Przechowywanie obiektów w pamięci programu Część II (obiekty typów definiowanych przez użytkownika)

Obiekty typów definiowanych przez użytkownika muszą być przystosowane do przechowywania w pojemnikach – wtedy mogą być wyszukiwane, sortowane, usuwane itp. dzięki przedefiniowaniu metod *equals* oraz *hashCode,* dziedziczonych po klasie *Object* i implementacji metody *compareTo* z interfejsu *Comparable.*

```
class Klasa_uzytkownika   extends Object
                          implements
                             Comparable<Klasa_uzytkownika>
```

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

```
public class Object
 { //..............
```

boolean	**equals**(Object obj) Indicates whether some other object is "equal to" **this** one.
int	**hashCode**() Returns a hash code value **for** the object.

```
}
```

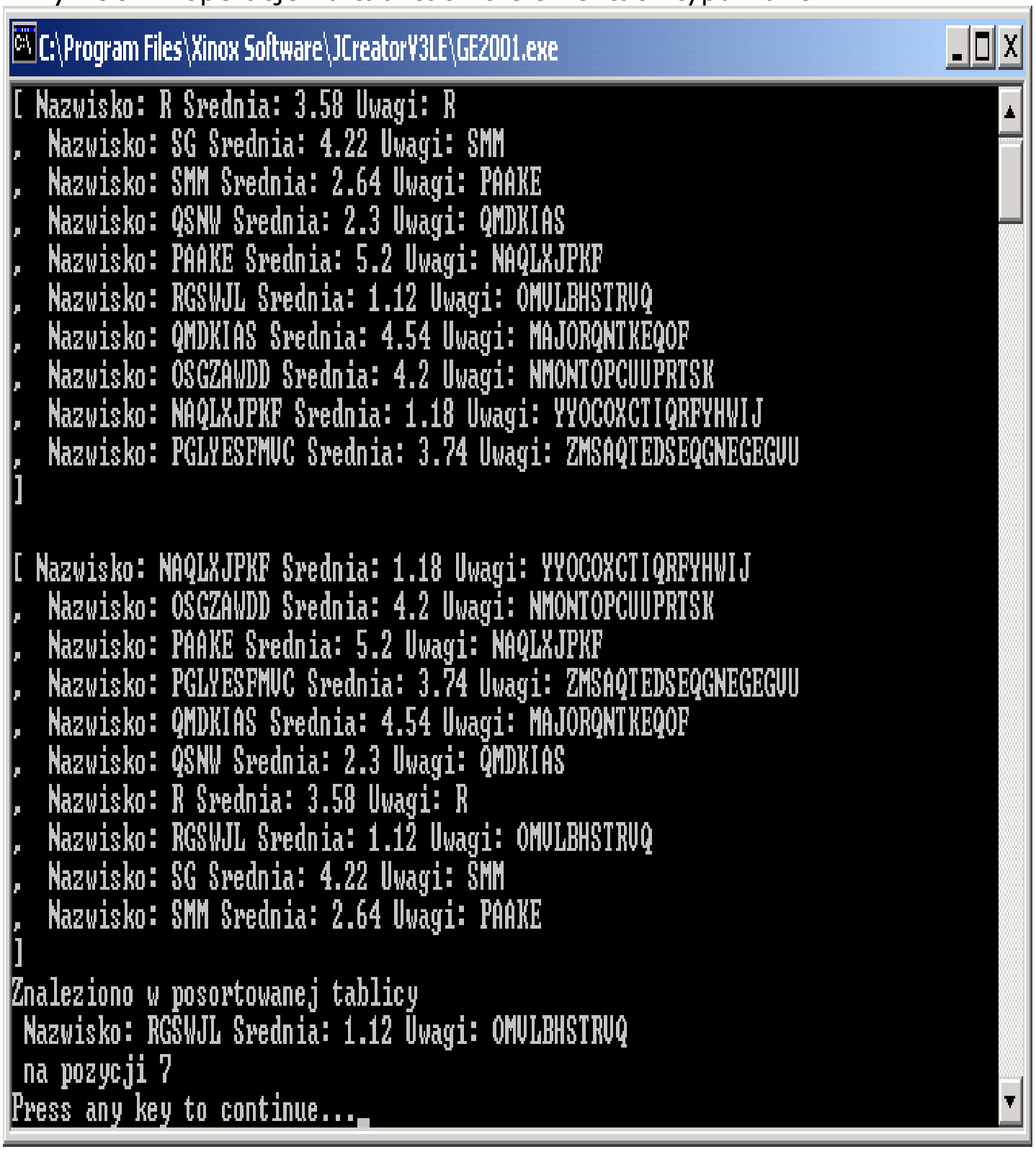
Klasa użytkownika musi mieć przedefiniowane metody *equals* oraz *hashCode* oraz zaimplementowaną metodę *compareTo.*

int	**compareTo**(Klasa_uzytkownika innyobiekt) Porównuje dwa obiekty wg. koncepcji programisty.
int	**hashCode**() Zwraca kod charakteryzujący dany typ użytkownika.
boolean	**equals**(Object anObject) Porównuje atrybuty obiektu wywołującego metodę z atrybutami obiektu przekazanego do metody.

2. Tablice

Klasa usługowa ***Arrays*** pozwala między innymi wyszukiwać i sortować tablice wypełnione typami obiektowymi i nie obiektowymi.

Przykład 1–operacje na tablicach o elementach typu *Dane1*



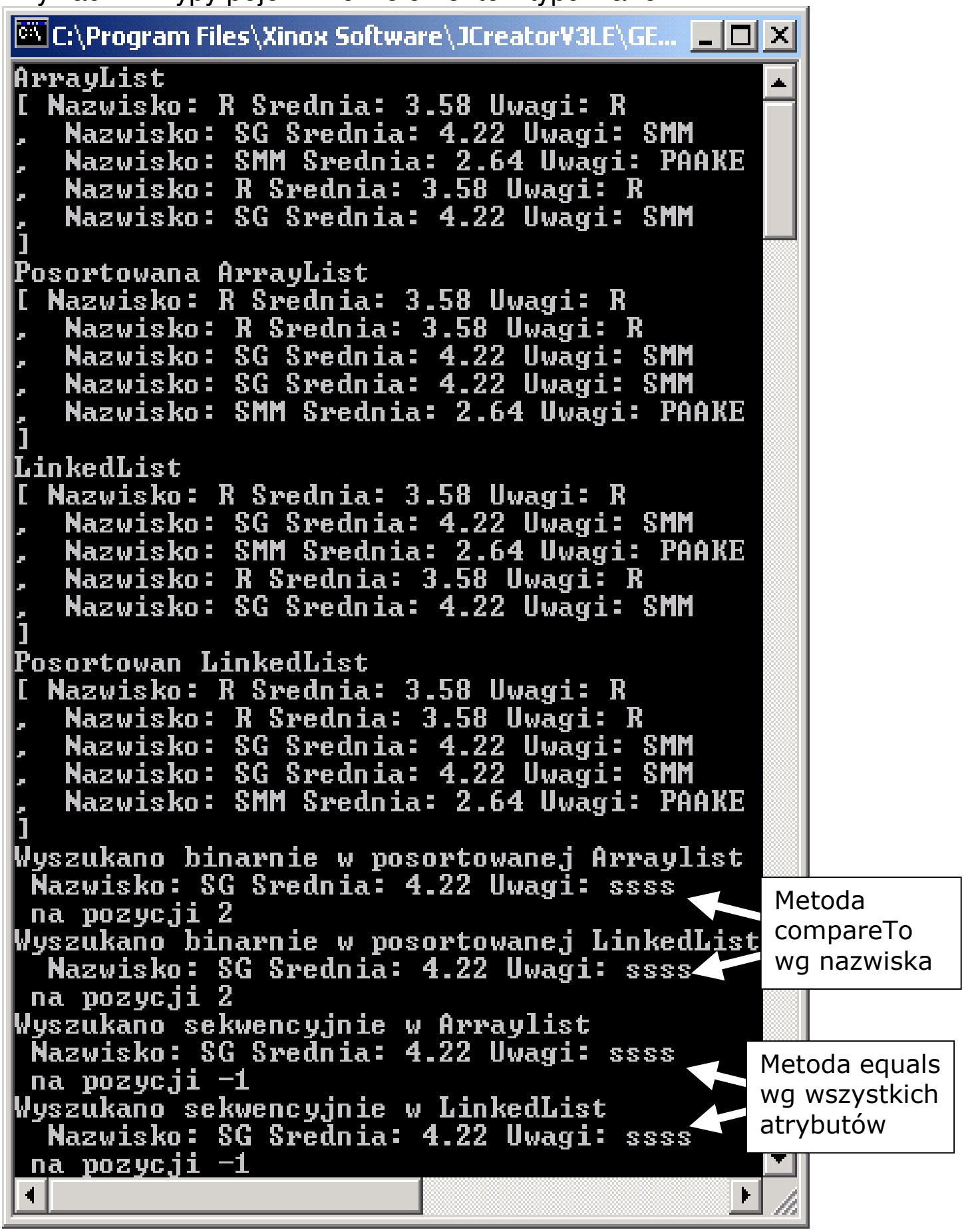
```
import javax.swing.*;
import java.util.*;

public class Dane1 implements Comparable <Dane1>
 { String nazwisko;
   float srednia;
   String uwagi;

public  void Nadaj_nazwisko(String lan)     { nazwisko=lan;}
public String Podaj_nazwisko()              { return nazwisko;}
public  void Nadaj_uwagi(String lan)        { uwagi=lan;}
public String Podaj_uwagi()                 { return uwagi;}
public  void Nadaj_srednia(float srednia_) { srednia=srednia_;}
public float Podaj_srednia()                { return srednia;}

public static Dane1 Wstaw(String nazwisko_,String uwagi_,String srednia_)
    { Dane1 d=new Dane1();
      d.Nadaj_nazwisko(nazwisko_);
      d.Nadaj_srednia(Float.parseFloat(srednia_));
      d.Nadaj_uwagi(uwagi_);
      return d; }

public String toString() //przesłonięta metoda toString z klasy Object
  {  String napis="";
     napis+=" Nazwisko: "+nazwisko;
     napis+=" Srednia: "+srednia;
     napis+=" Uwagi: "+uwagi+"\n";
     return napis; }

public boolean equals(Object o) //metoda umozliwiajaca przetwarzanie w pojemnikach typu Hash
  { Dane1 d=(Dane1)o;
    return  nazwisko.equals(d.nazwisko)
            && srednia==d.srednia
            && uwagi.equals(d.uwagi);}

public int hashCode()  //metoda umozliwiajaca przetwarzanie w pojemnikach typu Hash
 {return nazwisko.hashCode()+(int)srednia+uwagi.hashCode(); }

 public int compareTo(Dane1 o) //metoda umożliwiająca sortowanie, wyszukiwanie w
                               //pojemnikach List oraz przetwarzanie w pojemnikach typu TreeSet i TreeMap
  { return nazwisko.compareTo(o.nazwisko);  }
}
```

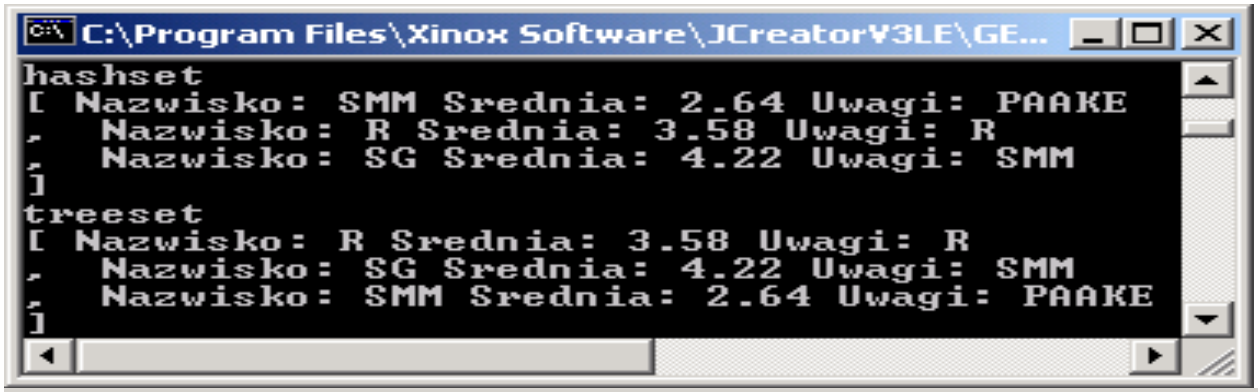
```
import java.lang.*;
import java.util.*;

public class Tablice1_1
 { static Dane1 tablica[];

   static public byte[] wypelnij(int n,int m,int offset,int zakres)
    { byte tablica1[]=new byte[n];
      Random r=new Random(m);
      for (int i=0; i<tablica1.length;i++)
        tablica1[i]=(byte)(offset+r.nextInt(zakres));
     return tablica1; }

   static Dane1 Wstaw(int i)
    {    String nazwisko=new String(wypelnij(i+1,i+1,65,26));
         String uwagi=new String(wypelnij(2*i+1,2*i+1,65,26));
         String srednia=new String(wypelnij(1,i+1,48,6))+"."
                        + new String(wypelnij(2,3*i+1,48,10));
         return Dane1.Wstaw(nazwisko,uwagi,srednia); }


  static public void wypelnij(int n)
   { tablica=new Dane1[n];
     for (int i=0; i<n;i++)
      { tablica[i]= Wstaw(i); }
    }

  public static void main(String args[])
   {
     wypelnij(10);
     System.out.println("\n"+Arrays.toString(tablica));
     Arrays.sort(tablica);
     System.out.println("\n"+Arrays.toString(tablica));
     Dane1 a=Wstaw(5);
     int b1=Arrays.binarySearch(tablica,a);
     System.out.println("Znaleziono w posortowanej tablicy\n"+a+
                              " na pozycji "+b1);
   }
 }
```

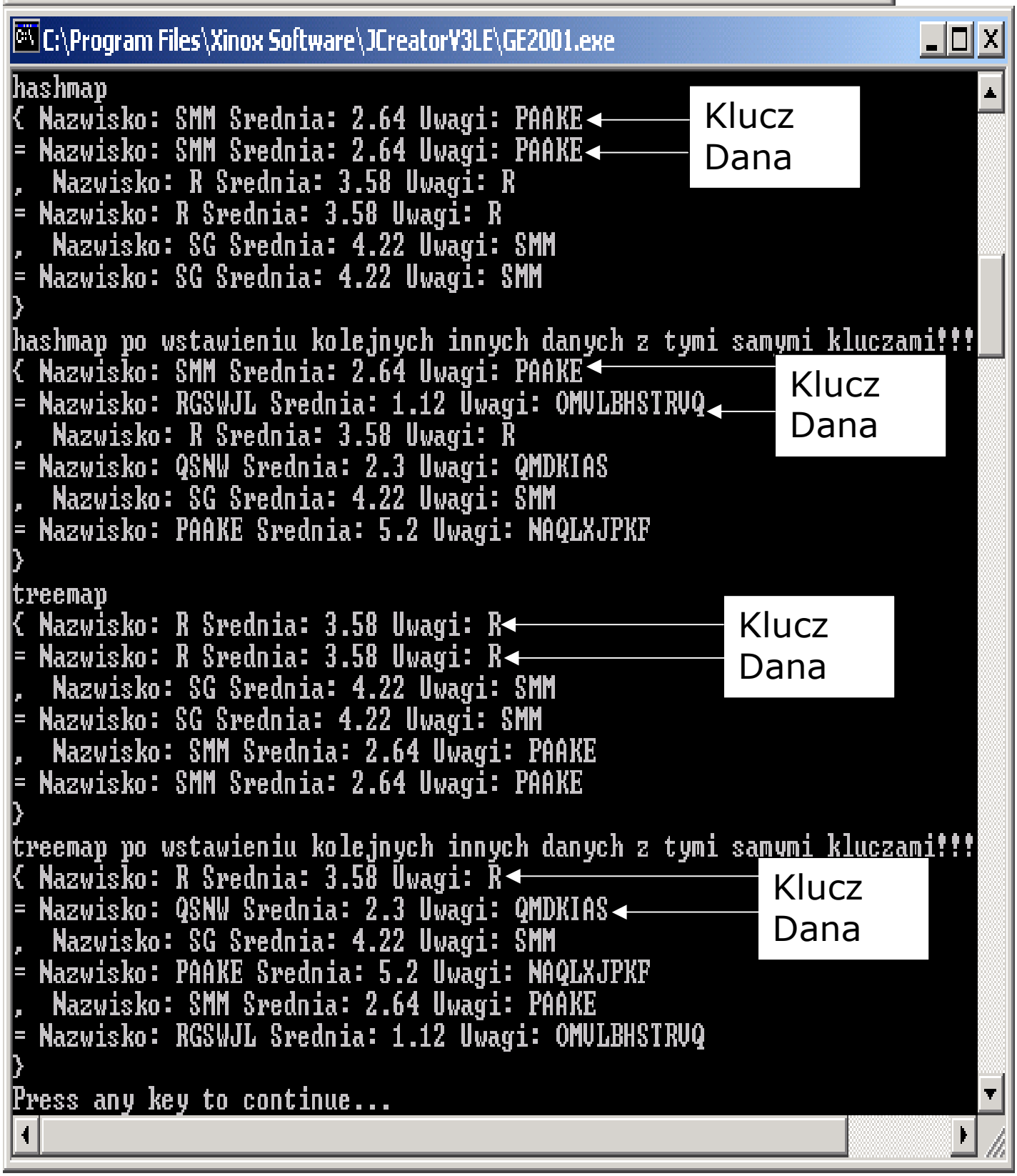
3. Pojemniki na obiekty

Przykłady zastosowania pojemników

Przykład 2 – Typy pojemników elementów typu *Dane1*




C:\Program Files\Xinox Software\JCreatorV3LE\GE...
hashset
[Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
, Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
]
treeset
[Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
, Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
]


C:\Program Files\Xinox Software\JCreatorV3LE\GE2001.exe
hashmap
{ Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
= Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
, Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: SG Srednia: 4.22 Uwagi: SMM
}
Klucz
Dana
hashmap po wstawieniu kolejnych innych danych z tymi samymi kluczami!!!
{ Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
= Nazwisko: RGSWJL Srednia: 1.12 Uwagi: OMVLBHSTRVQ
, Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: QSNW Srednia: 2.3 Uwagi: QMDKIAS
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: PAAKE Srednia: 5.2 Uwagi: NAQLXJPKF
}
Klucz
Dana
treemap
{ Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: SG Srednia: 4.22 Uwagi: SMM
, Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
= Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
}
Klucz
Dana
treemap po wstawieniu kolejnych innych danych z tymi samymi kluczami!!!
{ Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: QSNW Srednia: 2.3 Uwagi: QMDKIAS
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: PAAKE Srednia: 5.2 Uwagi: NAQLXJPKF
, Nazwisko: SMM Srednia: 2.64 Uwagi: PAAKE
= Nazwisko: RGSWJL Srednia: 1.12 Uwagi: OMVLBHSTRVQ
}
Klucz
Dana
Press any key to continue...

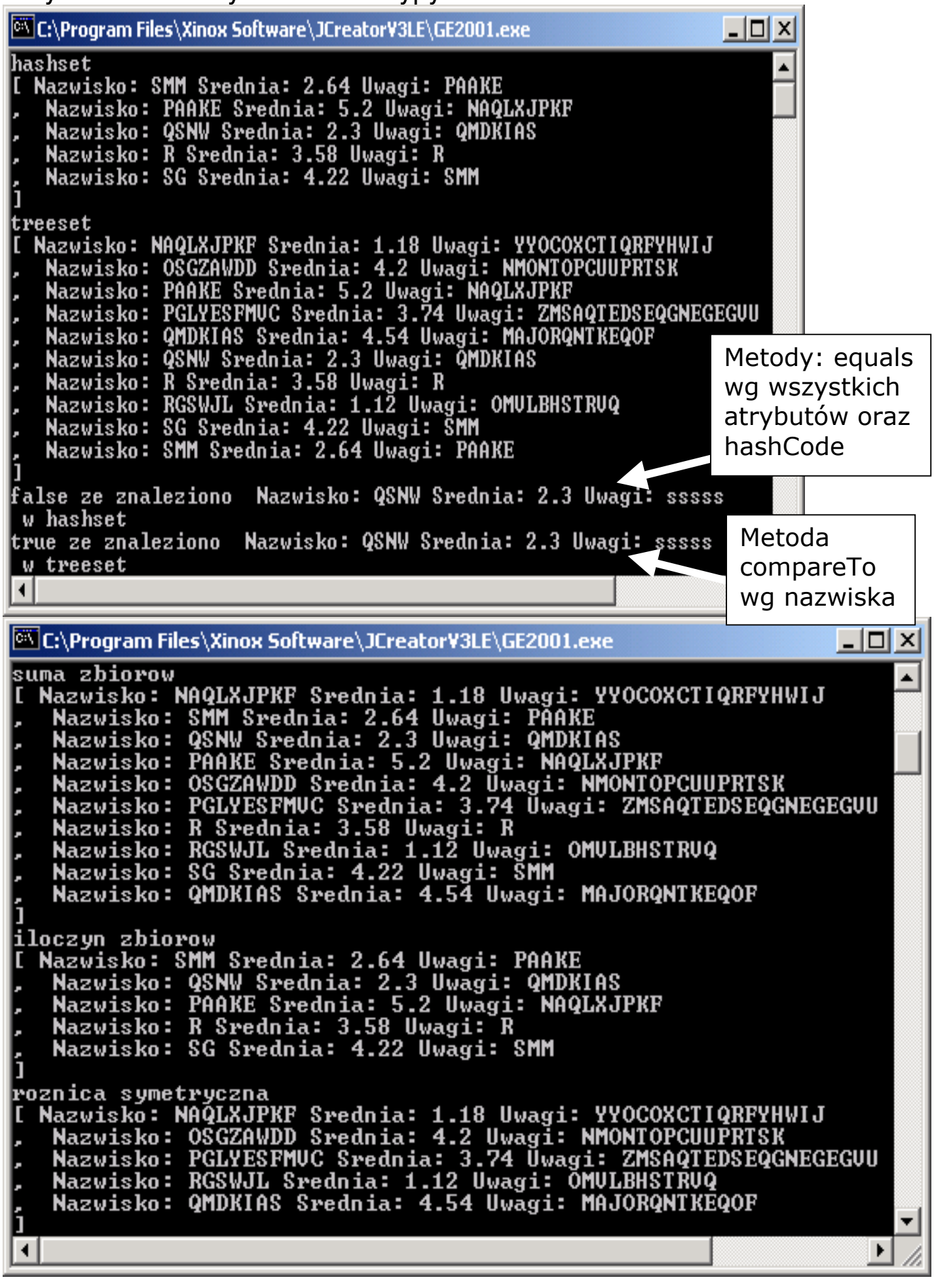
```
import javax.swing.*;
import java.util.*;

public class Dane1 implements Comparable <Dane1>
 { String nazwisko;
    float srednia;
    String uwagi;

   public  void Nadaj_nazwisko(String lan)        { nazwisko=lan;}
   public String Podaj_nazwisko()                 { return nazwisko;}
   public  void Nadaj_uwagi(String lan)                    { uwagi=lan;}
   public String Podaj_uwagi()                             { return uwagi;}
   public  void Nadaj_srednia(float srednia_)     { srednia=srednia_;}
   public float Podaj_srednia()                   { return srednia;}

   public static Dane1 Wstaw(String nazwisko_, String uwagi_,  String srednia_)
    {       Dane1 d=new Dane1();
            d.Nadaj_nazwisko(nazwisko_);
            d.Nadaj_srednia(Float.parseFloat(srednia_));
            d.Nadaj_uwagi(uwagi_);
            return d; }

  public String toString()
   {  String napis="";
      napis+=" Nazwisko: "+nazwisko;
      napis+=" Srednia: "+srednia;
      napis+=" Uwagi: "+uwagi+"\n";
      return napis; }

public boolean equals(Object o) //metoda umozliwiajaca wstawianie do pojemnikow typu Hash
   { Dane1 d=(Dane1)o;
     return  nazwisko.equals(d.nazwisko)
           && srednia==d.srednia
           && uwagi.equals(d.uwagi);  }

  public int hashCode()  //metoda umozliwiajaca wstawianie do pojemnikow typu Hash
    {   return nazwisko.hashCode()+(int)srednia+uwagi.hashCode();   }

  public int compareTo(Dane1 o) //metoda umożliwiająca sortowanie, wyszukiwanie w pojemnikach List
                                //oraz przetwarzanie w pojemnikach typu TreeSet i TreeMap
    { return nazwisko.compareTo(o.nazwisko);   }
 }
```

```
import java.lang.*;
import java.util.*;

public class Kolekcje2_1
{
static ArrayList <Dane1> arraylist= new ArrayList<Dane1>();
static LinkedList <Dane1> linkedlist= new LinkedList<Dane1>();
static HashSet <Dane1> hashset=new HashSet <Dane1>();
static TreeSet <Dane1> treeset=new TreeSet<Dane1>();
static HashMap<Dane1,Dane1>hashmap=new HashMap<Dane1,Dane1>();
static TreeMap <Dane1,Dane1> treemap=new TreeMap<Dane1,Dane1>();

 static public byte[] wypelnij(int n,int m,int offset,int zakres)
   { byte tablica1[]=new byte[n];
     Random r=new Random(m);
     for (int i=0; i<tablica1.length;i++)
        tablica1[i]=(byte)(offset+r.nextInt(zakres));
     return tablica1;
   }
 static Dane1 Wstaw(int i)
   {   String nazwisko=new String(wypelnij(i+1,i+1,65,26));
       String uwagi=new String(wypelnij(2*i+1,2*i+1,65,26));
       String srednia=new String(wypelnij(1,i+1,48,6))+"." + new String(wypelnij(2,3*i+1,48,10));
       return Dane1.Wstaw(nazwisko,uwagi,srednia);   }
```

```
static public void wypelnij1(int n, Collection<Dane1> kol)
   { for (int i=0; i<n;i++)  //można  przekazywać ArrayList, LinkedList, HashSet, TreeSet
     {  kol.add(Wstaw(i)); } }

static public void wypelnij2_1(int n, Map <Dane1,Dane1>mapa)
   { for (int i=0; i<n;i++) //można przekazywać TreeMap, HashMap
     { Dane1 dane=Wstaw(i);
       mapa.put(dane, dane);} }  //dane i klucz są identyczne

static public void wypelnij2_2(int n, Map <Dane1,Dane1>mapa)
   { for (int i=0; i<n;i++)
     { Dane1 dane1=Wstaw(i); // klucz
       Dane1 dane2=Wstaw(i+3); //dane  inne niż klucz
       mapa.put(dane1,dane2);} }

public static void main(String args[])
   { wypelnij1(3,arraylist);    wypelnij1(2,arraylist);
     System.out.println("ArrayList\n"+arraylist);
     Collections.sort(arraylist);
     System.out.println("Posortowana ArrayList\n"+arraylist);
     wypelnij1(3,linkedlist);     wypelnij1(2,linkedlist);
     System.out.println("LinkedList\n"+linkedlist);
     Collections.sort(linkedlist);
     System.out.println("Posortowan LinkedList\n"+linkedlist);

     Dane1 a=Wstaw(1);
     a.uwagi="ssss";
     int b1=Collections.binarySearch(arraylist,a); //metoda compareTo
     System.out.println("Wyszukano binarnie w posortowanej Arraylist\n"
                        +a+" na pozycji "+b1);
     int b2=Collections.binarySearch(linkedlist,a);   //metoda compareTo
     System.out.println("Wyszukano binarnie w posortowanej LinkedList\n "
                        +a+" na pozycji "+b2);
     b1=arraylist.indexOf(a); //metoda equals
     System.out.println("Wyszukano sekwencyjnie w Arraylist\n"
                        +a+" na pozycji "+b1);
     b2=linkedlist.indexOf(a); //metoda equals
     System.out.println("Wyszukano sekwencyjnie w LinkedList\n "
                       +a+" na pozycji "+b2);
```

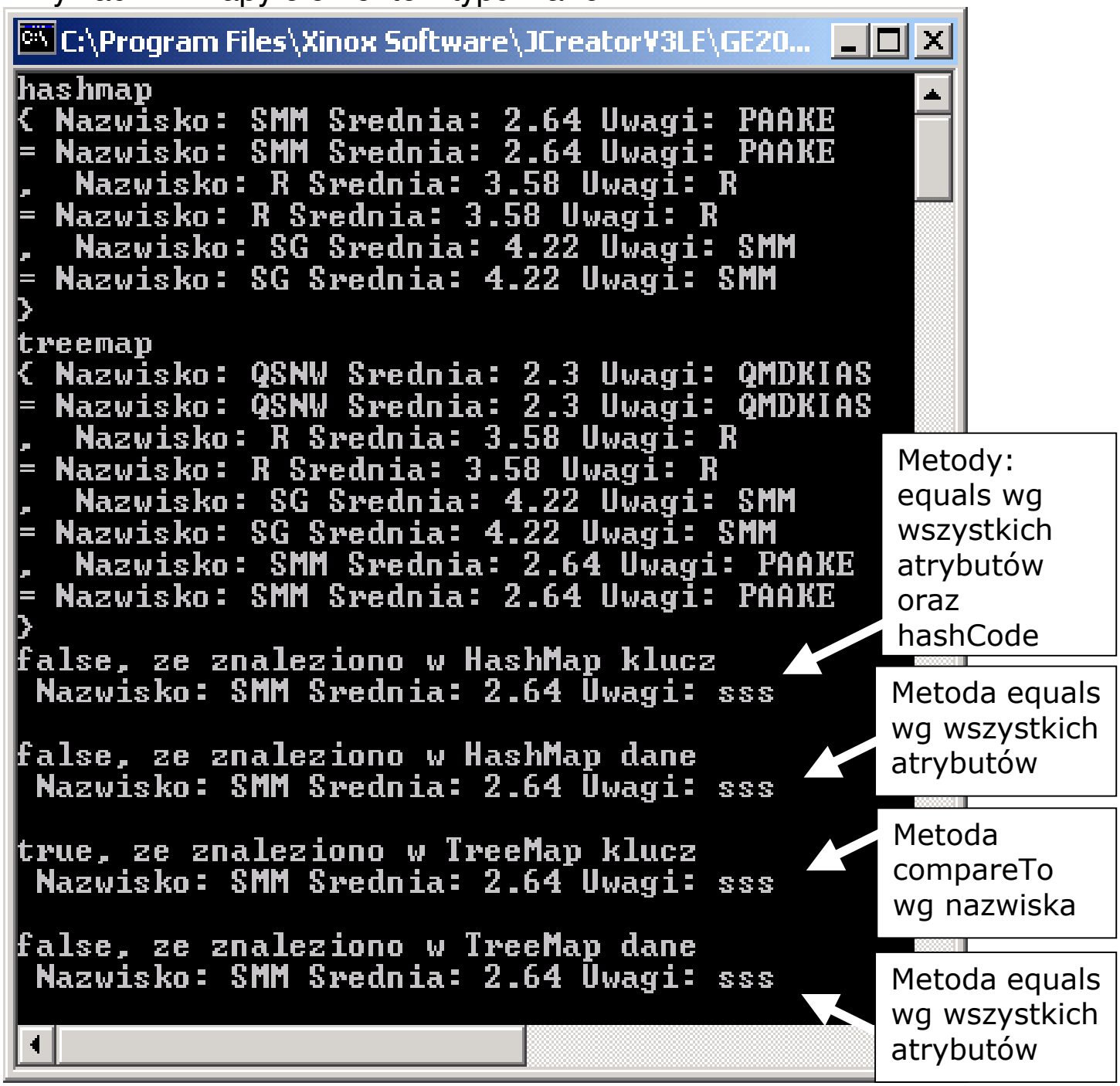
```
        wypelnij1(3,hashset);
        wypelnij1(3,hashset);
        System.out.println("hashset\n"+hashset);

        wypelnij1(3,treeset);
        wypelnij1(3,treeset);
        System.out.println("treeset\n"+treeset);

        wypelnij2_1(3,hashmap);
        wypelnij2_1(3,hashmap);
        System.out.println("hashmap\n"+hashmap);
        wypelnij2_2(3,hashmap);  //te same klucze, a dotychczasowe dane zastąpione nowymi danymi!!!
        System.out.println("hashmap po wstawieniu kolejnych innych danych"
                           +" z tymi samymi kluczami!!!\n"+hashmap);

        wypelnij2_1(3,treemap);
        wypelnij2_1(3,treemap);
        System.out.println("treemap\n"+treemap);
        wypelnij2_2(3,treemap); //te same klucze, a dotychczasowe dane zastąpione nowymi danymi!!!
        System.out.println("treemap po wstawieniu kolejnych innych danych"
                           +" z tymi samymi kluczami!!!\n"+treemap);
    }
}
```

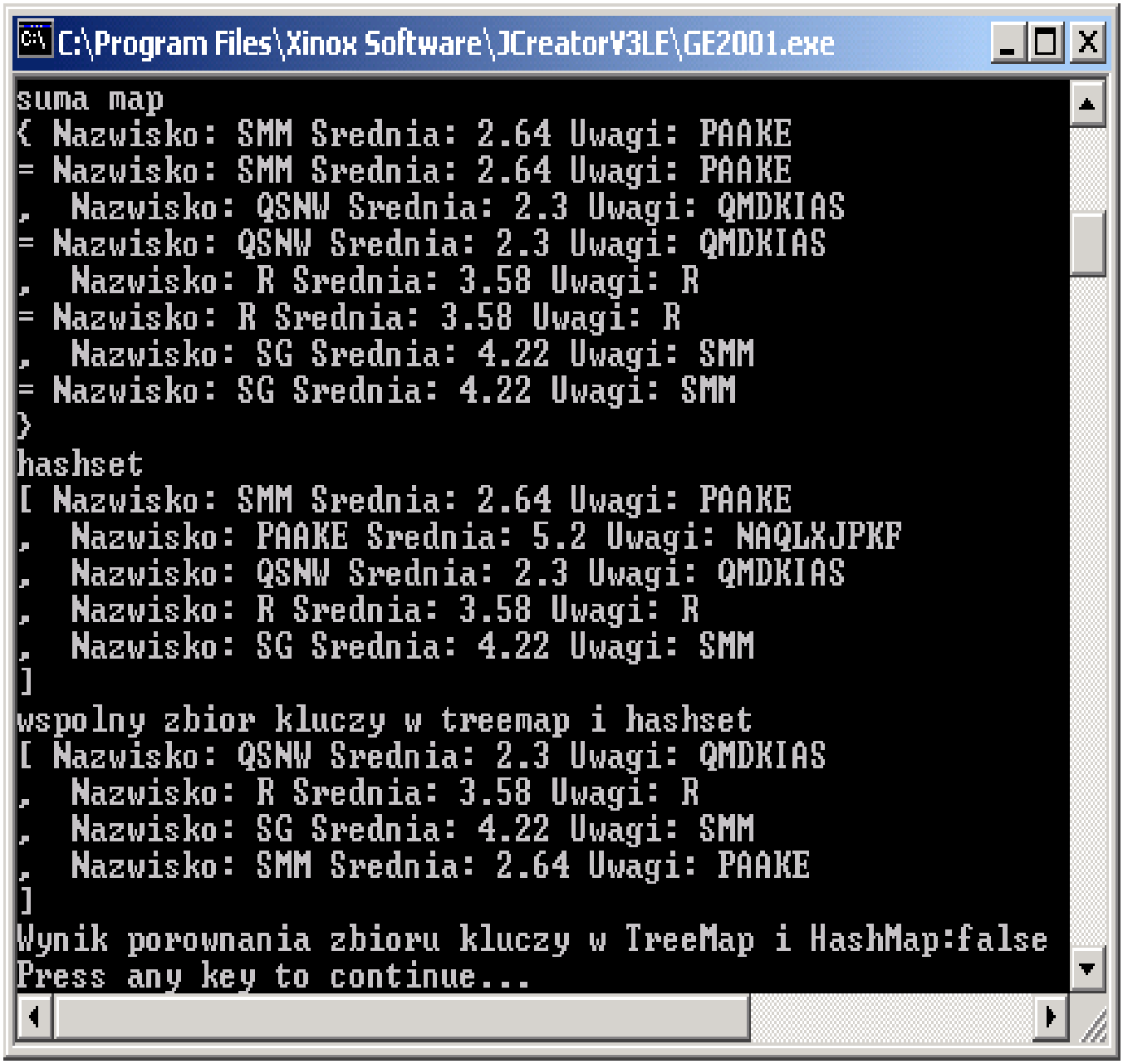
Przykład 3 – Zbiory elementów typy *Dane1*



```
import java.lang.*;
import java.util.*;

public class Zbiory1
 { static HashSet <Dane1> hashset=new HashSet <Dane1>();
   static TreeSet <Dane1> treeset=new TreeSet<Dane1>();

static public byte[] wypelnij(int n,int m,int offset,int zakres)
   { byte tablica1[]=new byte[n];
     Random r=new Random(m);
     for (int i=0; i<tablica1.length;i++)
       tablica1[i]=(byte)(offset+r.nextInt(zakres));
     return tablica1; }

  static Dane1 Wstaw(int i)
   { String nazwisko=new String(wypelnij(i+1,i+1,65,26));
     String uwagi=new String(wypelnij(2*i+1,2*i+1,65,26));
     String srednia=new String(wypelnij(1,i+1,48,6))+"."+ new String(wypelnij(2,3*i+1,48,10));
     return Dane1.Wstaw(nazwisko,uwagi,srednia);
    }

 static public void wypelnij (int n, Set<Dane1> kol)
   { for (int i=0; i<n;i++)
      { kol.add(Wstaw(i)); }
    }

static <K> void roznicasymetryczna(Set<K> set1, Set<K> set2)
    //uniwersalny nagłówek dla dowolnych typów elementów
   {  Set<K> roznicasym = new HashSet<K>(set1);
      roznicasym.addAll(set2);
      System.out.println("\nsuma zbiorow\n" + roznicasym.toString());
      Set<K> tmp =  new HashSet<K>(set1);
      tmp.retainAll(set2);
      System.out.println("\niloczyn zbiorow\n"+tmp.toString());
      roznicasym.removeAll(tmp);
      System.out.println("\nroznica symetryczna\n"+roznicasym.toString());
     }

 public static void main(String args[])
    { wypelnij(5,hashset);    wypelnij(5,hashset);
      wypelnij(10,treeset);    wypelnij(10,treeset);

      System.out.println("hashset\n"+hashset.toString());
      System.out.println("treeset\n"+treeset.toString());

      Dane1 klucz=Wstaw(5);
      klucz.uwagi="sssss";
      boolean b=hashset.contains(klucz); //metody equals i hashCode
      System.out.println(b+" ze znaleziono "+klucz+" w hashset");
      b=treeset.contains(klucz); //metoda compareTo
      System.out.println(b+" ze znaleziono "+klucz+" w treeset");
      roznicasymetryczna(hashset,treeset);  }
   }
```

Przykład 4 – Mapy elementów typu *Dane1*



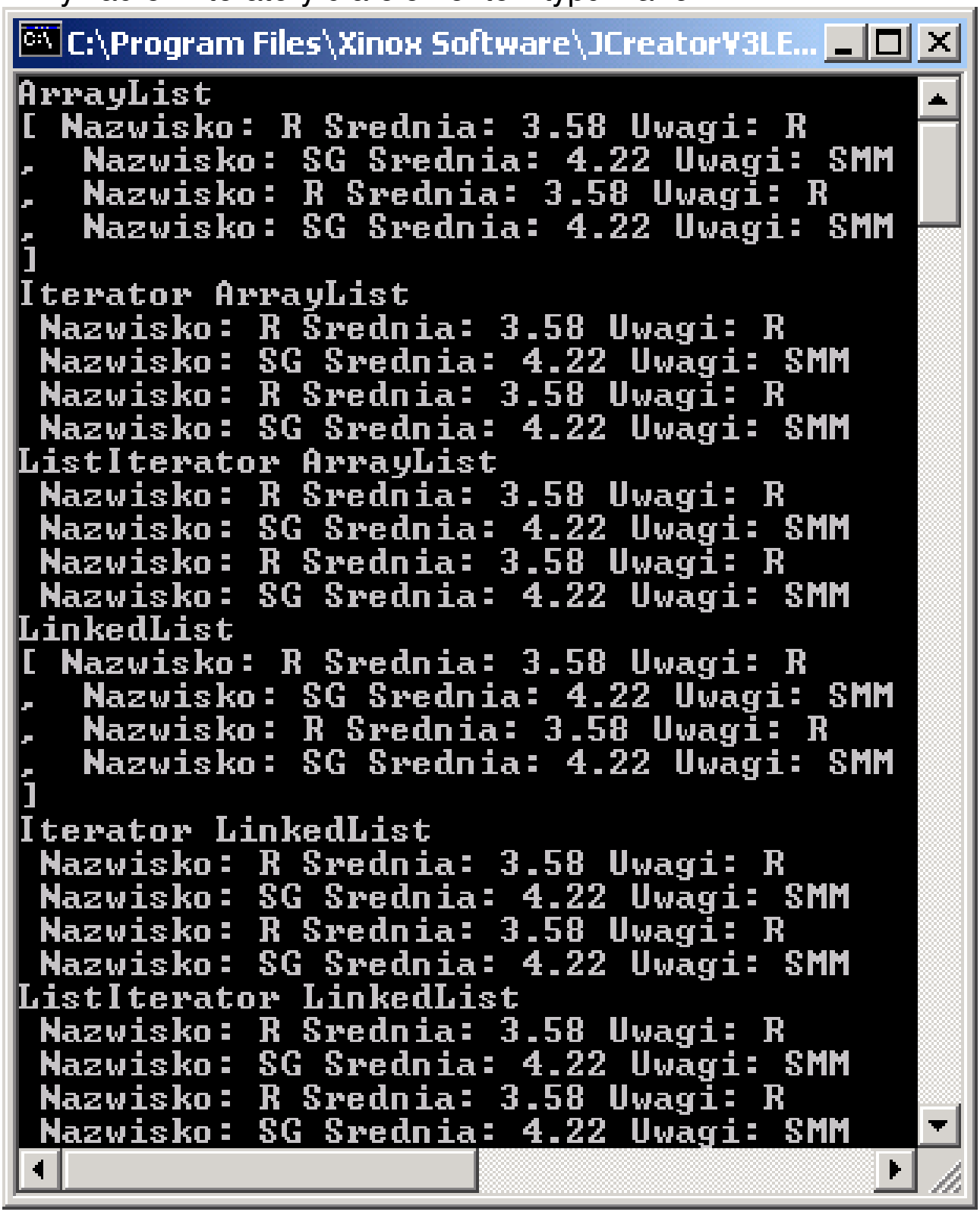


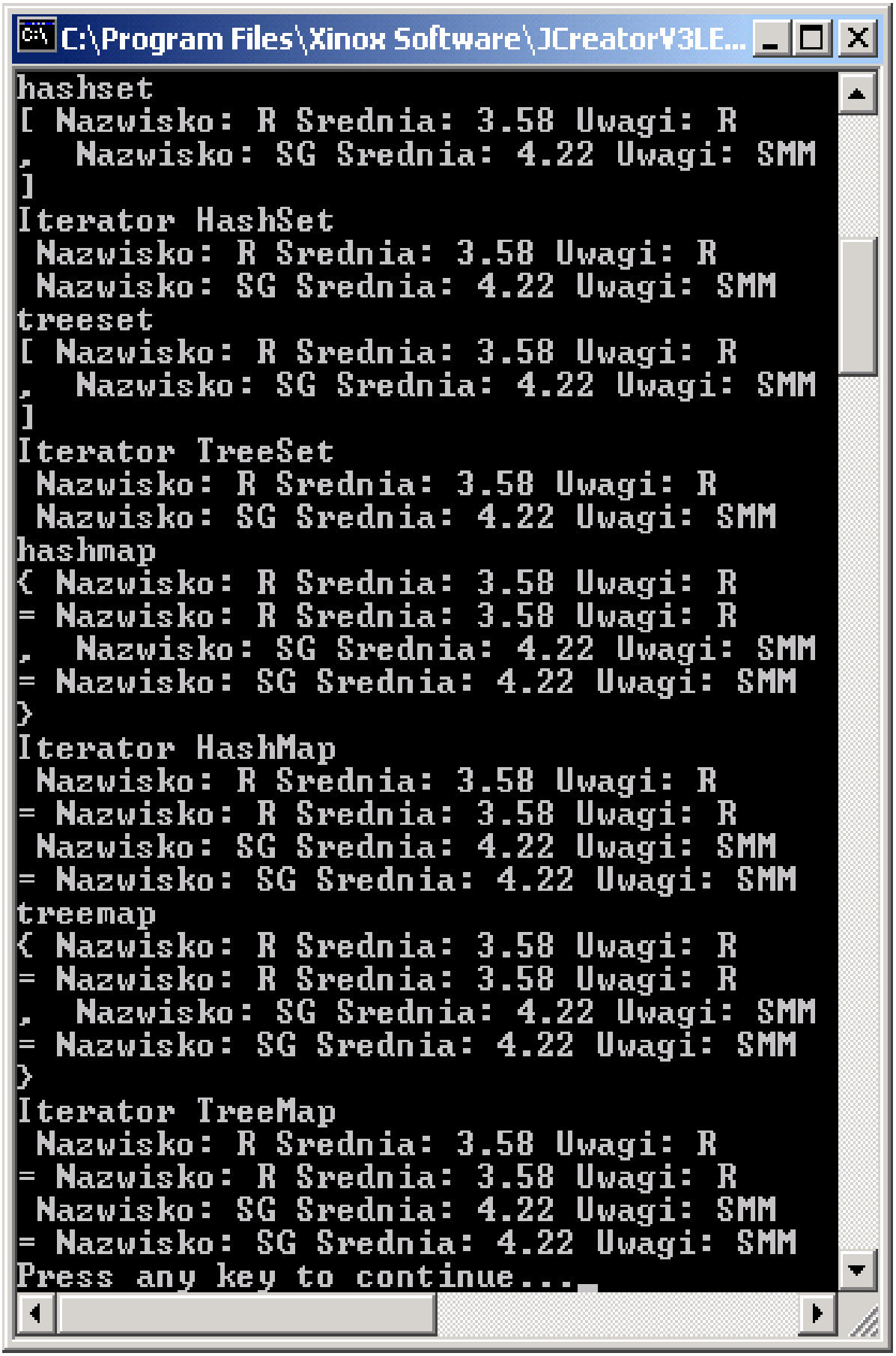

```
import java.lang.*;
import java.util.*;
public class Mapy1
{ static HashSet <Dane1> hashset=new HashSet <Dane1>();
  static HashMap<Dane1,Dane1>hashmap=new HashMap<Dane1,Dane1>();
  static TreeMap <Dane1,Dane1>treemap=new TreeMap<Dane1,Dane1>();

static public byte[] wypelnij(int n,int m,int offset,int zakres)
   { byte tablica1[]=new byte[n];
     Random r=new Random(m);
     for (int i=0; i<tablica1.length;i++)
       tablica1[i]=(byte)(offset+r.nextInt(zakres));
       return tablica1;
   }
 static Dane1 Wstaw(int i)
   {    String nazwisko=new String(wypelnij(i+1,i+1,65,26));
        String uwagi=new String(wypelnij(2*i+1,2*i+1,65,26));
        String srednia=new String(wypelnij(1,i+1,48,6))+"."  + new String(wypelnij(2,3*i+1,48,10));
        return Dane1.Wstaw(nazwisko,uwagi,srednia); }
  static public void wypelnij1(int n, Collection<Dane1> kol)
   { for (int i=0; i<n;i++)
     { kol.add(Wstaw(i)); } }
```

```
static public void wypelnij2(int n, Map <Dane1,Dane1>mapa)
   { for (int i=0; i<n;i++)
     { Dane1 dane=Wstaw(i);
       mapa.put(dane,dane);} }
```

```
 static <K,V> Map<K,V> sumamap(Map<K,V>pierwsza, Map<K,V>druga)
 {   Map<K, V> sumamap_ = new HashMap<K, V>(pierwsza);
      sumamap_.putAll(druga);
      return sumamap_;}

static <K,V>Set<K>  walidacja(Map<K,V> podstawowa,Set<K> wzorzec)
 {   Set<K> zle = new TreeSet<K>(wzorzec);
      Set<K> klucze = podstawowa.keySet();
      if(!klucze.containsAll(wzorzec))
         zle.retainAll(klucze);
      return zle;
  }
 public static void main(String args[])
    { wypelnij1(5,hashset);    wypelnij1(5,hashset);
      wypelnij2(3,hashmap);   wypelnij2(3,hashmap);
      wypelnij2(4,treemap);    wypelnij2(4,treemap);

      System.out.println("hashmap\n"+hashmap.toString());
      System.out.println("treemap\n"+treemap.toString());
      Dane1 a=Wstaw(2);
      a.uwagi="sss";
      boolean b1=hashmap.containsKey(a);//metody equals i hashCode
      System.out.println(b1+", ze znaleziono w HashMap klucz\n"+a);
      boolean b2=hashmap.containsValue(a); //metoda equals
      System.out.println(b2+", ze znaleziono w HashMap dane\n"+a);
      b1=treemap.containsKey(a); //metoda compareTo
      System.out.println(b1+", ze znaleziono w TreeMap klucz\n"+a);
      b2=treemap.containsValue(a); //metoda equals
      System.out.println(b2+", ze znaleziono w TreeMap dane\n"+a);
      Map<Dane1,Dane1>sumamap_=sumamap(treemap,hashmap);
      System.out.println("suma map\n"+sumamap_.toString());

      System.out.println("hashset\n"+hashset.toString());
      Set <Dane1>klucze_walidacji=walidacja(treemap,hashset);
      System.out.println("wspolny zbior kluczy w treemap i hashset\n"
                              +klucze_walidacji.toString());
      System.out.println("Wynik porownania zbioru kluczy w TreeMap i"
           +"HashMap:"+treemap.keySet().equals(hashmap.keySet())); } }
```

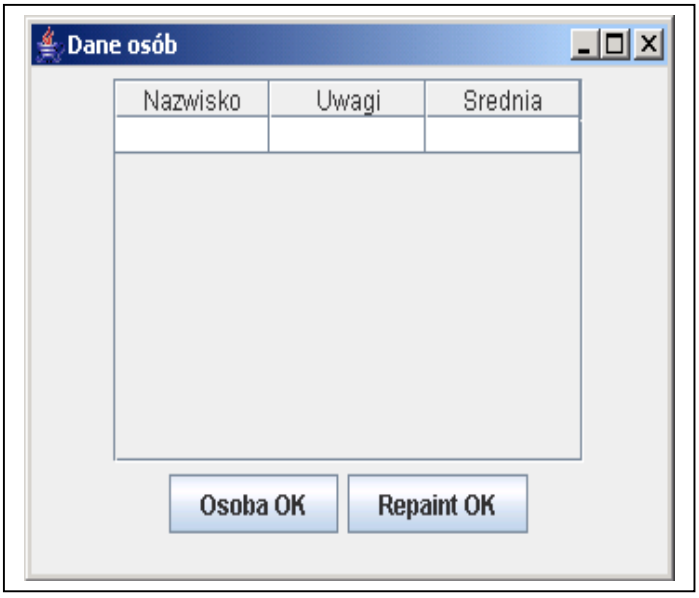
Przykład 5 – Iteratory dla elementów typu *Dane1*




C:\Program Files\Xinox Software\JCreatorV3LE...
hashset
[Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
]
Iterator HashSet
Nazwisko: R Srednia: 3.58 Uwagi: R
Nazwisko: SG Srednia: 4.22 Uwagi: SMM
treeset
[Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
]
Iterator TreeSet
Nazwisko: R Srednia: 3.58 Uwagi: R
Nazwisko: SG Srednia: 4.22 Uwagi: SMM
hashmap
{ Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: SG Srednia: 4.22 Uwagi: SMM
}
Iterator HashMap
Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: R Srednia: 3.58 Uwagi: R
Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: SG Srednia: 4.22 Uwagi: SMM
treemap
{ Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: R Srednia: 3.58 Uwagi: R
, Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: SG Srednia: 4.22 Uwagi: SMM
}
Iterator TreeMap
Nazwisko: R Srednia: 3.58 Uwagi: R
= Nazwisko: R Srednia: 3.58 Uwagi: R
Nazwisko: SG Srednia: 4.22 Uwagi: SMM
= Nazwisko: SG Srednia: 4.22 Uwagi: SMM
Press any key to continue...

```
import java.lang.*;
import java.util.*;

public class Kolekcje3_1
{ static ArrayList <Dane1> arraylist= new ArrayList<Dane1>();
  static LinkedList <Dane1> linkedlist= new LinkedList<Dane1>();
  static HashSet <Dane1> hashset=new HashSet <Dane1>();
  static TreeSet <Dane1> treeset=new TreeSet<Dane1>();
  static HashMap<Dane1,Dane1>hashmap=new HashMap<Dane1,Dane1>();
  static TreeMap <Dane1,Dane1> treemap=new TreeMap<Dane1,Dane1>();

static public byte[] wypelnij(int n,int m,int offset,int zakres)
  { byte tablica1[]=new byte[n];
    Random r=new Random(m);
    for (int i=0; i<tablica1.length;i++)
      tablica1[i]=(byte)(offset+r.nextInt(zakres));
      return tablica1;  }
  static Dane1 Wstaw(int i)
  {    String nazwisko=new String(wypelnij(i+1,i+1,65,26));
       String uwagi=new String(wypelnij(2*i+1,2*i+1,65,26));
       String srednia=new String(wypelnij(1,i+1,48,6))+"." + new String(wypelnij(2,3*i+1,48,10));
       return Dane1.Wstaw(nazwisko,uwagi,srednia);   }

 static  public void wypelnij1(int n, Collection<Dane1> kol)
 { for (int i=0; i<n;i++)
    { kol.add(Wstaw(i)); } }

static public void wypelnij2(int n, Map <Dane1,Dane1>mapa)
 { for (int i=0; i<n;i++)
    { Dane1 dana=Wstaw(i);
      mapa.put(dana,dana);} }
 static <K> void wyswietlIterator(String s, Iterator <K> it)
  { System.out.println(s);
    while(it.hasNext())
     { K k=it.next();
      System.out.print(k);}
  }
 static <K> void wyswietlIterator(String s, ListIterator <K> it)
  { System.out.println(s);
    while(it.hasNext())
     {  K k=it.next();
        System.out.print(k);}
  }

 public static void main(String args[])
  { wypelnij1(2,arraylist);  wypelnij1(2,arraylist);
    wypelnij1(2,linkedlist);  wypelnij1(2,linkedlist);
    wypelnij1(2,hashset);    wypelnij1(2,hashset);
    wypelnij1(2,treeset);     wypelnij1(2,treeset);
```

```
        wypelnij2(2,hashmap);  wypelnij2(2,hashmap);
        wypelnij2(2,treemap);    wypelnij2(2,treemap);
        System.out.println("ArrayList\n"+arraylist);
        wyswietlIterator("Iterator ArrayList",arraylist.iterator());
        wyswietlIterator("ListIterator ArrayList",arraylist.listIterator());

        System.out.println("LinkedList\n"+linkedlist);
        wyswietlIterator("Iterator LinkedList",linkedlist.iterator());
        wyswietlIterator("ListIterator LinkedList",linkedlist.listIterator());

        System.out.println("hashset\n"+hashset);
        wyswietlIterator("Iterator HashSet",hashset.iterator());

        System.out.println("treeset\n"+treeset);
        wyswietlIterator("Iterator TreeSet",treeset.iterator());

        System.out.println("hashmap\n"+hashmap);
         wyswietlIterator("Iterator HashMap",hashmap.entrySet().iterator());

        System.out.println("treemap\n"+treemap);
        wyswietlIterator("Iterator TreeMap",treemap.entrySet().iterator());
      }
}
```



Przykład programu interaktywnego

```
import java.awt.*;
import java.lang.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import java.awt.event.*;
import javax.swing.table.*;
```

```
class Panel extends JPanel implements ListSelectionListener
 { JTable tabela;
   DefaultTableModel model;
   Vector <Dane1> dane;
   int wiersz;

public Panel(Vector <Dane1> v)
 { super();
   dane=v;
   final String[] columnNames = {"Nazwisko","Uwagi","Srednia"};
   model=new DefaultTableModel(columnNames,0);
   tabela= new JTable(model);
   model.addRow(new Vector<String>(3)); //nowy rowek do wprowadzania danych w tabeli
   tabela.setPreferredScrollableViewportSize(new Dimension(250,150));
   tabela.setSelectionMode(ListSelectionModel.SINGLE_SELECTION );
   ListSelectionModel selekcja = tabela.getSelectionModel();
   selekcja.addListSelectionListener(this);
   JScrollPane suwak = new JScrollPane(tabela);
   add(suwak); }

  public void valueChanged(ListSelectionEvent e) //implementacja interfejsu
   { if (e.getValueIsAdjusting()) return; // ListSelectionListener do wyboru (podświetlania) wierszy tabeli
      ListSelectionModel lsm = (ListSelectionModel)e.getSource();
      if (!lsm.isSelectionEmpty())
        { wiersz = lsm.getMinSelectionIndex();
          System.out.println("Wiersz " + wiersz+ " jest wybrany.");  }   }

  public void wyswietl()
 {  Iterator iterator = dane.iterator();
    model.setRowCount(0); //czyszczenie zawartości tabeli
    int i=0;
    while(iterator.hasNext())
      { Dane1 t=(Dane1)iterator.next();
        Vector <String>vv=new Vector<String>(3);
        model.addRow(vv);
        model.setValueAt(t.Podaj_nazwisko(),i,0);
        model.setValueAt(t.Podaj_uwagi(),i,1);
        model.setValueAt(""+t.Podaj_srednia(),i,2);
        i++; }
    Vector <String>vv=new Vector<String>(3);
    model.addRow(vv);  } //nowy rowek do dalszej obsługi wprowadzania nowych danych
```

```
 void obslugaactionPerformed()
   { if(tabela.isRowSelected(wiersz))
       {  String nazwisko=(String)model.getValueAt(wiersz, 0); //pobranie z wybranych
          String uwagi=(String)model.getValueAt(wiersz, 1); // komórek łańcuchy
          String srednia=(String)model.getValueAt(wiersz, 2);
          if (nazwisko!=null&&uwagi!=null&&srednia!=null&&
             !nazwisko.equals("")&& !uwagi.equals("") &&!srednia.equals(""))
              { try
                {  Float.parseFloat(srednia);//kontrola poprawności formatu łańcucha srednia
                   Dane1 d=Dane1.Wstaw(nazwisko,uwagi, srednia);
                   dane.add(d);
                   Collections.sort(dane);
                   wyswietl();
                 } catch(NumberFormatException e)
                   { JOptionPane.showMessageDialog(this,
                                "Wprowadz poprawną średnią"); }
              }
        }
    }
}

class Okno extends JFrame implements ActionListener
 { Panel  panel;     //składniki interfesju graficznego użytkownika
   JButton osobaOK ;
   JButton repaintOK ;

 public Okno(Vector <Dane1>dane)
   { super("Dane osób");
     setSize(350,250);
     osobaOK=new JButton("Osoba OK");
     osobaOK.addActionListener(this);
     repaintOK=new JButton("Repaint OK");
     repaintOK.addActionListener(this);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     panel = new Panel(dane);
     panel.add(osobaOK);
     panel.add(repaintOK);
     setContentPane(panel);
     setVisible(true);}
```

```
public void actionPerformed(ActionEvent evt)
    { Object zrodlo=evt.getSource();
      if (zrodlo==osobaOK)
          panel.obslugaactionPerformed();
      else if (zrodlo==repaintOK)
         panel.wyswietl();
      repaint();    //wywołanie funkcji odświeżającej zawartość tabeli i pokazanie jej na ekranie
    } }

public class DaneOsob
  {  Vector<Dane1> dane=new Vector<Dane1>(5,5);
      static  public void main(String arg[])
        {  DaneOsob baza = new  DaneOsob();
           Okno okno = new Okno(baza.dane);  } }
```