/4/riport.txt`
  3. Az url-ből visszatörölve a 4-esig (`riport.txt` törlése) megkaphatók a 4-es próbálkozás adatai.

- Szövegek összehasonlításánál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")

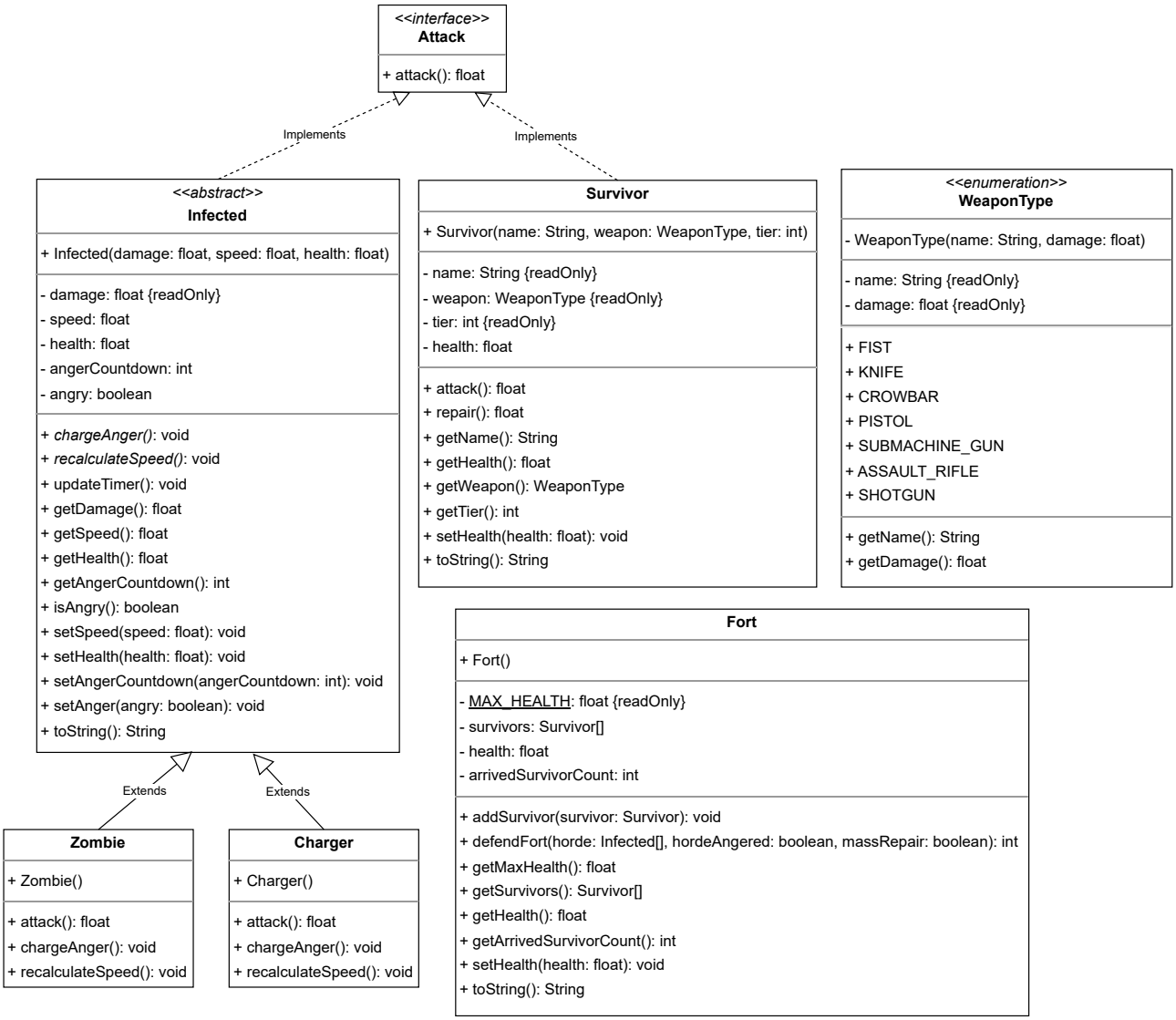
- A leírásokban bemutat példákban a stringek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül az `3 alma`, de a szóköz szükséges!

- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).

Zombi Inváció

A Föld káoszba borult. Egy új vírus söpört végig az összes kontinensen. Az embereket fertőzött húsevő lényekké változtatta. Azok a kevesek, akik a vírust elkerülve épségben maradtak, a túlélésért küzdenek egy olyan új világban, ahol az élőhalottak dominálják a mindennapokat.

1. ábra. Zombie Invasion osztálydiagram



Egy kis lábjegyzet a diagrammhoz:

Az aláhúzással jelölt adattag, azt ábrázolja, hogy az osztályhoz tartozik, nem az objektumhoz. A *dőlttel* jelölt metódus, azt ábrázolja, hogy az elkészítését még az osztály végzi, de implementálását, majd más osztály(ok) hajtják végre.

Kicsi összefoglaló az enumokhoz:

Az enum típus olyan, mint az interfészek és osztályok egyesítése olyan értelemben, hogy definiálhatunk bennük konstansokat, meghatározott értékkel, ugyanakkor ezek a konstansok nem egy egész értéket képviselnek, hanem az enum adattípus egy-egy konkrét objektumait, amik static és final módosítóval rendelkeznek. És mint ilyen, az enum minden objektuma rendelkezhet saját adattaggal, adattagokkal, illetve metódusokkal, valamint konstruktorokkal. Amikor felsoroljuk az enum értékeket, akkor mögöttük egy paraméter listával megadhatjuk azokat a paramétereket, amelyek ahhoz kellenek, hogy az enum konstruktora meghívódhasson az egyes értékek inicializálásakor. Ezek lesznek az adott enum objektumhoz tartozó értékek.

Attack Interface (2 pont)

Írj egy `Attack` nevű interface-t, ami egy élőlényt felruház sebzés osztás lehetőségével. A szükséges metódusokat az 1. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

WeaponType Enum (4 + 2 pont)

Írj egy `WeaponType` nevű enumerációt, ami egy fegyver fajtát reprezentál. Az adattagokat, valamint a szükséges metódusokat az 1. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Az adott enum objektumhoz tartozó értékek:

- `FIST` neve legyen: `Okol`, sebzése: `5`.
- `KNIFE` neve legyen: `Kes`, sebzése: `15`.
- `CROWBAR` neve legyen: `Feszitovas`, sebzése: `30`.
- `PISTOL` neve legyen: `Pisztoly`, sebzése:`45`.
- `SUBMACHINE_GUN` neve legyen: `Gepfegyver`, sebzése: `60`.
- `ASSAULT_RIFLE` neve legyen: `Gepkarabely`, sebzése: `75`.
- `SHOTGUN` neve legyen: `Soretes Puska`, sebzése: `100`.

Konstruktor (2 pont)

A fegyverfajta adattagjait a paraméterneveknek megfelelően állítsuk be.

Getterek (2 pont)

A getter metódusok értelemszerűen a megfelelő adattagok értékeit adják vissza.

Infected absztrakt osztály (15 + 1 pont)

Írj egy `Infected` nevű absztrakt osztályt, ami egy fertőzött élőlényt reprezentál. Az adattagokat, valamint a szükséges metódusokat az 1. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (2 pont)

A fertőzött adattagjait a paraméterneveknek megfelelően állítsuk be. Az *angerCountdown* adattag alapból 0-át, az *angry* adattag meg hamis értéket vegyen fel.

Getterek (5 pont)

A getter metódusok értelemszerűen a megfelelő adattagok értékeit adják vissza.

Setterek (4 pont)

A setter metódusok értelemszerűen a megfelelő adattagok értékeit állítsák be.

Metódusok (4 pont)

Az `updateTimer()` metódus frissíti a bedühödésig tartó támadási időt úgy, hogy bedühödés hatására, egyfolytában csökkenti ennek az időnek az értéket, ha 0-ra ér, akkor dühbe kerül az adott fertőzött. A düh 1 időegységig tart. **(2 pont)**

A `toString()` metódus egy fertőzött adatait adja vissza. Az alábbi formátumba legyen: "*Fertozott tipusa: {classNeve}, Alap sebzese: {damage}, Sebessege: {speed}, Meglevo eletpontja: {health}*" **(2 pont)**

Zombie osztály (8 + 1 pont)

Írj egy `Zombie` nevű osztályt, ami egy különleges fertőzött Zombit reprezentál. Az adattagokat, valamint a szükséges metódusokat az 1. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (2 pont)

Az ősosztály konstruktorának adattagjait ezen értékekkel állítsuk be.

- `Damage`: 10, `Speed`: 1, `Health`: 100.

Metódusok (6 pont)

Az `attack()` metódus azt adja vissza, hogy a zombi támadása esetén mennyi sebzést oszt. A sebzést úgy számoljuk, hogy `damage * speed`. Ha dühös, akkor kétszer annyi sebzést oszt. **(2 pont)**

Az `chargeAnger()` metódus reprezentálja a zombinak a bedühödését, a bedühödés 1 időegységebe fog kerülni.**(2 pont)**

Az `recalculateSpeed()` metódus frissíti a zombinak a sebességét ezen értékek alapján: **(2 pont)**

- `Health`: < 75 := `Speed`: 0.85.
- `Health`: < 50 := `Speed`: 0.7.
- `Health`: < 25 := `Speed`: 0.5.

Charger osztály (8 + 1 pont)

Írj egy `Charger` nevű osztályt, ami egy különleges fertőzött Charger-t reprezentál. Az adattagokat, valamint a szükséges metódusokat az 1. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (2 pont)

Az ősosztály konstruktorának adattagjait ezen értékekkel állítsuk be.

- `Damage`: 30, `Speed`: 0.7, `Health`: 150.

Metódusok (6 pont)

Az `attack()` metódus azt adja vissza, hogy a charger támadása esetén mennyi sebzést oszt. A sebzést úgy számoljuk, hogy `damage * speed`. Ha dühös, akkor 1,75-szeres sebzést oszt. **(2 pont)**

Az `chargeAnger()` metódus reprezentálja a chargernek a bedühödését, a bedühödés 2 időegységebe fog kerülni.**(2 pont)**

Az `recalculateSpeed()` metódus frissíti a chargernek a sebességét ezen értékek alapján: **(2 pont)**

- `Health`: < 120 := `Speed`: 0.6.
- `Health`: < 80 := `Speed`: 0.5.
- `Health`: < 40 := `Speed`: 0.4.

Survivor osztály (16 + 1 pont)

Írj egy `Survivor` nevű osztályt, ami egy túlélőt reprezentál. Az adattagokat, valamint a szükséges metódusokat az 1. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (2 pont)

A túlélő adattagjait a paraméterneveknek megfelelően állítsuk be. A *tier* adattag 1 és 5 közötti (zárt intervallumba) értéket vehet fel. Alapból egy túlélőnek az élete 100 egység.

Getterek (4 pont)

A getter metódusok értelemszerűen a megfelelő adattagok értékeit adják vissza.

Setterek (1 pont)

A setter metódusok értelemszerűen a megfelelő adattagok értékeit állítsák be.

Metódusok (9 pont)

Az `attack()` metódus azt adja vissza, hogy a túlélő mennyit sebez. A birtokolt fegyver sebzésének megfelelő sebzéssel fog sebezni. **(2 pont)**

A `repair()` metódus azt adja vissza, hogy a túlélő mennyi egységgel fogja javítani az erődítményt. A túlélő szintjével megfelelő értékkel javítja: **(3 pont)**

- Szint: `1` := `10`
- Szint: `2` := `20`
- Szint: `3` := `30`
- Szint: `4` := `40`
- Szint: `5` := `50`

A `toString()` metódus egy túlélő adatait adja vissza. **(4 pont)**

Az alábbi formátumba legyen, ha fegyver van nála: "*A nevem: {name}, a tulelo szintem: {tier}, a kezembe tartott fegyver: {weapon}, a maradek eletpontom {health}*"

Az alábbi formátumba legyen, ha nem rendelkezik fegyverrel: "*A nevem: {name}, a tulelo szintem: {tier}, az okleimmel harcolok, a maradek eletpontom {health}*"

Fort osztály (25 + 1 pont)

Írj egy `Fort` nevű osztályt, ami egy erődítményt reprezentál. Az adattagokat, valamint a szükséges metódusokat az 1. ábrán láthatjuk. Az arrivedSurvivorCount adattagja az erődítményben lévő túlélők számát adja meg. A *MAX_HEALTH* adattag alapértéke 850 egység legyen. Ügyelj a megfelelő láthatóságokra.

Konstruktor (2 pont)

Az arrivedSurvivorCount adattag értékét 0-val, az erődítmény életerejét a *MAX_HEALTH* adattag értékével, és a túlélőket tároló tömböt, pedig egy új 5 hosszúságú megfelelő típusú tömbbel inicializáljuk.

Getterek (4 pont)

A getter metódusok értelemszerűen a megfelelő adattagok értékeit adják vissza.

Setterek (1 pont)

A setter metódusok értelemszerűen a megfelelő adattagok értékeit állítsák be.

Metódusok (18 pont)

Az `addSurvivor()` metódus hozzáadja a paraméterben érkező túlélőt, az erődítmény védelmezése érdekében. Ha betelne az erődítmény, akkor a túlélők azt kibővítik úgy, hogy az előző befogadó mennyiség a kétszeresére nő, és így már befér az új túlélő is. **(4 pont)**

A `defendFort()` metódus felelős 1 időegységnyi fertőzött támadás lebonyolítására. Ezen támadás kimenetele úgy néz ki, hogy először mindig a túlélők fognak védekezni, majd aztán a fertőzöttek támadni. A túlélők sebzés osztása sorba történik, először a legelső zombit támadják, ha meghalt, akkor aztán a másodikat, ha az is meghalt, akkor a harmadikat, és így tovább, ameddig el nem fogy az összes túlélő sebzés mennyiség. A fertőzöttek sebzés osztása is ugyanígy történik, de ha áll még az erődítmény, akkor először azt kell lerombolniuk, hogy a túlélőket megtudják támadni. Ezután a halott fertőzött(ek) és túlélő(k) ebben az időegységben törlődnek, majd a még életben lévők átveszik azoknak a helyeit sorba. A metódus visszatérési értéke attól függ mi lett a harc kimenetele. Ha a harc még nem fejeződött be (tehát még van túlélő és fertőzött is), akkor térjünk vissza 0-val. Ha minden túlélő meghalt, akkor a fertőzöttek nyertek, térjünk vissza 1-gyel. Ha minden fertőzött meghalt, akkor a túlélők nyertek térjünk vissza 2-vel. (pro tip: Frissítsük a fertőzöttek értékeit.)
(10 pont)

A `toString()` metódus egy erődítmény adatait adja vissza. **(4 pont)**

A *tuleloCount* érték az életben lévő túlélők mennyiségét jelzi.
A *healthPercentage* érték az erődítmény életerejét mutatja százalékosan formában, 2 tizedesjegyre kerekítve.

Az alábbi formátumba legyen, ha az erődítmény még áll: "*Az eroditmeny eletereje {healthPercentage}%-on van, es {tuleloCount} tulelo van meg eletben.*"

Az alábbi formátumba legyen, ha az erődítményt már lerombolták: "*A megsemmisult eroditmeny romjai kozott {tuleloCount} tulelo van meg eletben.*"

Jó munkát!