**Zadatak 1:**

Napraviti klasu **Trajekt** koja ima:

* Atribut koji istovremeno predstavlja niz sedišta za putnike i mesta za automobile u trajektu.Sedište tj. mesto je slobodno ako je vrednost elementa niza TRUE a zauzeto ako je vrednost elementa niza FALSE. Sedišta za putnike se nalaze na početku niza, a njihov broj je dat u atributu ukupanBrojSedišta. Nakon poslednjeg sedišta, ostatak niza čine mesta za automobile
* Atribut ukupanBrojSedišta koji predstavlja ukupan broj sedišta u trajektu.
* Konstruktor koji kao ulazne argumente prima ukupan broj sedišta i ukupan broj mesta za automobile. Konstruktor inicijalizuje niz na kapacitet koji je jednak zbiru ova dva broja I postavlja vrednost atributa ukupanBrojSedišta na uneti broj sedišta. Ako je neki od unetih brojeva manji od nule, postaviti da trajekt ima 60 mesta za automobile, 60 sedišta i ispisati poruku o grešci. U svakom slučaju, postaviti da sva sedišta i mesta u trajektu budu slobodna.
* Metodu koja uvodi auto na mesto u trajektu. Metoda kao parametar prima redni broj mesta (uneti brojevi se kreću od 1 do ukupnog broja mesta za automobile). Obratiti pažnju na to da elementi niza koji predstavljaju mesta za automobile počinju tek posle svih sedišta za putnike. Ako traženo mesto nije slobodno, metoda proverava da li je naredno mesto slobodno i, ako jeste, samo ispisuje poruku o tome. Ako ni naredno mesto nije slobodno ili je traženo mesto na kraju niza, samo se ispisuje poruka o grešci
* Metodu koja proverava da li postoji dovoljno mesta na trajektu da se preveze autobus i vraća TRUE ako može a FALSE inače. Autobus može da se preveze ako negde na trajektu postoje četiri susedna mesta za automobile koja su sva slobodna.

***Moguće rešenje:***

class Trajekt{

boolean[] sedistaIMesta;

int ukupanBrojSedista;

Trajekt(int brojSedista, int brojMesta) {

if ( brojSedista < 0 || brojMesta < 0 ) {

ukupanBrojSedista = 60;

sedistaIMesta = new boolean[ 120 ];//60 sedista + 60 mesta = 120

System.out.println("Greska pri unosu broja sedista i/ili mesta za automobile");

}

else {

ukupanBrojSedista = brojSedista;

sedistaIMesta = new boolean[ brojSedista + brojMesta ];

}

//Na kraju postavljamo da sva sedista i mesta inicijalno budu slobodna

for (int i = 0; i < sedistaIMesta.length; i++) sedistaIMesta[i] = true;

}

void uvediAutoNaTrajekt(int brojMesta) {

//posto se (u nizu) mesta za automobile nalaze posle sedista za putnike,

//onda se pozicija na koju treba uvesti automobil dobija tako sto na broj sedista

//za putnike dodamo redni broj mesta na koji hocemo da uvedemo automobil i oduzmemo 1

if ( sedistaIMesta[ ukupanBrojSedista + brojMesta - 1 ] == true )

sedistaIMesta[ ukupanBrojSedista + brojMesta - 1 ] = false;

else

//Ako mesto nije na kraju niza i ako je naredno mesto slobodno, ispisuje se

//poruka o tome. U suprotnom se samo ispisuje poruka o gresci.

if ( ukupanBrojSedista + brojMesta - 1 < sedistaIMesta.length-1 &&

sedistaIMesta[ ukupanBrojSedista + brojMesta ] == true)

System.out.println("Mesto je vec zauzeto, ali je naredno mesto slobodno");

else

System.out.println("Greska! Mesto je vec zauzeto");

}

boolean imaLiMestaZaAutobus() {

//brojac i uzima vrednosti do sedistaIMesta.length - 3 zato sto pri svakom prolasku

//pored i-tog elementa, razmatramo i naredna 3 elementa

for (int i = ukupanBrojSedista; i < sedistaIMesta.length - 3; i++)

if ( sedistaIMesta[i] == true &&

sedistaIMesta[i+1] == true &&

sedistaIMesta[i+2] == true &&

sedistaIMesta[i+3] == true) return true; //nasli 4 susedna slob. mesta

//ako metoda pri izvrsavanju dodje do ove linije, znaci da nigde

//nisu pronadjena 4 uzastopna slobodna mesta

return false;

}

}

class TestTrajekt {

public static void main(String[] args) {

Trajekt t1 = new Trajekt(120, 20);

Trajekt t2 = new Trajekt(50, 50);

Trajekt t3 = new Trajekt(30, 0);

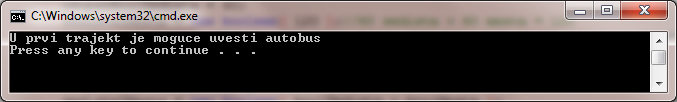
if ( t1.imaLiMestaZaAutobus() )System.out.println("U prvi trajekt je moguce uvesti autobus");

else System.out.println("U prvi trajekt NIJE moguce uvesti autobus");

}

}

***Ispis:***

******

**Zadatak 2.**

***Studenti rade sami:***

Napraviti klasu **Licnosti** koja ima:

* Statičku konstantu MAX\_BROJ\_UCESNIKA koja ima vrednost 20.
* Statičku metodu koja kao ulaz dobija znak i na ekranu ispisuje odgovarajući naziv tipa ličnosti. Ako se unese znak 'S', metoda ispisuje „Sangvinik“, za 'K' ispisuje „Kolerik“, za 'F' „Flegmatik“ i za 'M' „Melanholik“. Ako je unet neki drugi znak, metoda ispisuje na ekranu poruku o grešci.

Napraviti klasu **PsiholoskoTestiranje** koja ima:

* Atribut koji predstavlja niz oznaka tipova ličnosti osoba koje su podvrgnute testu. Dozvoljene vrednosti elemenata niza su 'S', 'K', 'F' i 'M'. Inicijalizovati niz na kapacitet jednak konstanti MAX\_BROJ\_UCESNIKA.
* Brojač koji predstavlja broj trenutno unetih oznaka ličnosti. Početna vrednost je nula
* Metodu za unos nove oznake tipa ličnosti u niz. Oznaka je data kao ulazni parametar a unos se vrši samo ako niz nije pun i ako je unet jedan od četiri dozvoljena znaka: 'S', 'K', 'F' i 'M'. Nova oznaka se unosi je na prvo slobodno mesto ali gledano od kraja niza ka početku (redom se popunjava poslednje mesto, pretposlednje itd. ka početku niza). Posle unosa, potrebno je ispisati naziv unetog tipa ličnosti (npr. „Kolerik“). U suprotnom, ispisuje se poruka o grešci
* Metodu koja na ekranu ispisuje sadržaj niza ali tako da se elementi sa početka i kraja niza ispisuju naizmenično – u cik-cak redosledu. To znači da se prvo ispiše prvi element niza, zatim poslednji, pa onda drugi element, zatim pretposlednji, pa treći element itd. Računati da je niz maksimalno popunjen tj. da nema praznih mesta.
* Napraviti klasu **TestPsiholoskoTestiranje** koja kreira jedan objekat klase PsiholoskoTestiranje i unosi u njega dva kolerika, tri sangvinika i jednog melanholika. Ispisati na ekranu sadržaj niza ali tako da se elementi sa početka i kraja niza ispisuju naizmenično – u cik-cak redosledu.

***Moguće rešenje:***

class Licnosti{

static final int MAX\_BROJ\_UCESNIKA=20;

static void prikaziTipLicnosti(char oznakaTipa) {

switch (oznakaTipa) {

case 'S': System.out.println("Sangvinik");break;

case 'K': System.out.println("Kolerik");break;

case 'F': System.out.println("Flegmatik");break;

case 'M': System.out.println("Melanholik");break;

default: System.out.println("Greska! Uneti znak ne odgovara ni jednom tipu licnosti");break;

}

}

}

class PsiholoskoTestiranje{

char[] oznakeLicnosti = new char[Licnosti.MAX\_BROJ\_UCESNIKA];

int brojUnetih = 0;

void unesiNovuOznaku(char oznakaLicnosti) {

//oznaka tipa licnosti se unosi ako je trenutni broj elemenata niza

//(brojUnetih) manji od maksimalnog broja (oznakeLicnosti.length) i

//ako je uneta oznaka tipa licnosti validna

if ( brojUnetih < oznakeLicnosti.length &&

(oznakaLicnosti == 'S' || oznakaLicnosti == 'F' || oznakaLicnosti == 'M' || oznakaLicnosti == 'K')) {

oznakeLicnosti[ oznakeLicnosti.length -1 -brojUnetih ] = oznakaLicnosti;

brojUnetih++;

//pozivamo static metodu iz klase Licnosti za prikaz unetog tipa licnosti

Licnosti.prikaziTipLicnosti(oznakaLicnosti);

}

else System.out.println("Greska! Nova oznaka licnosti nije evidentirana");

}

void prikaziElementeNizaNaizmenicno() {

for (int i = 0; i < (oznakeLicnosti.length-1)/2; i++) {

Licnosti.prikaziTipLicnosti( oznakeLicnosti[i] );

Licnosti.prikaziTipLicnosti( oznakeLicnosti[oznakeLicnosti.length-1-i] );

}

//ako je broj elemenata u nizu paran, svi elementi niza ce biti ispisani u prethodnoj

//for petlji; ako je broj elemenata niza neparan, element na samoj sredini niza nece

//biti ispisan;zato proveravamo da li je broj elemenata niza neparan, i ako jeste,

//ispisujemo 'sredisnji' element

if ( (oznakeLicnosti.length-1) % 2 != 0 )

Licnosti.prikaziTipLicnosti( oznakeLicnosti[ (oznakeLicnosti.length-1)/2+1 ] );

}

}

class TestPsiholoskoTestiranje {

public static void main(String[] args) {

PsiholoskoTestiranje testiranje = new PsiholoskoTestiranje();

testiranje.unesiNovuOznaku('K');

testiranje.unesiNovuOznaku('K');

testiranje.unesiNovuOznaku('S');

testiranje.unesiNovuOznaku('S');

testiranje.unesiNovuOznaku('S');

testiranje.unesiNovuOznaku('M');

testiranje.prikaziElementeNizaNaizmenicno();

}

}

***Ispis:***

