1. Tömbkezelés

Példa:

Hozzunk létre egy tetszőleges (konstans) méretű tömböt. Feltöltés után keressük meg a legnagyobb és legkisebb elemét, rendezzük le, majd a rendezett adatokat írjuk ki.

Megbeszélés:

1. Konstans létrehozása (final int MAX = 5;).
2. Tömb (nem dinamikus) létrehozása (int[] tomb = new int[MAX];):
   a. int[] tomb: deklaráció, a név lefoglalása,
   b. tomb = new int[MAX]: tárhely lefoglalása.
3. Maximum és minimum kereső algoritmus.
4. Rendező (buborékrendező) algoritmus.
5. Kiterjesztett for ciklus.

Megoldás:

```
package tömb_9;
import static java.lang.System.out;
import java.util.Scanner;
public class Tömb_9 {
	public static void main(String[] args) {
		final int MAX=5;
		Scanner billentyu = new Scanner (System.in);
		int[] tomb = new int[MAX];
//		Feltöltés
		for(int i=0; i<MAX; i++){
			out.printf("Kérem az %d. számot: ", i+1);
			tomb[i] = billentyu.nextInt();
		}
//		Minimum-maximum keresés
		int min, max;
		min = max = tomb[0];
		for(int i=1; i<MAX; i++){
			if(min > tomb[i])min = tomb[i];
			if(max < tomb[i])max = tomb[i];
		}
		out.println("A legkisebb érték = " + min);
		out.println("A legnagyobb érték = " + max);
//		Rendezés
		int csere;
		for(int j=0; j<MAX-1; j++)
			for(int i=0; i<MAX-j-1; i++)
				if(tomb[i]>tomb[i+1]){
					csere=tomb[i];
					tomb[i]=tomb[i+1];
					tomb[i+1]=csere;
		}
//		Kiírás kiterjesztett for ciklussal
		for(int i: tomb){
			out.println(i);
		}
}
```

2. Dinamikus tömb (ArrayList osztály)

Példa:

Tároljunk tetszőleges mennyiségű nevet egy dinamikus tömbben és próbáljunk ki néhány metódust.

Megbeszélés:

1. Az ArrayList osztály.
2. Elemek hozzáadása, eltávolítása.
3. next és nextLine metódusok.
4. Generikusok.

Megoldás:

```
package dinamikus_tömb;
import static java.lang.System.out;
import java.util.ArrayList;
import java.util.Scanner;
public class Dinamikus_tömb {
        public static void main(String[] args) {
                Scanner billentyu = new Scanner (System.in);
                ArrayList<String> nevek = new ArrayList<>();
                out.print("Kérek egy nevet: ");
                String nev = billentyu.nextLine();
                while(nev.length()!=0){
                        nevek.add(nev);
                        out.print("Kérek egy nevet: ");
                        nev = billentyu.nextLine();
                }
                out.println(nevek);
//Elemszám
                out.println("Az elemek száma = " + nevek.size());
//Tetszőleges elem kiírása
                out.print("Kérem a keresendő név indexét: ");
                int szam = billentyu.nextInt();
                out.println("A keresett név: " + nevek.get(szam));
//Elem módosítása
                out.print("Kérem a módosítandó név indexét: ");
                szam = billentyu.nextInt();
                out.println("A keresett név: " + nevek.get(szam));
                out.print("Kérem az új nevet: ");
                nev = billentyu.next();
                nevek.set(szam, nev);
                out.println(nevek);
//Elem törlése
                out.print("Kérek egy indexet: ");
                szam = billentyu.nextInt();
                out.println("A törlendő név: " + nevek.get(szam));
                nevek.remove(szam);
                out.println(nevek);
        }
}
```

3. A láncolt lista (LinkedList osztály)

Példa:

Tároljunk tetszőleges mennyiségű nevet egy láncolt listában.

Megbeszélés:

1. Az LinkedList osztály.
2. Elemek hozzáadása, eltávolítása.

Megoldás:

```
package láncolt_lista_1;
import static java.lang.System.out;
import java.util.LinkedList;
import java.util.Scanner;
public class Láncolt_lista_1 {
	public static void main(String[] args) {
		Scanner billentyu = new Scanner (System.in);
		LinkedList<String> nevek = new LinkedList<>();
		out.print("Kérek egy nevet: ");
		String nev = billentyu.nextLine();
		while(nev.length()!=0){
			nevek.add(nev);
			out.print("Kérek egy nevet: ");
			nev = billentyu.nextLine();
		}
		out.println(nevek);
		out.print("Kérek egy nevet a lista elejére: ");
		nev = billentyu.nextLine();
		nevek.addFirst(nev);
		out.print("Kérek egy nevet a lista végére: ");
		nev = billentyu.nextLine();
		nevek.addLast(nev);
		out.println(nevek);
		out.print("Kérem a beszurandó név indexét: ");
		int szam = billentyu.nextInt();
		out.print("Kérem a beszurandó nevet: ");
		nev = billentyu.next();
		nevek.add(szam, nev);
		out.println(nevek);
		out.print("Kérem a törlendő elem indexet: ");
		szam = billentyu.nextInt();
		out.println("A törlendő név: " + nevek.get(szam));
		nevek.remove(szam);
		out.println(nevek);
		out.print("Kérem a törlendő elemet: ");
		nev = billentyu.next();
		nevek.remove(nev);
		out.println(nevek);
	}
}
```

Példa:

Tároljunk tetszőleges mennyiségű nevet egy láncolt listában. Vizsgáljuk meg, hogy egy adott név megtalálható-e a listában. Módosítsunk egy tetszőleges indexű nevet.

Megoldás:

```
package láncolt_lista_2;
import static java.lang.System.out;
import java.util.LinkedList;
import java.util.Scanner;
public class Láncolt_Lista_2 {
	public static void main(String[] args) {
		Scanner billentyu = new Scanner (System.in);
		LinkedList<String> nevek = new LinkedList<>();
		out.print("Kérek egy nevet: ");
		String nev = billentyu.nextLine();
		while(nev.length()!=0){
			nevek.add(nev);
			out.print("Kérek egy nevet: ");
			nev = billentyu.nextLine();
		}
		out.println(nevek);
		out.println("A lista elemeinek száma = " + nevek.size());
//Elem keresése
		out.print("Kérek egy keresendő nevet: ");
		nev = billentyu.nextLine();
		if(nevek.contains(nev))out.printf("A %s név megtalálható!\n", nev);
		else out.printf("A %s név nem található meg!\n", nev);
		out.println(nevek);
//Adott elem módosítása
		out.print("Kérem a módosítandó elem indexet: ");
		int szam = billentyu.nextInt();
		out.print("Kérem az új nevet: ");
		nev = billentyu.next();
		nevek.set(szam, nev);
		out.println(nevek);
	}
}
```

4. Hallgatói feladatok:

Megbeszélés: Beszúrásos rendezés.

a. Az előző két példában megismertek alapján, tároljanak tetszőleges mennyiségű nevet rendezett láncolt listában.

b. Emeltszintű feladat: Hozzanak létre egy osztályt (objektumot) a következő tagváltozókkal:

int azon;
string név;
float fizetés;

A tagokat billentyűzetről olvassák be, majd az objektumot tárolják láncolt listában. A beolvasást és tárolást az *azon=0*-ig végezzék.

c. Kiemelt feladat: az előző feladatban megadottak, csak azonosítóra rendezve kell tárolni az adatokat.

5. Gyakorló feladatok:

A feladatok megoldásánál (a tömb feltöltésnél) úgy oldja meg a programot, hogy a felhasználó mindig tetszőleges számú elemmel tudja a tömböt vagy listát feltölteni (lp. valamilyen végjelig).

a. A program egy tetszőleges méretű (statikus) tömböt töltsön fel hőmérsékletekkel. Határozza meg, hogy a tömbben hány 0 és 100 fok közötti hőmérséklet található.
b. A program egy dinamikus (ArrayList) tömböt töltsön fel életkorokkal. Számolja meg, hogy hány 30-59 év közötti életkor található a tömbben.
c. A program egy tetszőleges méretű (statikus) tömböt töltsön fel osztályzatokkal. Számítsa ki a tömbben lévő osztályzatok átlagát.
d. A program egy tetszőleges méretű (statikus) tömböt töltsön fel egész számokkal, amelyek egy kocka oldalhosszúságát jelentik. Számítsa ki, a tömbben tárolt oldalhosszúságok alapján, a kockák térfogatát és felszínét. Ellenőrizze, hogy pozitív számokat adtunk-e meg.
e. A program egy dinamikus (ArrayList) tömböt töltsön fel hónapok sorszámaival. Feltöltés után írja ki, hogy a hónapok melyik évszakban találhatók („tavasz”, „nyár”, „ősz”, „tél”). Hibás adatmegadás esetén írjon hibaüzenatet!
f. A program egy láncolt listát (LinkedList) töltsön fel napok sorszámaival (1-től 7-ig). Írja ki – a tömbben tárolt adatok alapján – a napok nevét („hétfő”, „kedd”, stb.). Hibás adatmegadás esetén adjon hibajelzést!
g. A program egy dinamikus (ArrayList) tömböt töltsön fel számokkal. Határozza meg, hogy a tömbben hány páros és hány páratlan szám található.
h. A program egy láncolt listát (LinkedList) töltsön fel életkorokkal (maximum 100 éves korig). Majd életkoronként az alábbi szöveget jelenítse meg: ha az életkor 10, 20, 30, 40, 50, 60, 70, 80, 100: „Gratulálunk”; 1-29: „Fiatal”; 30-59: Középkorú”; 60-100: „Idős”. (Hibás adatmegadás esetén adjon hibajelzést!)
i. A program egy dinamikus (ArrayList) tömböt töltsön fel születési évekkel, majd olvassa be az aktuális évszámot. Határozza meg az életkorokat és azt, hogy hány 45 év feletti életkor van.
j. A program egy tetszőleges méretű (statikus) tömböt töltsön fel számokkal, ezek egy-egy kör sugarát jelentik. A tömbben tárolt adatok alapján számolja ki a körök területét és kerületét.
k. Olvasson be egy sztringbe (string) (legalább 3 karakterből álló) karaktersorozatot. Határozza meg és írja ki az első, utolsó és a középső elemet (páros számú karakterek esetén két középső elem van.
l. Olvasson be sztringet, majd vizsgáljuk meg, hogy visszafelé olvasva is ugyanaz-e a jelentése. Az eredményt szövegesen írja ki.
m. Olvasson be sztringet, majd számoljuk meg, hogy hány kisbetű karaktert adtunk meg.
n. Olvasson be sztringet, majd döntse el, hogy numerikus-e. (Numerikus, ha csak szám karaktereket tartalmaz!).