collection

Zadanie.1

Napisać program, który:

a) tworzy listę (implementacja tablicy ArrayList), dodając po jednym elemencie (korzystając z operacji podstawowej add). Następnie wypisuje całą listę, drugi i zerowy element. Podaje rozmiar listy.

b) tworzy listę (implementacja listy łączonej LinkedList), dodając po jednym elemencie (korzystając z operacji podstawowej add). Następnie wypisuje całą listę, usuwa dwa ostatnie elementy i wypisuje listę po usunięciu tych elementów.

```
import java.util.*;
  public class ListExample {
    public static void main(String args[]) {
      List<String> lista1= new ArrayList<String> ();
      lista1.add("Ala");
      lista1.add("Ola");
      lista1.add("Ewa");
      lista1.add("Ola");
      lista1.add("Zosia");
      System.out.println(lista1);
      System.out.println("2: " + lista1.get(2));
      System.out.println("0: " + lista1.get(0));
      System.out.println :("lista ma " + lista1.size()+ "elementow" );
      List <String> lista2 = new LinkedList() <String>;
      lista2.add("Ala");
      lista2.add("Ola");
      lista2.add("Ewa");
      lista2.add("Ola");
      lista2.add("Zosia");
      System.out.println(lista2 );
      lista2.removeLast();
      lista2.removeLast();
      System.out.println(lista2 );
  }
}
```

Zadanie. 2

Utworzyć kolekcję z elementów, które są przekazane z linii poleceń (korzystając z pętli for), a następnie (korzystając z Iteratora) wypisać te elementy. Elementy są typu String.

```
public class Iteracja {
   public static void main(String args[]) {
     Collection <String> coll = new ArrayList<String>();
     for (int i = 0; i < args.length; i++) {
         coll.add(args[i]);
     }
     for (Iterator<?> iter = coll.iterator(); iter.hasNext();){
          String element = iter.next();
          System.out.println("element = " + element);
     }
  }
}
```

Zadanie 3.

Napisać program, który elementy przekazane z linii poleceń wstawia do kolejki, a następnie elementy te są wypisywane i usuwane z kolejki (za każdym razem usuwana jest element stanowiący jej głowę). Wprowadzane elementy są typu String.

```
public class Odliczanie {
    public static void main(String[] args) {
        int liczba = Integer.parseInt(args[0]);
        Queue <Integer> kolejka = new LinkedList<Integer>();
        for (int i = liczba; i >= 0; i--) {
            kolejka.add(i);
            }
        while (!kolejka.isEmpty()) {
          System.out.println(kolejka.remove());
          }
    }
}
```

Zadanie 4.

Napisać program, który tworzy kolekcję tylko z elementów niepowtarzających się, przekazywanych z linii poleceń, (wykorzystać zbiór do eliminacji duplikatów). Elementy są typu String.

```
public class Eliminacja {
    public static void main(String args[]) {
        Set <String>zbior = new HashSet<String>();
        for (int i = 0; i < args.length; i++) {
          if (! zbior.add(args[i]))
          System.out.println("Duplikat!");
        }
    }
}
```

Zadanie 5.

Zmodyfikować program z zadania 4 tak, aby przy każdej próbie dodania duplikatu wypisywał go.

Następnie wypisuje ile jest słów unikalnych i wypisuje wszystkie słowa unikalne.

```
import java.util.*;

public class Eliminacja2  {
   public static void main(String[] args) {
      Set<String> zbior = new HashSet<String>();
      for (String a : args)
      if (!zbior.add(a))
         System.out.println("Duplikat: „+a);
      System.out.println(zbior.size()+ "unikalne słowa: " + zbior);
    }
}
```

Zadanie 6.

Napisać dwie metody:

show – która wyświetla elementy listy dowolnego typu.

append – która dodaje do listy elementy przekazane w tablicy, tablica i lista przekazywana jako parametr.

Następnie wykorzystać te metody w programie, który tworzy listę napisów (elementów typu String),Lista zaimplementowana jest jako ArrayList. zapisanych tablicy i wyświetla wszystkie elementy listy.

```
import java.util.*;
class ListUtils {
     static <T> void append(List<T> list, T[] items){
        for (T item : items)
            list.add(item);
     }
static void show(List<?> list) {
          for (Object o : list)
              System.out.println(o);
  }
}
class  Exemple{
  public static void main(String args[]) {
      String[] items = { "Bialy", "Czerwony", "Niebieski", "Zielony" };
      List<String> list1 = new ArrayList<String>();
      ListUtils.append(list1, items);
      ListUtils.show(list1);
  }
}
```

Zadanie 7.

Zmodyfikuj program z zadania 6 tak aby powstawała lista elementów typu Integer. Lista zaimplementowana jest jako LinkedList.

```
import java.util.*;
class ListUtils {
     static <T> void append(List<T> list, T[] items){
        for (T item : items)
            list.add(item);
     }
static void show(List<?> list) {
          for (Object o : list)
              System.out.println(o);
  }
}
class  Exemple{
  public static void main(String args[]) {
       Integer[] arr = {1, 2, 3};
       List<Integer> list1 = new LinkedList<Integer>();
       ListUtils.append(list1, arr);
       ListUtils.show(list1);
  }
}
```

Zadanie 8.

(łączenie dwóch list tego samego typu)

a) Utworzyć dwie listy napisów (elementów typu String), a następnie połączyć je używając operacji grupowych (bulk) addAll. Pozbyć się duplikatów. Wypisać wszystkie listy.
b) Utworzyć listę_1 elementów typu Integer, a następnie dokonując konwersji tablicy nowych elementów do listy, dołączyć do listy_1. Wypisać powstałą listę.

```
import java.util.*; // łączymy kolekcje o elementach tych samych typów
class ListUtils {
     static <T> void append(List<T> list, T[] items){
        for (T item : items)
            list.add(item);
     }
static void show(List<?> list) {
          for (Object o : list)
              System.out.println(o);
  }
}
class  Exemple{
  public static void main(String args[]) {
       String[] items = { "Bialy", "Czerwony", "Niebieski", "Zielony" };
       List<String> list1 = new ArrayList<String>();
       ListUtils.append(list1, items);
       ListUtils.show(list1);
       List <String> list2 =new ArrayList<String>(Arrays.asList("Zolty", "Fioletowy" ));
       list1.addAll(list2);
       System.out.println(list2);
       // pozbywamy się duplikatów
        Set <String> set1 = new HashSet<String>(list1);
       System.out.println(set1);
       List <Integer> list3 = new ArrayList<Integer>(Arrays.asList(1,2,3,4,5));
       System.out.println(list3);
       list3.addAll(Arrays.asList(35, 1, 2, 3));
       System.out.println(list3);
  }
}
```

Zadanie 9.

(łączenie dwóch list różnych typów)

a) Utworzyć dwie listy jedną napisów (elementów typu String), a drugą elementów typu Integer.
b) Połączyć te dwie listy: Utworzyć trzecią listę typu Object z listy1, a następnie dołączyć do tej listy listę2. Wypisać powstałą listę.
c) Zsumować wszystkie liczby Integer oraz wypisać napis składający się z elementów typu String.

```
import java.util.*; //teraz laczymy kolekcje o różnych typach elementów
class ListUtils {
     static <T> void append(List<T> list, T[] items{
        for (T item : items)
            list.add(item);
     }
     static <T> List<T> create(T[] items) {
      List<T> list = new ArrayList<T>();
       append(list, items);
       return list;
     }
     static void show(List<?> list) {
          for (Object o : list)
              System.out.println(o);
  }
}
class  Exemple{
  public static void main(String args[]) {
       String[] items = { "Bialy", "Czerwony", "Niebieski", "Zielony" };
       List<String> list1 = new ArrayList<String>();
       ListUtils.append(list1, items);
       ListUtils.show(list1);
       list2 = new ArrayList<Integer>(Arrays.asList(1,2,3,4,5));
        List<Object> list3 = new ArrayList<Object>(list1);
        list3.addAll(list2);
        System.out.println(list3); // przy iteracjach już musimy posługiwać się typem Object // i ew. robić konwersje zawężające
       int sum = 0;
       String txt = "";
       for(Object o : list3) {
          if (o instanceof Integer) sum += (Integer) o;
            else txt += (String) o;
          }
       System.out.println("Suma liczb: " + sum);
       System.out.println("Napis: " + txt);
  }
}
```

Zadanie 10.

a) Utworzyć listę elementów typu String, wypisać listę.
b) Następnie korzystając z metody mieszającej shuffle, zmienić kolejność elementów. Wypisać listę.
c) Usunąć fragment listy (od 2 do 4).wypisać listę.

```
import java.util.*;  // mieszamy elementy listy i usuwamy fragment listy
class ListUtils {
     static <T> void append(List<T> list, T[] items{
        for (T item : items)
            list.add(item);
     }
static void show(List<?> list) {
          for (Object o : list)
              System.out.println(o);
    }
     static void shuffle(List<?> list, Random rnd) {
     for (int i = list.size(); i > 1; i--)
        swap(list, i - 1, rnd.nextInt(i));
    }
}
class  Exemple{
  public static void main(String args[]) {
       String[] items = { "Bialy", "Czerwony", "Niebieski", "Zielony" };
       List<String> list1 = new ArrayList<String>();
       ListUtils.append(list1, items);
       ListUtils.show(list1);
       ListUtils.shuffle(list1);
       ListUtils.show(list1);
       lis1t.subList(2,4).clear();
       ListUtils.show(list1);
}
```

Zadanie 10.

Utworzyć listę elemntów typu String, a następnie przekonwertować ją do tablicy.

```
import java.util.*;
class ListUtils {
     static <T> void append(List<T> list, T[] items{
        for (T item : items)
            list.add(item);
     }
static void show(List<?> list) {
          for (Object o : list)
              System.out.println(o);
  }
}
class  Exemple{
// z listy robimy tabicę
   public static void main(String args[]) {
       String[] items = { "Bialy", "Czerwony", "Niebieski", "Zielony" };
       List<String> list1 = new ArrayList<String>();
       ListUtils.append(list1, items);
       ListUtils.show(list1);
       String[] tab = (String[]) list1.toArray(new String[0]);
       for (String string : tab) {
          System.out.println(string + " " + string.length()); }
   }
}
```

```
import java.util.*; // łączymy kolekcje o elementach tych samych typów
class ListUtils {
     static <T> void append(List<T> list, T[] items){
        for (T item : items)
            list.add(item);
     }
     static <T> List<T> create(T[] items) {
       ArrayList<T> list = new ArrayList<T>();
       append(list, items);
       return list;
     }
     static void show(List<?> list) {
          for (Object o : list)
              System.out.println(o);
   }
}
class  Exemple{
   public static void main(String args[]) {
        String[] items = { "Bialy", "Czerwony", "Niebieski", "Zielony" };
        List<String> list1 = new ArrayList<String>();
        ListUtils.append(list1, items);
        ListUtils.show(list1);
        List<String> list2 = ListUtils.create( new String[] { "Zolty", "Fioletowy"} );
        list1.addAll(list2);
        System.out.println(list2);
        // pozbywamy się duplikatów
         Set<String> set1 = new HashSet<String>(list1);
        System.out.println(set1);
        list3 = new ArrayList<Integer>(Arrays.asList(1,2,3,4,5));
        System.out.println(list3);
         list3.addAll(Arrays.asList(35, 1, 2, 3));
         System.out.println(list3);
   }
}
```