TEHNICI AVANSATE DE PROGRAMARE
LABORATORUL 4

1. Transmiterea parametrilor în Java:

În Java parametrii metodelor sunt transmiși prin valoare.

EXP 1: Testați ce va afișa următorul program:

```
class C{
	int x;
	C(int k){
		x = k;
	}
	public String toString(){
		return x + " ";
	}
}

public class TransmitereParametrii {
	static void interschimba1(C c1, C c2){
		C c3 = c1;
		c1 = c2;
		c2 = c3;
	}
	static void interschimba2(C c1, C c2){
		int c3 = c1.x;
		c1.x = c2.x;
		c2.x = c3;
	}
	public static void main(String args[]){
		C x= new C(1);
		C y= new C(2);
		interschimba1(x,y);
		System.out.print(x.toString() + y.toString());
		interschimba2(x,y);
		System.out.print(x.toString() + y.toString());
	}
}
```

2. Compararea obiectelor în Java, interfața Comparable

Interfața `java.lang.Comparable` este utilizată pentru compararea obiectului curent cu alte obiecte.

```
public interface Comparable<T>{
	int compareTo(T o)
}
```

`a.compareTo(b) < 0` =>> a este mai mic decât b
`a.compareTo(b) = 0` =>> a este egal decât b (se recomandă în acest caz și rescrierea metodei `equals`)
`a.compareTo(b) > 0` =>> a este mai mare decât b

EXP 2: Atunci când nu este folosit un tip generic, în metoda compareTo sunt necesare conversii.

```
public class Persoana implements Comparable{
	String nume;
	int varsta;
	Persoana(String nume, int varsta){
		this.nume=nume;
		this.varsta=varsta;
	}
	public int compareTo(Object persoana){
		if (persoana instanceof Persoana)
			return varsta-((Persoana)persoana).varsta;
		else throw new Error("nu se pot compara obiectele");
	}
	public String toString(){
		return nume+": "+varsta;
	}
}
```

EXP 3: Este recomandată utilizarea interfeței `Comparable<T>`

```
class PersoanaGeneric implements Comparable<PersoanaGeneric>{
	String nume, prenume;
	int varsta;
	PersoanaGeneric(String nume, String prenume, int varsta){
		this.nume = nume;
		this.prenume = prenume;
		this.varsta = varsta;
	}
	public int compareTo(PersoanaGeneric p){
		return varsta - p.varsta;
	}
	public static void main (String args[]){
		PersoanaGeneric p1=new PersoanaGeneric("a","b",20),
		p2=new PersoanaGeneric("c","d",18);
		if(p1.compareTo(p2)>0)
			System.out.println("p1 mai mare");
		else
			if(p1.compareTo(p2)==0)
				System.out.println("egale");
			else
				System.out.println("p2 mai mare");
	}
}
```

EXE 1: Modificați exemplul 3 astfel încât persoanele să fie comparate lexicografic.

3. Compararea obiectelor în Java, interfața Comparator

Interfața `java.util.Comparator` este utilizată pentru compararea a două obiecte de același tip.

```
public interface Comparator<T>{
	int compare(T o1, T o2); //compara obiectele o1 si o2
	boolean equals(Object o); //testeaza egalitatea cu un alt obiect de
				  //tip Comparator
}
```

EXP 4:
```
import java.util.Arrays;
import java.util.Comparator;

class StringComparator implements Comparator{
      public int compare(Object o1, Object o2){
            return ((String) o1).compareTo(((String) o2));
      }
}

class Sorter{
      public static void sort(Object[] data, Comparator comp){
            for (int i = data.length -1; i>=1; i--){
                  int indexOfMax = 0;
                  for (int j = 1; j <=i; j++){
                        if (comp.compare(data[j],data[indexOfMax]) > 0)
                                    indexOfMax = j;
                        }
                  Object temp  = data[i];
                  data[i] = data[indexOfMax];
                  data[indexOfMax] = temp;
            }
      }
}

public class TestSorter {
      public static void main(String args[]){
            String[] B = {"abc", "ade", "fg", "h", "Abc"};
            Comparator stringComp = new StringComparator();
            Sorter.sort(B, stringComp);
            Arrays.sort(B, new StringComparator());
            System.out.println(B[0]+" "+B[1]+" "+B[2]+" "+B[3]+" "+B[4]);
      }
}
```

EXE 2: Modificați exemplul 4 astefel încât să folosiți tipuri generice.
OBS: Metodele statice nu pot folosi tipul generic declarat în clasă. Se pot însă crea metode statice generice ca în exemplul de mai jos:

EXP 5: Tipul generic T poate fi folosit la tipul returnat, parametrii și în interiorul metodei.

```
import java.util.ArrayList;
import java.util.List;

public class MetodaGenerica {
      public static<T> List<T> toList(T[] vect){
            List<T> list = new ArrayList<T>();
            for(T element: vect)
                  list.add(element);
            return list;
      }
}
```

EXP 6: Acest exemplu folosește clase anonime.

```
public class TestSorter {
      public static void main(String args[]){
            Integer[] N = {5,4,3,2,1};
         Sorter.sort(N, new Comparator(){
            public int compare(Object o1, Object o2){
                  return (Integer) o1 - (Integer) o2;
                  }
            });
         Sorter.sort(N, new Comparator<Integer>(){
            public int compare(Integer o1, Integer o2){
                  return o1 - o2;
                  }
            });
         System.out.println(N[0]+" "+N[1]+" "+N[2]+" "+N[3]+" "+N[4]);
      }
}
```

EXE 3: Să se creeze o lista ordonată de obiecte. Pentru fiecare obiect se va reține numărul aparițiilor. Ordonarea se va face în funcție de greutate. În caz de egalitate ordonarea va se va face lexicografic în funcție de descirerea obiectelor.Modificați, după cum este necesar, clasele propuse spre utilizare:

```
class Obiect{
      int greutate;
      String descriere;
      Obiect(int greutate){
            this.greutate = greutate;
      }
}

class Lista<T extends Comparable<T>>{
      class Nod{
            T info;
            int nrAparitii;
      }
      Lista(){
      }
      Lista(T t){
      }
      void add(T t){
      }
      void remove(T p){
      }
      public String toString(){
      }
}
```

EXE 4: Să se creeze o agendă care va conține pentru fiecare persoană numele, prenumele, numărul de telefon, adresa de e-mail. Să se permită cautarea în lista după nume, număr de telefon, adăugarea sau ștergerea unei persoane, modificarea informațiilor.