PROGRAMAZIO MODULARRA ETA OBJEKTUEI BIDERATUTAKOA

AZTERKETA PARTZIALA / FINALA, 2015ko Maiatzak 28

Izen Abizenak:_____

Azterketa Moldakor Informatizatuak (AMI)

Azterketa moldakor informatizatuak ebazteko aplikazio bat garatzeko eskatu digute. AMI[1] baten ezaugarria, berau osatzen duten ekintzen zerrenda da. AMI hauen berezitasuna, ekintza hauek aldez aurretik zeintzuk izango diren jakitea ezinezkoa dela da, exekuzio garaian hautatu eta aurkeztuko zaizkio erabiltzaileari.

AMI hauen funtzionalitatea ondorengoa da; ekintza bat ongi burutuz gero, garatu beharreko hurrengo ekintza apur bat zailagoa izango da, aldiz, ekintza gaizki burutuz gero, garatu beharreko hurrengo ekintza errazagoa izango da. Beraz, AMIak azterketa berezi batzuk izango dira, non, burutu beharreko ekintzak dinamikoki erabakiko diren azterketa aurrera doan heinean. Aurrez esan bezala, azterketa arruntetan ez bezala, AMIko ekintzak ez daude aldez aurretik erabakita, baizik eta aurreko ekintzaren erantzunaren ebaluazioaren eta zailtasun (*int*)[2] baten arabera erabakiko dira. AMI baten bitartez lortutako puntuazio finala (nota), positiboa, negatiboa edo zero izan daiteke.

AMI bat burutzeko, ekintza guztiak zailtasunaren arabera gutxienetik handienera lista batean ordenaturik egongo dira, alegia, EkintzenZerrenda klasean. Bertatik zenbait ekintza soilik aurkeztuko zaizkio erabiltzaileari, hau da, erabiltzaileak ez ditu EkintzenZerrendako ekintza guztiak burutu beharko, soilik horietako bakar batzuk, 1. irudiko algoritmoan ikus daitekeen bezala.

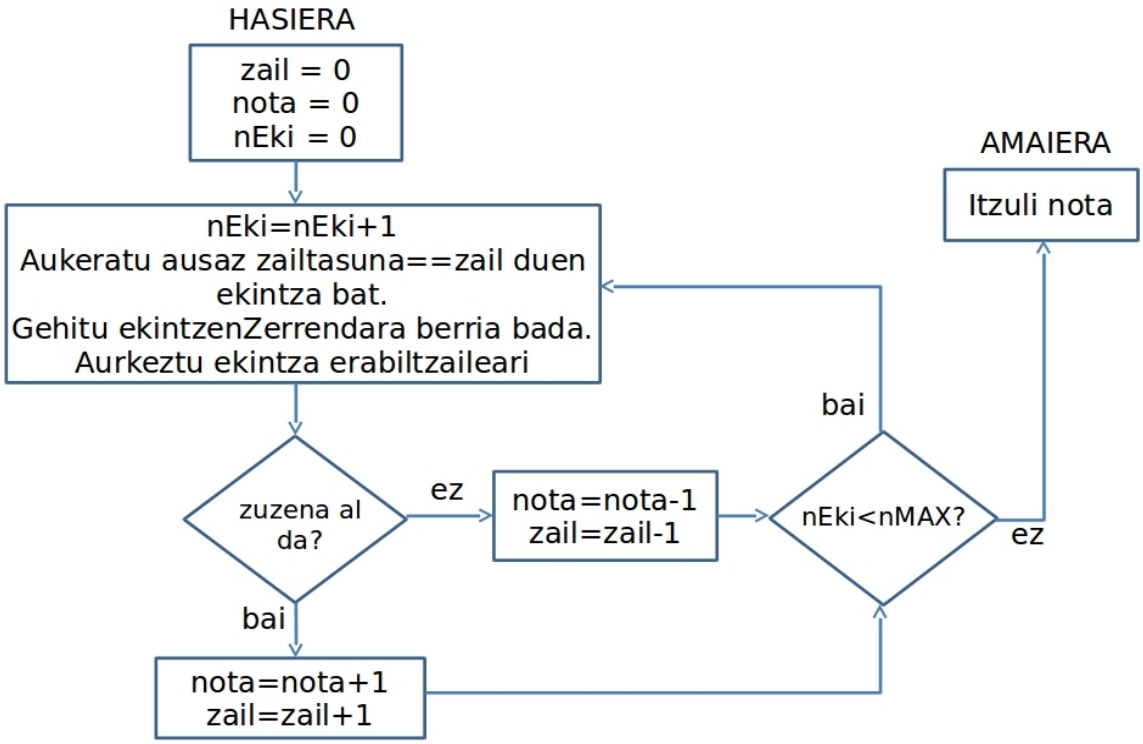
Lehenik, EkintzenZerrendan zailtasuna 0koa duen edozein ekintza hautatu eta erabiltzaileari eskainiko zaio. Erabiltzaileak ekintza ongi ebazten badu, zailtasuna 1 duten ekintzen artean edozein aukeratu eta erabiltzaileari eskainiko zaio. Aldiz, gaizki ebatziko balu, eskainiko zaion hurrengo ekintza -1 zailtasuna dutenen arteko bat izango da.

---

1 Ariketa honetan, AMIa bakarra izango da, hau da, une bakoitzean ikasle batek burutuko du azterketa.

2 Ekintza bakoitzaren zailtasuna osoko zenbaki baten bitartez aurkezten da. Ekintza errazek, zailtasun negatiboa dute, geroz eta zailtasun txikiagoa orduan eta ekintza errazagoa. Ekintza zailek ordea, zailtasun positiboa dute, geroz eta handiagoa orduan eta ekintza zailagoa. Zailtasuna zerokoa duten ekintzak, zailtasun ertainekoak dira. Adibidez, -4eko zailtasuna duen ekintza bat, -3ko zailtasuna duen ekintza baino errazagoa izango da. Zailtasuna 20koa duen ekintza bat, -1eko zailtasuna duen ekintza bat baino askoz ere zailagoa izango da. Esan bezala, zailtasun bereko ekintzak bata bestearen atzean egongo dira.

Prozesua behin eta berriz errepikatuko da AMIn aurredefinitutako (*static int nMAX*) eskaini behar diren ekintza kopurura iritsi arte. Laburbilduz, erabiltzaileari *zail* zailtasuna duen ekintza eskainiko zaio, ongi erantzuten badu, eskainiko zaion hurrengo ekintza *zail+1* zailtasunekoa izango da, bestela *zail-1*, *nEki* *nMax*-era iritsi arte.



AMI baten kudeaketa algoritmoa.

EkintzenZerrendan dauden ekintzak, mota desberdinekoak izan daitezke: aukera anitzeko galderak, erantzun irekia duten galderak, irudiak ordenatzeko galderak eta ekintza konposatuak. Ekintza guztiek (*int*) zailtasuna dute ezaugarri eta gainera *public boolean ebatzi()* metodoa izango dute. Metodo honen eginbeharra, ekintza pantailan erakustea, erabiltzailearen erantzuna jasotzea (horretarako aurrerago aurkeztuko den Teklatua klaseari zerbitzua eskatuko zaio) eta berau zuzentzea izango da. Metodoak ez du parametrorik behar, eta bere irteera *true* izango da ekintza ongi burutu bada, *false* bestela.

Aukera anitzeko nahiz erantzun irekia duten galderek, atributuen artean enuntziatua (*String*) izenekoa izango dute eta soluzioen zerrenda bat (*String*). Aukera anitzeko galderek soilik erantzun zuzen posible bakarra izango dute euren pantailaratzeko soluzioen zerrendan, horregatik, posizioa (*int*) ezaugarria izango dute, erantzun zuzenaren posizioa zehazteko (adib: zein da Arabako hiriburua? - 1. Gasteiz / 2. Bilbo / 3. Donostia / ...). Erantzun irekiko galderek, soluzioen zerrendan dauden aukera guztiak zuzenak lirateke (adib: zein da Arabako hiriburua? - Vitoria / Vitoria-Gasteiz / ...). Ezaugarri hauek kontuan izanda, bi galdera moten *ebatzi()* metodoa desberdina izango da: aukera anitzeko galderetan, erabiltzaileari, galdera eta soluzioen zerrenda osoa aurkeztuko zaizkio; ondoren, erabiltzaileak tekleatutako aukera (*int*)

irakurriko da; sartutako aukera posizio egokiarekin bat egiten badu, ekintza ontzat hartuko da. Erantzun irekiko galderetan, erabiltzaileari soilik galdera aurkeztuko zaio eta erabiltzaileak teklatutik sartutako erantzuna (*String*) irakurriko da. Erabiltzaileak sarturiko erantzuna, soluzioen zerrendako baten batekin bat egiten badu, orduan ekintza zuzentzat joko da.

Irudiak ordenatzeko galderetan, euren atributuen artean, irudien zerrenda bat (adib: haur baten irudia, heldu baten irudia eta gazte batena), ordenatzeko erabiliko den irizpidea (*String*), azken hau aplikatzean gorakor ordena jarraitu behar den adierazteko atributu boolear bat ere egongo da eta bukatzeko, erantzun zuzenaren posizioen zerrenda izango dira (posizioak *int* izango dira). Irizpidea, azken finean deskribapen bat izango da (adib: ordena alfabetikoki) eta gorakor atributua (*boolean*), irizpidea nola aplikatu behar den jakiteko erabiliko da, adibidez, haur-heldu-gazte adibidean, gorakorra *true* edo *false* izatearen arabera erantzuna bat edo beste izango da. Erantzuna ebaluatu ahal izateko posizioen zerrendan, irudien ordena egokia gordeko da (adibide honetan, irizpidea “adina” bada eta “gorakorra”=true: 1,3,2,...). Ekintza hau burutzeko, Irudia klaseko *ebatzi()* metodoan, erabiltzaileari lehenik ordenatze irizpidea eta gorazko edo beheranzko ordena jarraitu behar duen aurkeztuko zaio, eta ondoren irudi sorta. Ondoren, erabiltzaileari ordena sartzeko eskatuko zaio eta posizio zerrendako posizio guztiekin bat egiten badu, orduan ekintza zuzentzat joko da.

Azkenik, ekintza konposatuak, bi ekintza edo gehiagoz osaturikoak dira (diren motakoak direnak), beraz, euren ezaugarrien artean ekintzen zerrenda bat izango dute. Ekintza konposatu bat ontzat emango da, baldin eta berau osatzen duten ekintza guztiak banan banan ongi burutuak izan badira. Banakako ekintza hauetako bat gaizki burutzen denean, ekintza konposatua amaitu egingo da, txartzat joaz (alegia, ez dira ekintza konposatu hori osatzen duen gainontzeko ekintzarik aurkeztuko erabiltzaileari).

Erabiltzailearen erantzunak jasotzeko teklatutik, lagungarria izango zaizuen Teklatua EMA izeneko klasea duzue garaturik, beraz EZ DA inplementatu behar, bakarrik zerbitzua eskatu. Klase honen metodo publikoen artean, *public int irakurInt()* eta *public String irakurString()* ditu, non euren eginkizuna, erabiltzaileak teklatutik sarturiko letra edo zenbakia itzultzea den.

OHARRAK:

Penalizazioa jasoko duzue baldin eta:

- Beharrezkoak ez diren metodoen inplementazioak, nahiz eta beste testuinguru batean beharrezkoak izan.
- *Private* izan behar diren metodoak *public* jartzea.
- Orain arte ikusi eta erabilitako arauak ez jarraitzea, alegia, elementu desberdinen izenak hautatzeko arauak (atributuak, metodoak, parametroak, aldagaiak).
- Diseinuak informazio partekaketa maximizatzea lortzen **EZ** badu.

EBALUAZIO JARRAIA AUKERATU DUZUENOK

(Oharra: azterketa gainditzeko gutxienez 0.75 lortu behar da)

Eskatzen dena:

1. (**0.6 puntu**) **Klase diagrama** marraztu, adieraziz klase bakoitzeko atributu eta metodo guztiak, euren ikusgarritasuna argi adieraziaz (private –, public +, protected ▲). Klaseen arteko erlazioak ere zeintzuk diren adierazi. Diseinuak, informazio partekaketa maximizatzea lortu behar du.

2. (**0.6 puntu**) EkintzenZerrenda klaseari dagokion *public void pantailaratuEkiZailKopArabera()* **sekuentzia diagrama** garatu. Metodo honen eginbeharra, EkintzenZerrendan zailtasunaren arabera ordenaturik dauden ekintzen kopuruaren arabera pantailaratzea izango da (kopuru handienetik txikira).

   Oharra: Atal hau burutzeko orain arte aipatu ez diren bi klaseen beharra izango duzue, bertan tarteko informazioa gordetzeko:

   - *ZailtasunekoKopurua* klasea non bere ezaugarriak zailtasuna (*int*) eta kopurua (*int*) izango diren. Bertan, zailtasun bereko zenbat ekintza dauden gordeko da.

   - *ZailtasunekoKopuruZerrenda* klasea non bertan *ZailtasunekoKopurua* motako objektuak gordeko diren.

   Adibidez, 11 ekintza badaude, exekuzio adibidea:

```
3 ekintza 1 zailtasunekoa
2 ekintza 0 zailtasunekoa
2 ekintza 3 zailtasunekoa
1 ekintza -3 zailtasunekoa
1 ekintza -2 zailtasunekoa
1 ekintza -1 zailtasunekoa
1 ekintza 2 zailtasunekoa
```

3. (**0.3 puntu**) AMI klaseko *public int garatuAMI()* metodoa inplementa ezazu. Metodo honek, 1. irudiko algoritmoa jarraitzen du, une bakoitzean, erabiltzaileari aurkeztu beharreko ekintza bat aukeratzen du, *nMax* kopuru aldiz. Behin AMIa amaitzerakoan, berau burutzeko erabiltzaileari aurkeztu zaizkion ekintza guztiak objektu honi dagokion atributu batean gordek izango dira eta amaierako nota itzuliko da. Erabilitako metodo guztiak inplementatu beharko dira (zein klaseri dagozkien adieraziz), *public boolean ebatzi()* eta *bilatuZailtasunekoEkintza(int pZail)* izan ezik, soilik hauen deiekin nahikoa litzateke, hau da, espezifikazioekin.

4. (**1 puntu**) Ondoren aipatzen diren salbuespenen klaseak inplementatu itzazu, eta kasu bakoitzean eskatzen den soluzioa garatu:

    1. *public Ekintza bilatuZailtasunekoEkintza(int pZail)* metodoak EkintzaEzTopatuaException salbuespena jaurtiko du zerrendan pZail zailtasuneko ekintzarik aurkitzen ez bada. Kasu honetan, pantailan mezu bat agertuko da eta azterketaren kudeaketa bertan behera utziko da.

    2. Gerta daiteke, ekintzen zerrendako zailtasun zehatz bateko erabiltzaileari aurkeztuko zaion ekintza jada aurkeztu izana. Kasu hau gertatzen bada, EkintzaErrepikatuaException salbuespena jaurtiko da. Bere tratamendua, zailtasun bereko beste ekintza bat bilatzean datza. Kontuan izan beste ekintza bat bilatzerako garaian berriz ere EkintzaErrepikatuaException salbuespena gerta daitekeela. Are gehiago, agian behin eta berriz gerta daiteke kasu hau, adibidez, zailtasun zehatz bateko ekintza bakarra badago zerrendan. Horretxegatik, 10 aldiz gauza bera gertatzen bada, pantailan mezu bat agertuko da eta azterketaren kudeaketa bertan behera utziko da.