Programowanie 2

Zadanie E

autor: Krzysztof Rajda

Ilość punktów: **11**

Ave, Caesar, morituri te salutant!

Krwawo stłumiwszy powstanie Galów, pokonawszy Pompejusza, najgroźniejszego politycznego wroga i rozkochawszy w sobie piękną Kleopatrę, przed objęciem nieograniczonej władzy w Rzymie Cezara czekało już tylko jedno wyzwanie - zaskarbić sobie przychylność rzymskiego ludu.

W tym celu Cezar polecił zgromadzić najwaleczniejszych gladiatorów Republiki i zorganizować czterodniowy triumf, którego nie widział jeszcze świat - bowiem w swej mądrości wiedział, że prosty lud nie pragnie niczego tak mocno, jak igrzysk.

To zadanie powierzone zostało właśnie Tobie. Tylko od Twoich zdolności planowania zależy, czy Cezar zdoła zapanować nad obywatelami Wiecznego Miasta…

Wymagania techniczne:

- w pierwszej linii kodu zamieść komentarz ze swoim imieniem i nazwiskiem
- jedyna dozwolona biblioteka zewnętrzna to *iostream*

1. Na igrzyska przybyli gladiatorzy ze wszystkich stron świata, a z dzikich ostępów ściągnięto groźne bestie. Mają oni wspólny cel - walczyć na arenie. Przeżyć i zdobyć wieczystą chwałę, lub zginąć ku uciesze gawiedzi.

   Stwórz klasę abstrakcyjną *Gladiator*, zawierającą **metody publiczne:**

   a. konstruktor domyślny *Gladiator()*
   b. *unsigned int Gladiator::getRemainingHealth()* zwracającą pozostałą gladiatorowi ilość punktów życia w skali [0, 100], zaokrąglane w dół
   c. *unsigned int Gladiator::getDamage()* określającą ilość obrażeń które gladiator zada wrogowi
   d. *void takeDamage(unsigned int damage)* która powoduje spadek ilości punktów życia gladiatora o zadaną parametrem wartość (z uwzględnieniem indywidualnych parametrów klasy)
   e. *unsigned int Gladiator::getRapidity()* określająca ilość punktów szybkości wojownika, która wpływa na kolejność zadawania ciosów i zdolność do uników
   f. *void Gladiator::applyWinnerReward()* aplikująca bonus z wygranej
   g. void *Gladiator::cure()* przywracająca gladiatora do pełni zdrowia
   h. *void Gladiator::printParams()* zwracającą parametry gladiatora w postaci napisu o formacie:
      ID:%_PKT_ZYCIA:PKT_ATAKU:PKT_SZYBKOŚCI

   oraz **metodę niepubliczną**:

   h. *void Gladiator::die()* która pozbawia gladiatora życia (zeruje punkty życia)

2. Igrzyska mają swojego sponsora, który oprócz zaskarbienia sobie przychylności widzów, jest także panem życia i śmierci gladiatorów.

   W naszym przypadku sponsorem jest *Caesar*. Poprzez gest kciuka skazuje on bądź ułaskawia pokonanego zawodnika - w metodzie *judgeDeathOfLife(Gladiator*)*. Jego zamiłowanie do matematycznych zabaw jest ogólnie znane. A że na arenie może się pobawić życiem poddanych - cóż, przywilej władcy. **Cezar skazuje na śmierć co trzeciego poddanego osądowi, ale tylko jeśli ilość zadanych ciosów w danej walce była parzysta.**

   Jako niepodzielny władca życia gladiatorów, sponsor ma dostęp do ich wnętrza niezależnie od ich ustawień prywatności.

3. Walki toczą się w amfiteatrze, walczyć mogą ze sobą gladiatorzy dowolnych profesji.

   Zaimplementuj więc klasę *Amphitheatre* i w niej publiczną metodę *Amphitheatre::fight(Gladiator*, Gladiator*).* Amfiteatr w konstruktorze powinien przyjmować obiekt klasy *Ceasar**

   Walkę zaczyna gladiator z większą wartością szybkości (lub podany w pierwszym argumencie, gdy mają jej po równo). Dalej nazywać go będziemy *pierwszym gladiatorem*. Jego przeciwnik staje się *drugim gladiatorem*.

   Walka polega na naprzemiennej wymianie ciosów w następujących krokach:
   1. pierwszy gladiator zadaje przeciwnikowi obrażenia zgodnie z własną metodą *getDamage*
   2. rozpatrywane są skutki ciosu pierwszego gladiatora
   3. jeśli drugi gladiator jest w stanie to zrobić, zadaje swój cios przeciwnikowi
   4. rozpatrywane są skutki ciosu drugiego gladiatora
   5. ogłaszane są statystyki wymiany ciosów - metoda *printParams* najpierw dla pierwszego gladiatora, potem dla drugiego

   Walka toczy się do momentu, aż jednemu z wojowników wartość życia spadnie poniżej 10%. Wtedy sponsor decyduje o tym, czy pokonany powinien przeżyć, zgodnie z metodą *judgeDeathOfLife*.

   Jeśli wskutek potężnego ataku przeciwnika wartość życia gladiatora spadnie do 0%, umiera on bez pytania sponsora o pozwolenie.

   Jeśli po wymianie 20 ciosów walka nadal nie przyniosła rozwiązania, sponsor pytany jest o decyzję odnośnie dalszych losów każdego z gladiatorów (najpierw pierwszego, potem drugiego), a walka jest przerywana.

   Na koniec całej walki (po rozpatrzeniu nagród i leczenia) także wyświetlane są statystyki - najpierw pierwszego, potem drugiego gladiatora.

   Do walki w ogóle nie dochodzi, gdy jeden ze przeciwników jest martwy od początku.

   Zwycięzca pojedynku (a zwycięzcą jest ten, kto pozostał żywy po walce) zwiększa wartość swoich parametrów ataku i szybkości o 2pkt (metoda *applyWinnerReward()*)

oraz jest leczony do pełni zdrowia (metoda *cure()*), zanim będzie w stanie przystąpić do kolejnej walki

4. Na arenie walczą gladiatorzy różnych typów:
    a. ludzie - klasa *Human*. Podczas otrzymywania obrażeń (*takeDamage*) ich wartość jest pomniejszana o ilość punktów obrony wynikającą ze zbroi i ilość punktów szybkości, określającą zdolność do uników. Ludzie bazowo mają 200 pkt życia, 30 pkt ataku, 10 pkt szybkości i 10 pkt obrony. Identyfikatorem ludzi jest ich imię, podawane w konstruktorze

    b. zwierzęta - klasa *Beast*. Podczas otrzymywania obrażeń, pomniejszamy je o połowę punktów szybkości (mniejsze zdolności planowania niż u ludzi), zaokrągloną w dół. Zwierzęta mają natomiast silny instynkt przetrwania, więc gdy ich ilość punktów życia spadnie poniżej 25%, zadają dwukrotnie silniejsze obrażenia. Zwierzęta bazowo mają 150 pkt życia, 40 pkt ataku i 20 pkt szybkości. Identyfikatorem zwierząt jest ich gatunek, podawany w konstruktorze

    c. człowiekiem będący, acz bardziej bestii podobny wojownik z północy - *Berserker,* uzbrojony w bojowy topór (35 pkt ataku) i małą tarczę (15 pkt obrony). Wielki topór i jeszcze większe mięśnie powodują, że jego szybkość pozostawia wiele do życzenia (5 pkt). Jako człowiek, ma 200 pkt życia.
    Znaczny upływ krwi wzbudza u niego bojowy szał, więc gdy ilość pkt życia spadnie poniżej 25%, przemienia się w dziką bestię, odrzucając tarczę i zadając dwukrotnie mocniejsze ciosy

    d. gladiatorzy mogą także walczyć w grupach:
        - do grupy dodaje się gladiatorów przy pomocy metody *void Squad::addGladiator(Gladiator*)*.
        - identyfikatorem grupy jest napis *“Squad”*
        - wartość ataku grupy jest sumą punktów ataku jej członków.
        - każdy członek grupy otrzymuje średnią wartość zadanych grupie obrażeń (wartość ataku przeciwnika/ilość członków grupy) zaokrągloną w dół, pomniejszoną indywidualnie o wartości obrony i szybkości.
        - szybkość grupy jest szybkością jej najwolniejszego członka.

- ilość punktów życia grupy jest równa ilości punktów najmniej rannego członka (w rozumieniu *getRemainingHealth*) - z racji tego, po każdym otrzymanym ciosie inny członek grupy może być najmniej ranny.
- po śmierci któregoś z członków grupy (gdy jego punkty życia wg *getRemainingHealth* spadną do 0%), jest on usuwany z grupy
- nie można do grupy dodac martwego gladiatora
- nie można do grupy dodać gladiatora który już w niej jest

Przedstawione powyżej wartości parametrów są **bazowe**, a indywidualne parametry każdego z gladiatorów zależą od ilości wygranych przez niego walk.

Poniżej uproszczony diagram wymaganych klas. W standardzie UML pola i metody poprzedzone *"+"* są uważane za publiczne, poprzedzone "*-*" za niepubliczne