DATU-EGITURAK ETA ALGORITMOAK I **2. MAILA**
2005ko ekainaren 27a

1 ariketa (3 puntu)

Elementu osoko esteka sinpleko zerrenda ordenatu batean, diseina eta inplementatu hurrengo metodoa LinkedList klasean:

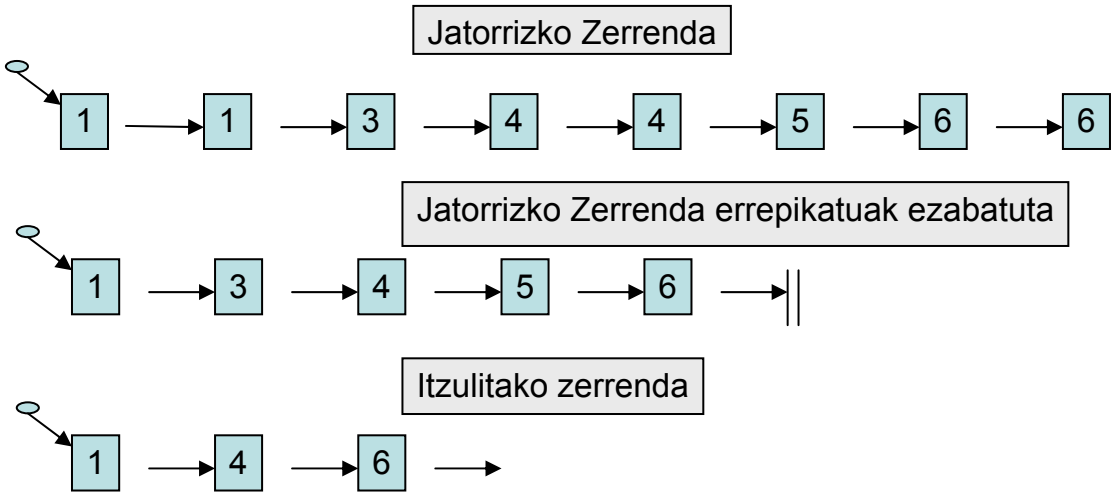
LinkedList errepikatuakEzabatu()

Metodoak hurrengoa egiten du:

1. Elementu errepikatuak ezabatzen ditu jatorrizko zerrendatik
2. Nodo errepikatu guztiak beste zerrenda batean itzultzen ditu.

Ohar: Ezin da inongo nodo berririk sortu, bigarrengo zerrendaren nodoak lehendabiziko zerrendaren nodoak izen behar dira.

Adibidea:




```
class Node<T> {
      T element;
      Node<T> next;
   }
public class LinkedList<T> {
      Node<T> top;
      Node<T> next;
}
```

2 ariketa (3 puntu)

Zuhaitz bitar bat **GorriBeltza (GB)** da, baldin eta soilik baldin honako baldintzak betetzen baditu:

1) **hosto** guztiak **beltzak** dira,
2) **gorria** den edozein adabegik **bi ume beltz** ditu,
3) edozein adabegi hartuta, bere ondorengo hosto batera doan **edozein bidek** adabegi **beltzen kopuru berdina** du.

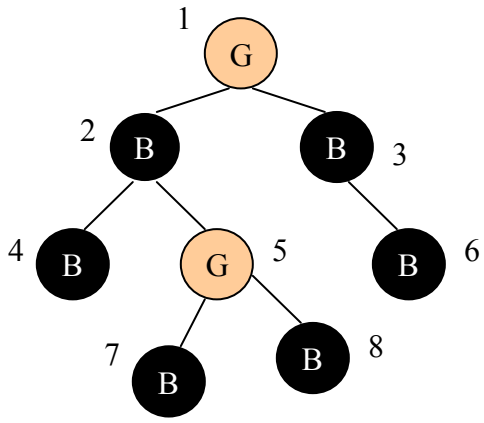
Espezifikatu, diseinatu eta inplementatu Java-n zuhaitz bitar iteratzaile klasean hurrengo metodoa:

boolean isGorriBeltza()

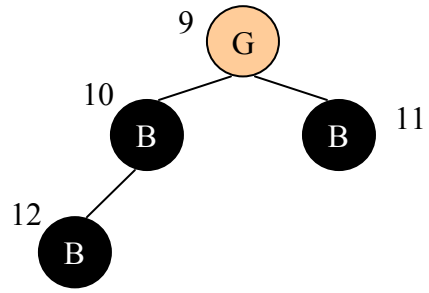
egiazkoa izango dena baldin eta soilik baldin zuhaitza gorribeltza bada.

OHARRAK:

1. *Node klasean boolean isBeltza() metodo boolearra erabil dezakezue. isBeltza() egiazkoa izango da baldin eta soilik baldin adabegia beltza bada.*
2. *Murgiltze-teknika erabili bide bateko adabegi beltzen kopurua zenbatzeko.*



GB da:
Adabegi guztiek hiru baldintzak betetzen dituzte.



EZ da GB
9 adabegiak ez baitu hirugarren baldintza betetzen:
9tik 12rako bideak **2** adabegi beltz ditu
9tik 11rako bideak abagegi beltz **1** du

Zuhaitzaren egitura hurrengoa da:

```
class BTNode<T> {
      T  content;
      BTNode<T> left;
      BTNode<T> right;
      boolean isBeltza()
   }
public class BilaketaZuhaitzBitarra<T> {
      BTNode<T> root;
      BTNode<T> actual;
}
```

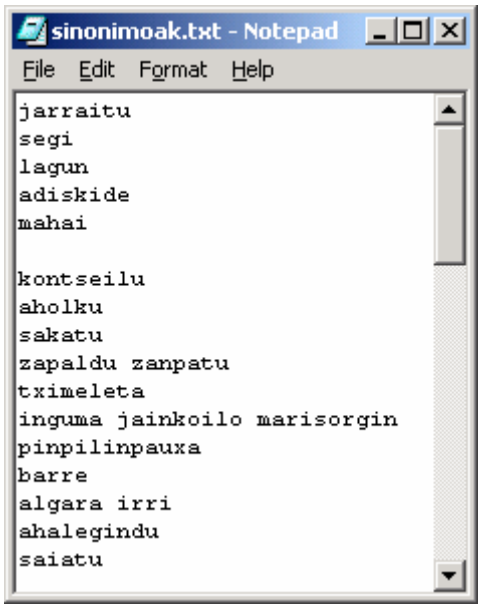
3. ariketa: (4 puntu)

Testu batean dauden hitz batzuen ordez beren sinonimoak jarri nahi dira. Horretarako Sinonimoak hiztegi klase bat sortzea erabaki da.

Sinonimoen hiztegia ("sinonimoak.txt") lerro-bikoteka dago antolatua:

- Lerro bakoitiak: hitz bakarra.
- Lerro bikoitiak: aurreko lerroko hitzaren sinonimoak, zuriune batez edo gehiagoz bereizita.

Gerta liteke hitz batek sinonimorik ez izatea; horrelakoetan lerro hutsa dator hitzaren ondoren (adibidez, *mahai* hitzak ez du sinonimorik). Testu-fitxategi hori Sinonimoak klaseak daukan hash taula batean kargatzen da. **Hash taulako gakoak lerro bakoitietako hitzak dira, eta gako horiei dagozkien datuak sinonimoen ilaretan gordetzen dira.**



Honakoa eskatzen da:

a) Espezifikatu, diseinatu eta inplementatu Javan **void taulaKargatu(String fitx)** metodoa Sinonimoak klasean. Metodo horrek sinonimoen hash taula osatzen du, fitxategi-izen bat ("sinonimoak.txt") emanda.

b) Espezifikatu, diseinatu eta inplementatu Javan **void aldatu(String fitx)** metodoa. Metodo horrek jatorrizko testu bat ("testua.txt") moldatzen du sinonimoen hash taula erabiliz, eta emaitza pantailatik aurkezten du. Moldaketa egiteko testuko hitzen ordez beren sinonimoak jartzen dira, honako arau hauek betez:

- Hitza ez badago hash taulan edo sinonimorik ez badauka, hitza bera idazten da irteera-fitxategian.
- Hitza hash taulan badago (gako moduan) eta sinonimoren bat badauka, sinonimoen ilarako hasierako elementua idatziko da pantailan. Sinonimo hori ez da berriz aukeratuko, harik eta hitz horren gainontzeko sinonimoak erabiltzen diren arte.

Adibidez:

testua.txt="barre egin lagun bat ikusi eta mahai batean eseri barre eginez"
pantailatik="algara egin adiskide bat ikusi eta mahai batean eseri irri eginez"

Bi ariketa horietan beharrezkoak diren klaseak erabili behar dira. Bereziki, fitxategiko hitzak tratatzeko, *FileReader* klasea erabili behar da.

FileReader klasea

```
class FileReader {
-- fitxategiaren lehenengo lerroan kokatzen da
public void FileReader(String fich);
-- true lerro gehiago geratzen badira, false beste kasuan
public boolean isLerroGehiago();
-- Vector objektu bat itzultzen du lerro horretako hitz guztiekin
public Vector getHitzak();
}
```

Queue<T> klasea

```
class Queue<T> {
boolean isEmpty();
-- pos: true ilara hutsa bada, eta false beste kasuan.

void enqueue(T elem);
-- pos: elem elementua txertatzen du ilaran. Ilara beteta badago, ez du
ezer egiten.

T dequeue ();
-- pos: ilararen lehen elementua itzuli eta ezabatzen du.

T front);
-- pos: ilararen lehen elementua itzultzen du. Hutsa bada null.
}
```

HashTable<K,V> klasea

```
class HashTable<K,V> {
void insert (K key, V value);
-- pos: key gakoa duen elementua value objektuarekin txertatzen du. Gakoa
existitzen bda, ez du ezer egiten.
void modify (K key, V value)
--pos: key gakoak daukan elementua aldatzen da value elementuarengatik. Gakoa
existitzen ez bada, ez du ezer egiten.
V find (K key);
-- pos: key dagokion objektua itzultzen du. Gakoa taulan ez badago, ez du ezer
egiten.
void remove (K key);
-- pos: key gakoa duen elementua ezabatzen du hash taulatik. Gakoa existitzen ez
bada, ez du ezer egiten.
boolean exist (K key);
-- pos: true itzultzen du bsb key gakoa existitzen bada.
int numPos ();
-- pos: hash taularen elementu kopurua itzultzen du.
}
```