Ariketa01

Idatzi zenbaki osoen ArrayList bat hasieratu eta horren gainean ondorengo eragiketak egiten dituen programa:

- Erabiltzaileak sartzen duen n zenbakiaren lehen 10 zatitzaileekin bete. Inplementatu funtzionalitate hau azpiprograma baten bidez. Zenbakiak ez baditu 10 zatitzaile atera mezu bat eta beste zenbaki bat eskatu.
- Erakutsi array-a. Inplementatu azpiprograma baten bidez.
- Ezabatu elementu guztiak.
- -100 eta 100 arteko 50 ausazko zenbakiekin bete. Inplementatu azpiprograma baten bidez.
- Erakutsi array-a.
- Ezabatu elementu guztiak.
- -20 eta 20 arteko 10 ausazko zenbaki ez errepikatuekin bete. Inplementatu azpiprograma baten bidez.
- Erakutsi array-a.

Ariketa02

Sortu **PuntuSistema** izeneko klase bat. Klase honek **Puntuen** (berreskuratu Objektuetara Bideratutako Programazioko 1. ariketa multzoko 5. ariketako kodea) ArrayList bat izango du atributu bezala. Klase honek bi eraikitzaile izango ditu. Eraikitzaile batek Puntuen ArrayList bat jasoko du parametro bezala. Besteak Puntu batzuk gehituko ditu ArrayListean (zuk aukeratu). Jarraian metodo hauek inplementatu:

- **erakutsi()**: Zerrendako elementuak bistaratzen ditu. Erabili *for-each* egitura bat iteratzeko.
- **alderantzizkatu():** Zerrenda berri bat itzuliko du, baina elementuak atzetik aurrera adierazita egongo dira.
- **bigarrenKoadrantean():** Bigarren koadrantean dagoen zerrendako lehen puntua itzuliko du (eta *null* ez badago). Erabili iteratzaile bat iteratzeko.
- **zeroZero():** (0,0) puntuaren lehen agerpenaren indizea itzuliko du. Ez badago mezu bat bistaratuko du eta -1 balioa itzuliko du. Ez erabili *equals* metodoa.
- **batBatDago():** (1,1) puntua zerrendan dagoen ala ez itzuliko du. *equals* metodoa berridatzi eta erabili.
- **ezabatuDistantzia10():** Zerrendan (0,0) puntutik 10eko distantzia baino gutxiagora dagoen lehen puntua ezabatzen du. Erabili indizeak iteratzeko.
- **ezabatu4Koadrantea():** 4. koadrantean dauden puntuak zerrendatik ezabatzen ditu. Erabili iteratzaile bat iteratzeko.

Sortu *main* metodo bat eta probatu inplementatutako metodoak.

Ariketa03

Zinema baten kudeaketarako sistema bat programatuko duzu. Honetarako ondorengo klaseak inplementatu beharko dituzu:

- **Pelikula**:
    - Atributuak: izenburua, ekoizpen urtea eta iraupena (minutuetan).
    - Eraikitzaile bakarra, atributu guztien balioak jasotzen ditu parametro bezala.
    - **toString** metodoa: atributu guztien balioak itzuliko ditu karaktere-kate batean.
    - **isEqual** metodoa: uneko pelikula parametro bezala pasatzen zaion pelikularen berdina den ala ez itzuliko du. Honetarako pelikulek izenburu berdina (ez desberdindu

maiuskula eta minuskulen artean), urte berdina eta iraupen berdina (5 minutuko marjinarekin) izan behar dute.

- **Zinema**. Klase honek zinema baten une jakin bateko egoera gordetzen du (une horretan ematen ari diren pelikulak barne).
    - Atributuak: izena, pelikulen zerrenda bat (ArrayList) eta okupatutako aretoak (array bat, zinemak dituen areto kopuruaren tamaina izango du eta posizio bakoitzean areto horretan ematen ari diren pelikula gordeko da edo *hutsik* agertuko da).
      Pelikulen zerrendak eta okupatutako aretoek zinemaren uneko egoera errepresentatzen dute. Izan hau kontutan hurrengo metodoek beharbada bi egitura hauek aldatuko dituztela.
    - Eraikitzaile bakarra izango du eta honek bakarrik izena eta areto kopurua jasoko ditu. Hasieran zinemak ez du pelikularik ezta okupatutako aretorik izango.
    - **pelikularenAretoa** metodoa: pelikula baten izena, urtea eta iraupena emanda zinemako zein aretotan ematen ari diren itzuliko du (aretoak 1etik hasten dira). Erabili inplementatu duzun *isEqual* metodoa eta pentsatu ematen ez badute metodo honek zer itzuliko duen.
    - **gehituPelikula** metodoa: pelikula bat emanda libre dagoen lehen aretoan kokatuko du.
    - **gehituPelikula** metodoa (bai, aurrekoaren izen berdina, metodoen gainkarga): pelikula bat eta areto bat jasota (gogoratu aretoak 1etik hasten direla) pelikula areto horretan kokatuko du.
    - Moldatu aurreko bi metodoak pelikula bat hurrengo baldintzetako bat ematen denean kokatu ezin izateko (mezu bat erakutsiko da dagokion arrazoiarekin):
        - Pelikula dagoeneko zineman badago.
        - Adierazi den aretoa okupatuta badago.
        - Areto librerik geratzen ez bada.
    - **erakutsi** metodoa: zinemaren izena eta ematen ari den pelikula guztien izenburu eta aretoak erakutsiko ditu pantailan aretoaren arabera ordenatuta. Adibidez, hau izan daiteke irteera posible bat:
      ```
      Trueba
      Django Unchained – Areto 2
      Les Misérables – Areto 3
      Argo – Areto 7
      ```
    - **ezabatuPelikula** metodoa: karaktere-kate bat jasota izenburua kate honekin hasten diren pelikula guztiak zinematik ezabatuko ditu.
    - **aretoLibreak** metodoa: libre dauden aretoen zenbakien zerrenda itzuliko du.
    - **aldatuAretoa** metodoa: pelikula bat jasota pelikula hori ematen ari den begiratuko du eta ematen ari badira aretoa erakutsiko du. Pelikula ez bada ematen mezu bat erakutsiko du. Pelikula existitzen bada eta aretoa itzuli bada, erabiltzaileari pelikula zein aretora mugitu nahi duen galdetuko dio eta emandako areto hau existitu eta libre dagoen konprobatuko du. Horrela bada pelikula esleituko du. Bestela, areto egoki bat sartu arte prozesua errepikatuko du.
    - **garbituZinema** metodoa: zinemako pelikula guztiak ezabatuko ditu.
    - **motzagoakBaino** metodoa: parametro bezala jasotzen duen iraupena (minututuetan) baino gutxiago irauten duten pelikulen izenen zerrenda bat itzuliko du.
    - **motzagoakBaino** metodoa: aurrekoaren berdina baina iraupena ordu eta minutuetan jasoko ditu.

- **Proba**. Klase honek *main* metodoa izango du eta metodo honetan inplementatutako metodo guztiak probatuko dira.

Ariketa04

Sortu **Agenda** izeneko klase bat. Klase honek bi atributu izango ditu, sarrerak izeneko HashMap bat eta agendak izan ditzakeen sarrera kopuru maximoa (konstantea). Mapak pertsona baten izena (gakoa) eta telefonoen array bat izango du (pertsona batek telefono bat baino gehiago izan dezake). Eraikitzaile bakarra izango du eta agendak izango duen sarrera kopuru maximoa jasoko du. Honetaz gain mapa hasieratuko du.
Honetaz gain hurrengo metodoak izango ditu:

- **gehituSarrera**: pertsona baten izena eta telefonoen array bat jasoko ditu eta agendara gehituko ditu. Dagoeneko agendan pertsona horrentzat telefonoak existitzen badira honako mezua erakutsiko zaio erabiltzaileari: “Zure agendan PERTSONA_IZENAren ondorengo zenbakiak gordeta daude: 1111, 2222, …, n. Telefono berriengatik aldatu nahi dituzu? Bai/Ez.” Erabiltzaileak emandako erantzunaren arabera gehituko dira ala ez. Honetaz aparte, agenda guztiz beteta badago “Agenda beteta dago” mezua erakutsiko da.
- **erakutsi**: agendaren edukia pantailan erakutsiko du, lerro bat sarrera bakoitzeko pertsonaren izen eta telefono guztiekin.
- **bilatuIzena**: pertsona baten izena emanda pertsona honen telefono guztiak itzuliko ditu karaktere-kate batean. Pertsonaren telefonorik ez badago “PERTSONA_IZENA ez dago agendan” mezua erakutsiko da.
- **zenbakiKopurua**: telefono zenbaki bat emanda zenbaki hori zenbat pertsonek duten itzuliko du. Gogoratu HashMap baten gakoak bakarrak direla, baina balioak bai errepikatu daitezkeela.
- **main**: Agendaren egitura eta aurreko metodo guztiak probatzeko ekintzak izango ditu.

Ariketa05

Kirol txapelketa baten kudeaketa sistema prestatzea eskatu dizute. Hau aurrera eramateko ondorengo klaseak definitu behar dituzu. Beharrezkoak ikusten dituzun gainerako klase eta metodoak inplementatu.

- **Kirolari**:
    - Atributuak: Izena, adina eta espezialitatea. Espezialitatea Enum klase bateko balioetako bat izango da, KORRIKALARI, TXIRRINDULARI edo IGERILARI.
    - Eraikitzaile bakarra izango du, honek atributu guztiak hasieratuko ditu.
- **Txapelketa**:
    - Atributuak: Kirolarien 3 multzo izango ditu. Multzo bakoitzak espezialitate bateko kirolariak bakarrik gordeko ditu.
    - Eraikitzailea bakarra izango da eta 3 multzoak hasieratuko ditu.
    - **gehituKirolaria** metodoa: Kirolari bat jasota espezialitatearen araberako multzoan gehituko du.
    - **aldatuEspezialitatea** metodoa: Kirolari bat eta espezialitate berri bat emanda kirolariaren espezialitatea aldatu, zegoen multzotik ezabatu eta dagokion multzo berrian gehituko du.
    - **erakutsiKirolariak** metodoa: Multzo guztietako kirolariak pantailan erakutsiko ditu, lehenik multzoaren espezialitatea adieraziz eta jarraian lerro bakoitzean kirolarien izena eta adina bakarrik erakutsiz.
    - **duatletak** metodoa: Bi espezialitatetan parte hartzen duten kirolarien multzoa erakutsiko du. Duatletarik ez badago horrela adieraziko da mezu baten bidez.

    - **triatletak** metodoa: Hiru espezialitatetan parte hartzen duten kirolarien multzoa erakutsiko du. Triatletarik ez badago horrela adieraziko da mezu baten bidez.
- **Main**. Txapelketa klasean dauden metodoak probatuko ditu, honakoa egingo du:
    - Sortu txapelketa bat.
    - Sortu hainbat kirolari eta gehitu dagokien multzoetan. Kontutan hartu kirolari bat multzo batean baino gehiagotan gehitu daitekeela lantzen dituen espezialitateen arabera.
    - Sailkatu kirolariak eta erakutsi hiru multzoetako korrikalari, txirrindulari eta igerilariak.
    - Aldatu bi kirolarien espezialitateak eta erakutsi hiru multzoetako elementuak berriro.
    - Erakutsi duatletak.
    - Erakutsi triatletak.

Ariketa06

Idatzi erabiltzaileari bi zenbaki oso eskatu eta bi zenbaki horien Zatitzaile Komun Handiena kalkulatzen duen programa. Erabili eta moldatu azpiprogramen ataleko ariketetan erabilitako kodea (3. unitateko [Azpiprogramak eta datu-egitura estatikoak] 1. ariketa multzoko 2. ariketa).

Ariketa07

Idatzi erabiltzaileari bi zenbaki oso eskatu eta bi zenbaki horien Multiplo Komun Txikiena kalkulatzen duen programa. Erabili eta moldatu azpiprogramen ataleko ariketetan erabilitako kodea (3. unitateko [Azpiprogramak eta datu-egitura estatikoak] 1. ariketa multzoko 2. ariketa).

Ariketa08

Idatzi burbuilaren ordenaketa metodoa (*Bubble sort*) erabiliz zerrenda bat ordenatzen duen eta ordenatzeko behar duen denbora kalkulatzen duen programa. Honetarako, lehenik definitu hiru azpiprograma:

- 100 ausazko zenbaki osooen zerrenda bat (1 eta 200 artekoak) sortu eta itzultzen duen azpiprograma.
- Zerrenda bat ordenatuta dagoen itzultzen duen azpiprograma.
- Zerrenda bat eta bi indize emanda, bi posizio horietan dauden elementuak trukatzen dituen azpiprograma. Honek ez du ezer itzuliko.

Ondoren, jarraitu ondorengo pausoak algoritmoa burutzeko:

- Zerrenda sortu eta erakutsi.
- Gorde aldagai batean uneko ordua *Instant.now()* metodoa erabiliz (*java.time* liburutegia).
- Zerrenda ordenatuta ez dagoen bitartean:
    - Zerrendaren bukaerara ailegatu arte:
        - Bi elementu hartu, konparatu eta lehena bigarrena baino handiagoa bada trukatu.
        - Bestela, indizea handitu hurrengo bi elementuak hartzeko.
    - Indizea hasieratu eta errepikatu.

Puntu honetan zerrenda ordenatuta egongo da. Kalkulatu berriro uneko ordua eta erabili *Duration.between()* iraupena kalkulatzeko eta erakutsi. Erakutsi ordenatutako zerrenda ere.

Funtzionatzen duenean, handitu elementu kopurua 5000 elementutara, egin zenbait exekuzio eta begiratu behar duen denbora.

Ariketa09

Idatzi kokteleraren ordenaketa metodoa (*Cocktail shaker sort*) erabiliz zerrenda bat ordenatzen duen eta ordenatzeko behar duen denbora kalkulatzen duen programa. Berrerabili aurreko ariketan definitutako hiru azpiprogramak.

- Ondoren, jarraitu ondorengo pausoak algoritmoa burutzeko:
- Zerrenda sortu eta erakutsi.
- Gorde aldagai batean uneko ordua *Instant.now()* metodoa erabiliz (*java.time* liburutegia).
    - Zerrenda ordenatuta ez dagoen bitartean:
    - Hasierako eta bukaerako posizioak gorde (hasieran 0 eta luzera – 1 izango dira)
    - Bukaerako posiziora ailegatu arte:
        - Bi elementu hartu, konparatu eta lehena bigarrena baino handiagoa bada trukatu.
        - Bestela, indizea handitu hurrengo bi elementuak hartzeko.
        - Bukaeran, elementu handiena bukaeran kokatuta egongo da.
    - Hasierako posiziora ailegatu arte:
        - Bi elementu hartu, konparatu eta lehena bigarrena baino txikiagoa bada trukatu.
        - Bestela, indizea txikitu hurrengo bi elementuak hartzeko.
        - Bukaeran, elementu txikiena hasierako kokatuta egongo da.
- Hasierako posizioa batean handitu eta bukaerakoa batean txikitu eta errepikatu.

Puntu honetan zerrenda ordenatuta egongo da. Kalkulatu berriro uneko ordua eta erabili *Duration.between()* iraupena kalkulatzeko eta erakutsi. Erakutsi ordenatutako zerrenda ere.

Funtzionatzen duenean, handitu elementu kopurua 5000 elementutara, egin zenbait exekuzio eta begiratu behar duen denbora. Aurreko algoritmoarekin konparatuta zein da azkarragoa?

Ariketa10

Idatzi nahasketa bidezko ordenaketa metodoa (*Merge sort*) erabiliz zerrenda bat ordenatzen duen eta ordenatzeko behar duen denbora kalkulatzen duen programa. Berrerabili aurreko ariketan definitutako hiru azpiprogramak.