SortedAssoTriple (вариант 1)

Проектирайте и реализирайте обобщен (generic) клас SortedAssoTriple за „подредена именувана тройка“. Трите елемента са снабдени с уникални имена и са **подредени в прав азбучен ред по имената** си. Създайте:

- конструктор по имена и стойности;
- сетър setValue на стойностите на елементите, определени по име **или** по номер (броенето започва от 0);
- предефинирайте метода toString, така че да връща низ във вида:
  {име_на_първи_елемент: стойност_на_първи_елемент; име_на_втори_елемент: стойност_на_втори_елемент; име_на_трети_елемент: стойност_на_трети_елемент}

Реализацията запишете в стандартен текстов файл с име Аномер_в_класа.txt, който да се намира в споделената папка.

Примерен тестващ клас

```
public class Test {
 public static void main(String[] args) {
  SortedAssoTriple<Integer> s;
  SortedAssoTriple<String> t;
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"One",3);
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Equal names not allowed");
  }
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"Three",3);
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Equal names not allowed");
  }
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"Next",3);
   s.setValue("One", 10);
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Invalid name");
  }
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"Next",3);
   s.setValue(4, 10);
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Invalid index");
  }
  try{
   t=new SortedAssoTriple<String>("One","first","Two","second","Next","third");
   System.out.println(t.toString());
   t.setValue(1, "Changed");
   System.out.println(t.toString());
  }catch(Exception e){
   System.out.println("Invalid index");
  }
 }
}
```

Резултати на стандартния изход:

```
Equal names not allowed
{One:1;Three:3;Two:2}
{Next:3;One:10;Two:2}
Invalid index
{Next:third;One:first;Two:second}
{Next:third;One:Changed;Two:second}
```

SortedAssoTriple (вариант 2)

Проектирайте и реализирайте обобщен (generic) клас SortedAssoTriple за „подредена именувана тройка“. Трите елемента са снабдени с уникални имена и са **подредени в обратен азбучен ред по имената** си. Създайте:

- конструктор по имена и стойности;
- сетър setName на името на елемент, определен по име **или** по номер (броенето започва от 0);
- предефинирайте метода toString, така че да връща низ във вида:
  {име_на_първи_елемент: стойност_на_първи_елемент; име_на_втори_елемент: стойност_на_втори_елемент; име_на_трети_елемент: стойност_на_трети_елемент}

Проекта запишете стандартен текстов файл с име Аномер_в_класа.txt, който запишете в споделената папка.

Примерен тестващ клас

```
public class Test {
 public static void main(String[] args) {
  SortedAssoTriple<Integer> s;
  SortedAssoTriple<String> t;
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"One",3);
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Equal names not allowed");
  }
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"Three",3);
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Equal names not allowed");
  }
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"Next",3);
   System.out.println(s.toString());
   s.setName(1, "NewName");
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Name already used");
  }
  try{
   s=new SortedAssoTriple<Integer>("One",1,"Two",2,"Next",3);
   s.setName("Two", "Next");
   System.out.println(s.toString());
  }catch(Exception e){
   System.out.println("Name already used");
  }
  try{
   t=new SortedAssoTriple<String>("One","first","Two","second","Next","third");
   System.out.println(t.toString());
   t.setName(1, "One");
   System.out.println(t.toString());
  }catch(Exception e){
   System.out.println("Name already used");
  }
 }
}
```

Резултати на стандартния изход:

```
Equal names not allowed
{Two:2;Three:3;One:1}
{Two:2;One:1;Next:3}
{Two:2;Next:3;NewName:1}
Name already used
{Two:second;One:first;Next:third}
{Two:second;One:first;Next:third}
```

Реализация

```
public class SortedAssoTriple<T> {
 public static final boolean REVERSE=true;//При true подредбата е в обратен азбучен ред
 //Частен клас {стойност, име}
 private class Elem{
  T value;
  String name;
  void swap(Elem e){//Размяна на два елемента
   //Стойностите
   T t=this.value;
   this.value=e.value;
   e.value=t;
   //Имената
   String n=this.name;
   this.name=e.name;
   e.name=n;
  }
 }
 //Свойства: три елемента
 private Elem e0=new Elem(),e1=new Elem(),e2=new Elem();
 //Подреждане в прав азбучен ред на имената
 private void sort(){
  if (e0.name.compareTo(e1.name)>0) e0.swap(e1);
  if (e1.name.compareTo(e2.name)>0) e1.swap(e2);
  if (e0.name.compareTo(e1.name)>0) e0.swap(e1);
 }
 //Подреждане в обратен азбучен ред на имената
 private void revSort(){
  if (e0.name.compareTo(e1.name)<0) e0.swap(e1);
  if (e1.name.compareTo(e2.name)<0) e1.swap(e2);
  if (e0.name.compareTo(e1.name)<0) e0.swap(e1);
 }
 //Номер по име
 private int getNoByName(String n){
  if (n.equals(e0.name)) return 0;
  if (n.equals(e1.name)) return 1;
  if (n.equals(e2.name)) return 2;
  return -1; //Ако не е намерен
 }
 //Конструктор
 public SortedAssoTriple (String n0,T v0,String n1, T v1,String n2,T v2) throws
IllegalArgumentException{
  //Не се допускат еднакви имена
  if (n0.equals(n1)||n0.equals(n2)||n1.equals(n2)) throw new IllegalArgumentException();
  else{
   e0.name=n0;
   e0.value=v0;
   e1.name=n1;
   e1.value=v1;
   e2.name=n2;
   e2.value=v2;
   //Подреждане (според REVERSE - растящо или намаляващо)
   if (REVERSE) revSort(); else sort();
  }
 }
 //Гетъри
 //На стойност по номер
 public T getValue(int n) throws IllegalArgumentException{
  switch(n){
   case 0: return e0.value;
   case 1: return e1.value;
   case 2: return e2.value;
   default: throw new IllegalArgumentException();
  }
 }
```

```
 //На стойност по име
 public T getValue(String n){
  int no=getNoByName(n);
  return getValue(no);
 }
 //На име по номер
 public String getName(int n) throws IllegalArgumentException{
  switch(n){
   case 0: return e0.name;
   case 1: return e1.name;
   case 2: return e2.name;
   default: throw new IllegalArgumentException();
  }
 }
 //Сетъри
 //На стойност по номер
 public void setValue(int n, T v) throws IllegalArgumentException{
  switch(n){
   case 0: {e0.value=v;break;}
   case 1: {e1.value=v;break;}
   case 2: {e2.value=v;break;}
   default: throw new IllegalArgumentException();
  }
 }
 //На стойност по име
 public void setValue(String n, T v){
  int no=getNoByName(n);
  setValue(no, v);
 }
 //На име по номер
 public void setName(int n, String newName)throws IllegalArgumentException{
  if (n<0||n>2) throw new IllegalArgumentException();
  int t=getNoByName(newName);
  //Ако това име вече го има и НЕ Е същото - недопустимо
  if (t>=0 && t!=n) throw new IllegalArgumentException();
  //Иначе - промяна
  switch(n){
   case 0:{e0.name=newName; break;}
   case 1:{e1.name=newName; break;}
   case 2:e2.name=newName;
  }
  //След промяна на името следва (евентуално) пренареждане
  if (REVERSE) revSort();else sort();
 }
 //На име по (старо) име
 public void setName(String nm, String newName)throws IllegalArgumentException{
  int n=getNoByName(nm);
  //Ако няма елемент с такова име
  if (n<0) throw new IllegalArgumentException();
  setName(n,newName);
 }
 //Създаване на низ
 public String toString(){
  return "{"+e0.name+":"+e0.value+";"+e1.name+":"+e1.value+";"+e2.name+":"+e2.value+"}";
 }
}
```