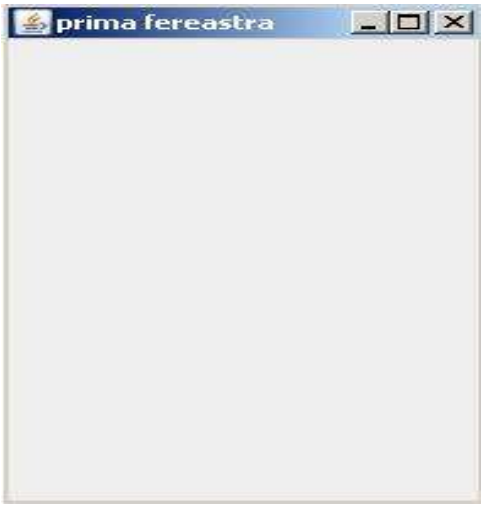
Laborator 1 – Aplicatii Java

1. Introducere in programarea vizuala (Pachetele AWT si Swing. Ferestre)
2. Colectii

3.1. Introduceti urmatorul program JAVA:

```
import javax.swing.*;
        public class Pv{
        public static void main(String args[ ]){
        JFrame fer=new JFrame("prima fereastra");
        fer.setSize(200,300);
        fer.setLocation(300,400);
        fer.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fer.setVisible(true);
        }
}
```

Rezultatul executiei acestui program este aparitia urmatoarei ferestre:



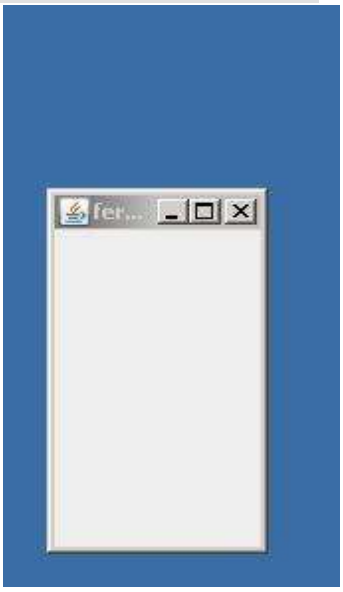
3.2. Introduceti urmatorul program JAVA:

Se poate construi o clasa, numita Fereastra, al carei constructor sa returneze obiectul cu toate datele din exemplul anterior:

```
import javax.swing.*;
class Fereastra extends JFrame{
    Fereastra(String Nume, int lat, int inalt, int dreapta, int stanga)
    {
                super(Nume);
                setSize(lat,inalt);
                setLocation(dreapta, stanga);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setVisible(true);
```

```
    }
}
    public class Pv{
        public static void main(String args[ ]){
            Fereastra f=new Fereastra("fereastra", 100, 200, 25, 100);
        }
}
```

Rezultatul executiei acestui program este aparitia urmatoarei ferestre:

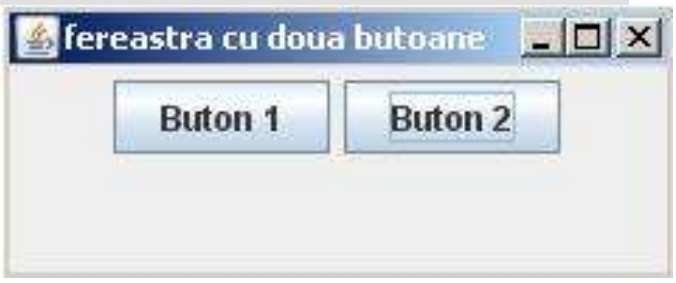


3.3. Introduceti urmatorul program JAVA:

Exemplu de program care afiseaza doua butoane (fisierul *Pv1.java*). Apasarea butoanelor nu are nici un efect.

```
import java.awt.*;
import javax.swing.*;
class Fer extends JFrame{
        public Fer(String titlu)
        {
                super(titlu);
                setSize(200,100);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                Container x = getContentPane();
                x.setLayout(new FlowLayout());
                JButton A = new JButton("Buton 1"); x.add(A);
                JButton B = new JButton("Buton 2"); x.add(B);
                setVisible(true);
        }
}
public class Pv1{
        public static void main(String args[ ]){
           Fer fp = new Fer("fereastra cu doua butoane");
        }
}
```

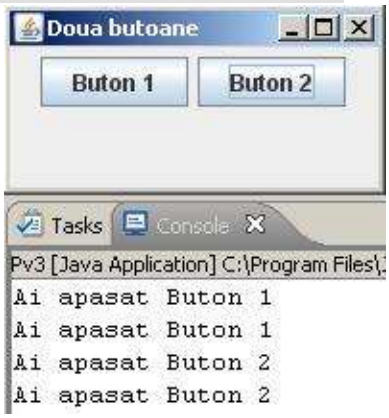
Rezultatul executiei acestui program este aparitia urmatoarei ferestre:



3.4. Introduceti urmatorul program JAVA:

Am extins programul anterior. Cand se apasa un buton, in fereastra **CMD** (Console din Eclipse) va aparea sirul retinut de butonul apasat.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class Fer extends JFrame implements ActionListener{
	public Fer(String titlu)
	{
		super(titlu);
		setSize(200,100);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container x = getContentPane();
		x.setLayout(new FlowLayout());
		JButton A = new JButton("Buton 1"); x.add(A);
		JButton B = new JButton("Buton 2"); x.add(B);
		A.addActionListener(this);
		B.addActionListener(this);
		setVisible(true);
	}
public void actionPerformed(ActionEvent e)
{
	if(e.getActionCommand().compareTo("Buton 1") == 0)
		System.out.println("Ai apasat Buton 1");
	else
		System.out.println("Ai apasat Buton 2");
	}
}
public class Pv3{
	public static void main(String args[ ]){
	Fer fp = new Fer("Doua butoane");
	}
}
```



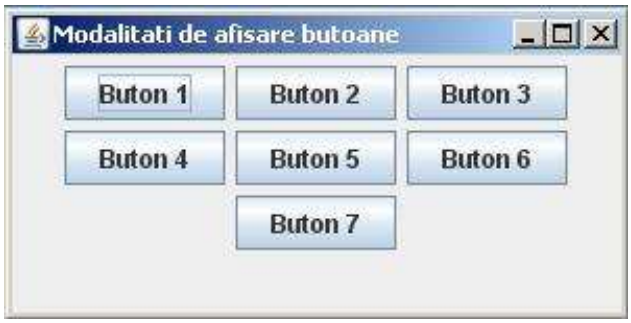
Rezultatul executiei acestui program este aparitia urmatoarei ferestre:

3.5. Introduceti urmatorul program JAVA:

```
import java.awt.*;
```

```
import javax.swing.*;
class Fer extends JFrame
{
        public Fer(String titlu)
        {
                super(titlu);
                setSize(300, 150);
                Container x = getContentPane();
                x.setLayout(new FlowLayout());
                JButton A = new JButton("Buton 1"); x.add(A);
                JButton B = new JButton("Buton 2"); x.add(B);
                JButton C = new JButton("Buton 3"); x.add(C);
                JButton D = new JButton("Buton 4"); x.add(D);
                JButton E = new JButton("Buton 5"); x.add(E);
                JButton F = new JButton("Buton 6"); x.add(F);
                JButton G = new JButton("Buton 7"); x.add(G);
                setVisible(true);
    }
}
public class Pv4
{
  public static void main(String args[ ])
  {
    Fer fp=new Fer("Modalitati de afisare butoane");
  }
}
```

Rezultatul executiei acestui program este aparitia urmatoarei ferestre:



3.6. Introduceti urmatorul program JAVA:

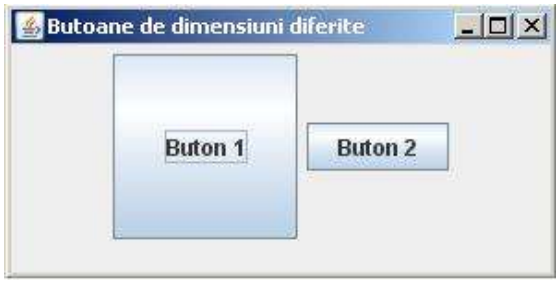
Prin utilizarea gestionarului FlowLayout se adauga unei ferestre, doua butoane:

- primul de dimensiune stabilita,
- iar al doilea de dimensiune implicita:

```
import java.awt.*;
import javax.swing.*;
class Fer extends JFrame
{
```

```
        public Fer(String titlu)   {
                super(titlu);
                setSize(300, 150);
                Container x = getContentPane();
                x.setLayout(new FlowLayout());
                JButton A = new JButton("Buton 1");
                A.setPreferredSize(new Dimension(100,100));
                x.add(A);
                JButton B = new JButton("Buton 2"); x.add(B);
                setVisible(true);
    }
}
public class Pv6 {
  public static void main(String args[ ])  {
            Fer fp=new Fer("Butoane de dimensiuni diferite");
  }
}
```

Rezultatul executiei acestui program este aparitia urmatoarei ferestre:



3.7. Introduceti urmatorul program JAVA:

Clasa BorderLayout – imparte suprafata ferestrei in 5 parti:

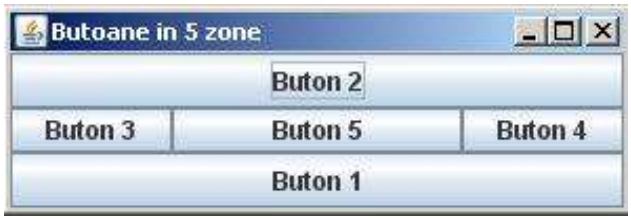
- nord(NORTH),
- sud(SOUTH),
- est(EAST),
- vest(WEST)
- si centru (CENTER).

In fiecare parte se poate aseza o componenta.

```
import java.awt.*;
import javax.swing.*;
class Fer extends JFrame
{
        public Fer(String titlu)  {
            super(titlu);
            setSize(300, 100);
            Container x = getContentPane();
            x.setLayout(new BorderLayout());
            JButton A = new JButton("Buton 1"); x.add(A, BorderLayout.SOUTH);
            JButton B = new JButton("Buton 2"); x.add(B, BorderLayout.NORTH);
            JButton C = new JButton("Buton 3"); x.add(C, BorderLayout.WEST);
            JButton D = new JButton("Buton 4"); x.add(D, BorderLayout.EAST);
```

```
            JButton E = new JButton("Buton 5"); x.add(E,
BorderLayout.CENTER);
            setVisible(true);
    }
}
public class Pv6 {
 public static void main(String args[ ])  {
            Fer fp=new Fer("Butoane in 5 zone");
 }
}
```

Rezultatul executiei acestui program este aparitia urmatoarei ferestre:



3.8. Aplicatie de tip fereastra care are un obiect de tip JButton, si unul de tip JLabel. Atunci cand se efectueaza un click de mouse pe buton, eticheta, eticheta va afisa textul "A-ti facut click de n ori" unde n reprezinta numarul de apasari inregistrate pana in acel moment.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TestFrame extends JFrame implements ActionListener{
        private JButton buton;
        private JLabel eticheta;
        private JPanel panou;
        private int contor=0;

        public TestFrame()
        {
                addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent ev){
                                System.exit(0);
                }});
                Container contentPane = getContentPane();
                panou = new JPanel();
                panou.setLayout(new BorderLayout());
                buton = new JButton("Apasa");
                panou.add(buton, BorderLayout.SOUTH);
                buton.addActionListener(this);
                eticheta = new JLabel();
                eticheta.setHorizontalAlignment(JLabel.CENTER);
                panou.add(eticheta,BorderLayout.NORTH);
                contentPane.add(panou);
        }
```

```
        public void actionPerformed(ActionEvent ev) {
                if(ev.getSource() instanceof JButton)
                {
                        if(ev.getSource() == buton )
                        {
                          contor++;
                          eticheta.setText("A-ti facut click de " + String.valueOf(contor)+
" ori");
                        }
                }
        }
        public static void main ( String[] args ) {
                TestFrame f = new TestFrame();
                f.setSize(150,100);
                f.setVisible(true);
        }
}
```

Rezultatul executiei acestui program este aparitia urmatoarei ferestre:

Probleme propuse spre rezolvare

1. Modificati programul 3.1, astfel incat sa afisati pe ecran doua ferestre, in zone diferite si care sa fie cu titlurile “Window 1”, respectiv “Window 2”.

2. Modificati programul 3.2, astfel incat sa afisati pe ecran doua ferestre, in zone diferite si care sa fie cu titlurile “Fereastra 1”, respectiv “Fereastra 2”.

3. Modificati programul 3.3, astfel incat sa afisati pe ecran sase butoane, notate “Button 1”, “Button 2”, “Button 3”, “Button 4”, “Button 5”, “Button 6”.

4. Modificati programul 3.3, astfel incat sa afisati pe ecran sase butoane, notate “Buton 1”, “Buton 2”, “Buton 3”, “Buton 4”, “Buton 5”, “Buton 6”, iar la apasarea fiecarui buton sa se afiseze mesaj corespunzator butonului apasat.

5. Modificati programul 3.5, astfel incat sa afisati pe ecran noua butoane, notate corespunzator si care sa apara cate doua pe linie.

6. Modificati programul 3.6, astfel incat sa afisati pe ecran patru butoane, notate corespunzator si care sa fie de dimensiuni diferite.

7. Introduceti programul 3.8 si verificati executia astfel incat rezultatul sa fie cel specificat.

2. Colectii

Scrieti, compilati si rulati toate exemplele din acest laborator:

1. Urmatorul exemplu foloseste diferite clase de colectii si adauga cate un element in aceste colectii

```
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.TreeMap;
import java.util.TreeSet;
Exemplu de tipuri de clase
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
public class Main {
  public static void main(String[] args) {
    List lnkLst = new LinkedList();
    lnkLst.add("element1");
    lnkLst.add("element2");
    lnkLst.add("element3");
    lnkLst.add("element4");
    displayAll(lnkLst);
List aryLst = new ArrayList();
    aryLst.add("x");
    aryLst.add("y");
    aryLst.add("z");
    aryLst.add("w");
    displayAll(aryLst);
Set hashSet = new HashSet();
    hashSet.add("set1");
    hashSet.add("set2");
    hashSet.add("set3");
    hashSet.add("set4");
    displayAll(hashSet);
SortedSet treeSet = new TreeSet();
    treeSet.add("1");
    treeSet.add("2");
    treeSet.add("3");
    treeSet.add("4");
    displayAll(treeSet);
LinkedHashSet lnkHashset = new LinkedHashSet();
```

```
        lnkHashset.add("one");
        lnkHashset.add("two");
        lnkHashset.add("three");
        lnkHashset.add("four");
        displayAll(lnkHashset);
    Map map1 = new HashMap();
        map1.put("key1", "J");
        map1.put("key2", "K");
        map1.put("key3", "L");
        map1.put("key4", "M");
        displayAll(map1.keySet());
        displayAll(map1.values());
    SortedMap map2 = new TreeMap();
        map2.put("key1", "JJ");
        map2.put("key2", "KK");
        map2.put("key3", "LL");
        map2.put("key4", "MM");
        displayAll(map2.keySet());
        displayAll(map2.values());
    LinkedHashMap map3 = new LinkedHashMap();
        map3.put("key1", "JJJ");
        map3.put("key2", "KKK");
        map3.put("key3", "LLL");
        map3.put("key4", "MMM");
        displayAll(map3.keySet());
        displayAll(map3.values());
      }
    static void displayAll(Collection col) {
        Iterator itr = col.iterator();
        while (itr.hasNext()) {
            String str = (String) itr.next();
            System.out.print(str + " ");
        }
        System.out.println();
      }
    }
```

Rezultatul obtinut este:

```
element1 element2 element3 element4
x y z w
set1 set2 set3 set4
1 2 3 4
one two three four
key4 key3 key2 key1
M L K J
key1 key2 key3 key4
JJ KK LL MM
key1 key2 key3 key4
JJJ KKK LLL MMM
```

2. Sa se determine dimensiunea(numarul de elemente) unei colectii?

Se vor utiliza metodele:
collection.add() – pentru adaugarea unui nou element intr-o colectie
collection.size() – pentru a afla dimensiunea colectiei din clasa Collection

```
import java.util.*;
public class CollectionTest {
  public static void main(String [] args) {
    int size;
    HashSet collection = new HashSet ();
    String str1 = "Galben", str2 = "Alb", str3 = "Verde", str4 = "Albastru";
    Iterator iterator;
    collection.add(str1);
    collection.add(str2);
    collection.add(str3);
    collection.add(str4);
    System.out.print("Informatiile colectiei: ");
    iterator = collection.iterator();
    while (iterator.hasNext()){
      System.out.print(iterator.next() + " ");
    }
    System.out.println();
    size = collection.size();
    if (collection.isEmpty()){
      System.out.println("Collection este vida");
    } else{
      System.out.println( "Numarul de elemente din Collection: " + size);
    }
    System.out.println();
  }
}
```

Rezultatul obtinut este:

```
Informatiile colectiei: Albastru Alb Verde Galben
Numarul de elemente din Collection: 4
```

3. Sa se afle valoarea minima si maxima dintr-o colectie de tip List?

```
import java.util.*;
public class Main {
  public static void main(String[] args) {
    List list = Arrays.asList("one Two three Four five six
    one three Four".split(" "));
    System.out.println(list);
    System.out.println("max: " + Collections.max(list));
```

```
    System.out.println("min: " + Collections.min(list));
  }
}
```

Rezultatul:

```
[one, Two, three, Four, five, six, one, three, Four]
max: three
min: Four
```

4. Sa se implementeze o clasa Catalog care sa permita urmatoarele operatii:

1) Adaugarea unui student (nume, media)
2) Afisarea tuturor studentilor
3) Cautarea unui student dupa nume
4) Stergerea unui student dupa nume
5) Ordonarea studentilor alfabetic dupa nume
6) Ordonarea studentilor dupa medie

```
import java.util.*;
class Student{
	public String nume;
	public int media;
	public Student (String nume, int media){
		this.nume = nume;
		this.media = media;
	}
	public void afisare (){
		System.out.println ("Nume: " + nume + " media: " + media);
	}
};
class ComparatorNume implements Comparator
{
	public int compare (Object o1, Object o2)
	{
		Student s1 = (Student) o1;
		Student s2 = (Student) o2;
// String.compareTo este echivalenta cu functia strcmp
		return s1.nume.compareTo (s2.nume);
	}
}
class ComparatorMedie implements Comparator
{
	public int compare (Object o1, Object o2)
	{
		Student s1 = (Student) o1;
		Student s2 = (Student) o2;
		return s1.media - s2.media;
	}
```

```
}
class Catalog
{
	// in acest exemplu, putea fi si private ArrayList lista
	private LinkedList lista;
	public Catalog ()
	{
	// daca foloseam ArrayList, inlocuiam aceasta linie cu:
	// lista = new ArrayList ()
		lista = new LinkedList ();
	}
public void adaugare (Student s){
	lista.add (s);
}
public void afisare (){
	Iterator it = lista.iterator ();
	while (it.hasNext ())
	{
		Student crt = (Student) it.next ();
		crt.afisare ();
	}
}
public Student cautare (String nume)
{
	Iterator it = lista.iterator ();
	while (it.hasNext ())
	{
		Student crt = (Student) it.next ();
		if (crt.nume.equals (nume))
			return (crt);
	}
	return null; //daca nu s-a gasit
}
public void stergere (String nume)
{
	Iterator it = lista.iterator ();
	while (it.hasNext ())
	{
		Student crt = (Student) it.next ();
		if (crt.nume.equals (nume)) it.remove ();
	}
}
public void ordonare_medie ()
{
	Collections.sort (lista, new ComparatorMedie ());
}

public void ordonare_alfabetic (){
	Collections.sort (lista, new ComparatorNume ());
```

```
}
}
public class Problema_rezolvata
{
public static void main(String[] args)
{
        Catalog c = new Catalog ();
        Student s1 = new Student ("Ionescu", 10);
        Student s2 = new Student ("Vasile", 5);
        Student s3 = new Student ("Popescu", 8);
        Student s4 = new Student ("Georgescu", 1);
        c.adaugare (s1);
        c.adaugare (s2);
        c.adaugare (s3);
        c.adaugare (s4);
        System.out.println ("Dupa adaugare:");
        System.out.println("----------------------------");
        c.afisare ();
        Student s = c.cautare ("Georgescu");
        if (s != null){
                System.out.println("----------------------------");
                System.out.println ("Am gasit studentul:");
                s.afisare ();
                System.out.println("----------------------------");
        }
        else
                System.out.println ("Georgescu nu a fost gasit !");
        c.stergere ("Georgescu");
        System.out.println("----------------------------");
        System.out.println ("Dupa stergere:");
        System.out.println("----------------------------");
        c.afisare ();
        c.ordonare_alfabetic ();
        System.out.println("----------------------------");
        System.out.println ("Dupa ordonare alfabetic:");
        System.out.println("----------------------------");
        c.afisare ();
        c.ordonare_medie ();
        System.out.println("----------------------------");
        System.out.println ("Dupa ordonare dupa medie:");
        System.out.println("----------------------------");
        c.afisare ();
        }
}
```

Probleme propuse spre rezolvare

1. Sa se creeze o clasa care sa contina ca data membra privata un vector (tablou unidimensional) de numere întregi de dimensiune variabila, care se stabileste pentru fiecare obiect la constructia acestuia.
În aceasta clasa redefiniti functia toString(), care sa afiseze numerele continute.
În functia main() a clasei principale construiti un obiect din clasa creata care sa contina numerele 2, 9, 4, 5, 7, 8 si afisati continutul acestuia folosind functia toString() a clasei respective.

2. Sa se creeze o clasa Complex, pentru definirea unui numar complex, cu partea reala si imaginara ca numere de tip double.
Definiti constructorii, functia de afisare (toString()), functia de testare a egalitatii a doua obiecte (equals()) , o functie care realizeaza adunarea a doua numere complexe.
În functia main() creati doua obiecte din clasa Complex, cu valorile 2, 4, respectiv 5, 6 a partilor reale si imaginare.
Verificati daca cele doua obiecte sunt egale si afisati rezultatul la consola.
Calculati si afisati suma celor doua numere complexe.

3. Sa se creeze un program care sa contina urmatoarele clase si interfete:
Clasa Persoana care contine variabila membra private String nume, functiile necesare pentru citirea si scrierea acestei variabile, constructori, etc.
Interfata Adresa care declara functiile abstracte String getAdresa() si void setAdresa(String s);
Clasa Student care extinde clasa Persoana si implementeaza interfata Adresa.
Clasa Profesor care extinde clasa Persoana si implementeaza interfata Adresa.
În functia main() a programului sa se creeze un obiect Student, cu numele Ionescu si adresa Targu-Jiu si un obiect Profesor cu numele Popescu si adresa Craiova.
Afisati la consola datele celor doua obiecte (numele si adresa).

Referinte bibliografice:

1. Curs practic de Java, Cristian Frasinaru – capitolul Colectii
2. http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html
3. http://www.tutorialspoint.com/java/java_collections.htm