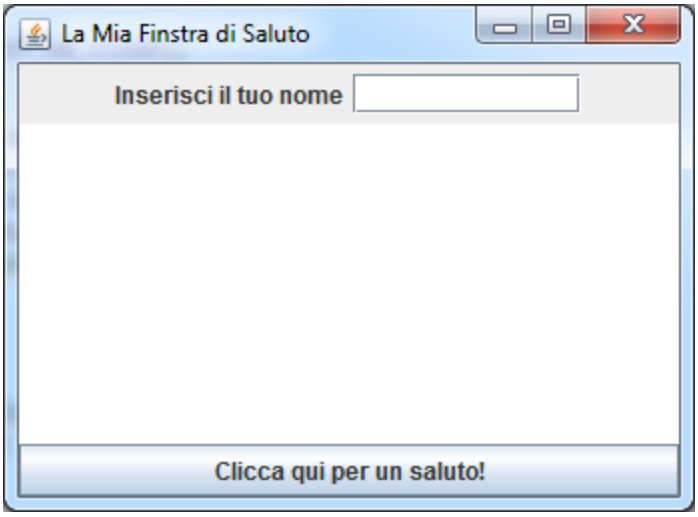
Esercitazione N4: Costruzione di una applicazione GUI utilizzando i componenti di base per realizzare l'input e l'output e cioè Label, TextBox, TextArea Button e Panel (Pannelli)

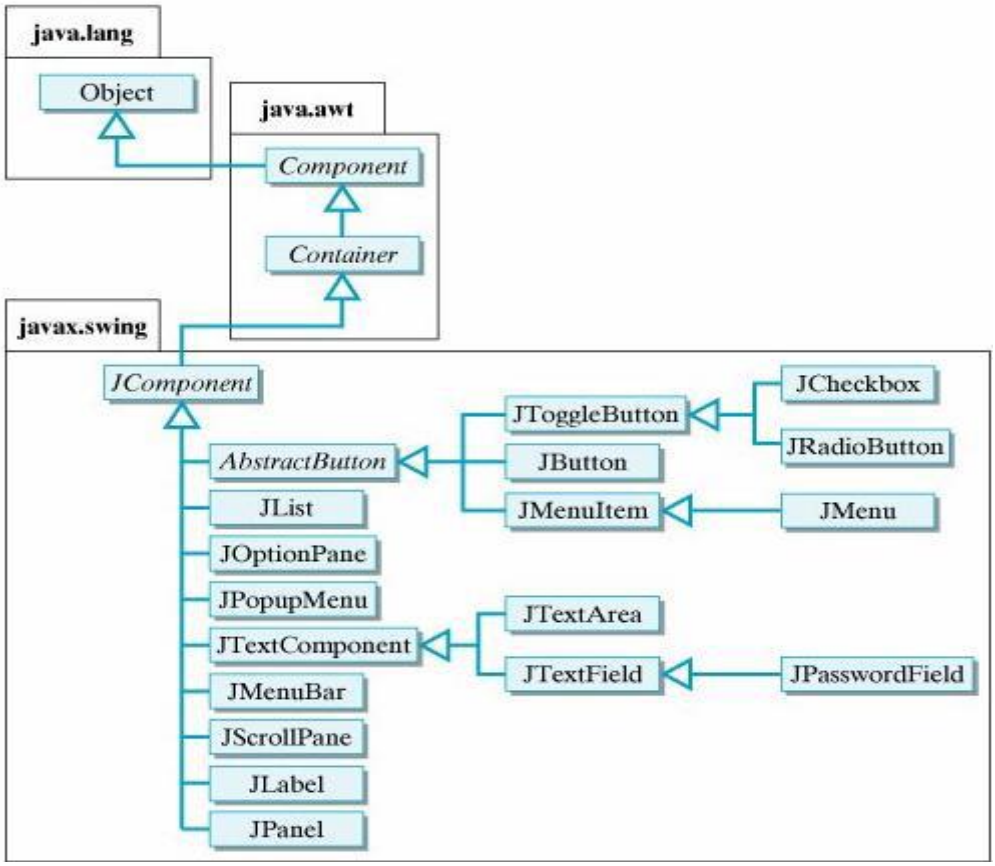
I componenti che utilizzeromo saranno quindi:

- Una **`JTextField`** che accetterà l'input dall'utente.
- Una **`JTextArea`** che visualizzerà l'output del programma.
- Un **`JButton`** che permette all'utente di attivare un evento per il saluto.
- Una **`JLabel`** che serve come testo di commento per la JTextField.
- Un Panel per raggruppare componenti

La nostra finestra dovrebbe essere come mostra la figura:



Diamo innanzitutto un'occhiata alle componenti principali di input/output:



Ora per costruire un oggetto partendo dalle componenti sopraelencate basta ovviamente usare i costruttori delle classi corrispondenti agli oggetti desiderati. Il codice seguente mostra come fare per i nostri 4 oggetti:

```
// Dichiarazione delle variabili di istanza delle componenti
private JLabel prompt;
public JTextField txfNome;
public JTextArea txaDisplay;
private JButton btnSaluto;
// Istanze delle component
prompt= new JLabel("Inserisci il tuo nome");
txfNome = new JTextField(10);// 10 caratteri
txaDisplay = new JTextArea(10, 30);// 10 righe x 30 colonne
btnSaluto = new JButton("Clicca qui per un saluto!");
```

Public metodi e costruttori per le componenti di base:

JButton
+ JButton(text : String) + addActionListener(al : ActionListener) + setEnabled(b : boolean)

JLabel
+ JLabel(text : String)

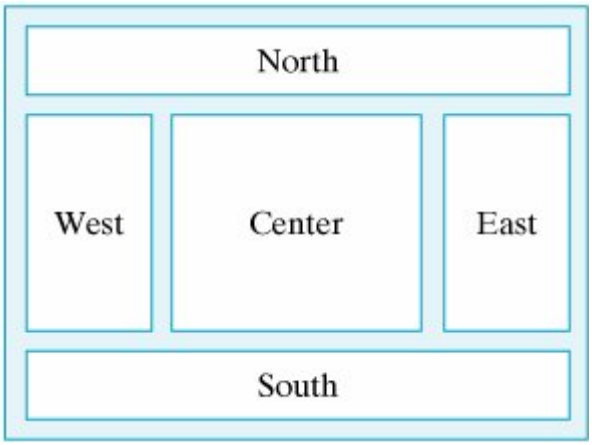
JTextField
+ JTextField(col : int) + JTextField(text : String) + getText() : String + addActionListener(al : ActionListener) + setEnabled(b : boolean)

JTextArea
+ JTextArea(row : int, col : int) + append(text : String) + setText(text : String)

Ora che sappiamo come costruire le nostre componenti di base dovremmo aggiungerle alla nostra finestra **JFrame**. Ma invece di aggiungerli direttamente ad essa noi la aggiungeremo ad un **Content Pane** che a sua volta sarà aggiunto alla finestra. Perché in java non si possono aggiungere direttamente componenti a un Jframe ma bisogna passare attraverso un pannello di contenuto del **JFrame**. La classe conteiner di java a diversi metodi per aggiungere dei componenti ad esso. I più importanti sono:

```
add(Component comp)  // aggiunge il componente al conteiner
add(Component comp, int index)  // posizione data dall'indice
add(String region, Component comp)  //posizione data dalla regione
```

Il metodo add in particolare che andremo ad utilizzare dipenderà da dove vogliamo andare a disporre le componenti sul conteiner. Il Layout del conteiner dipende dal suo default layout manager. Un oggetto associato con il conteiner che determina le dimensioni e le posizioni dei component sul conteiner. Per un ContentPane il layot di default è un **BorderLayout** con questo layout le componenti possono essere disposte al centro del cointener o lungo i bordi utilizzando le coordinate North South East West. La figura seguente ne mostra le coratteristiche:



Per aggiungere componenti ad un BordeLayout si usa il metodo:
add(region,component) dove region specifica una delle aree mostrate in figura e component può essere un componente qualsiasi. Esempio:

```
Container contentPane = getContentPane();//Si preleva il contenitore
contentPane.add(BorderLayout.CENTER, txaDisplay); // Add JTextArea
```

oppure in colpo solo

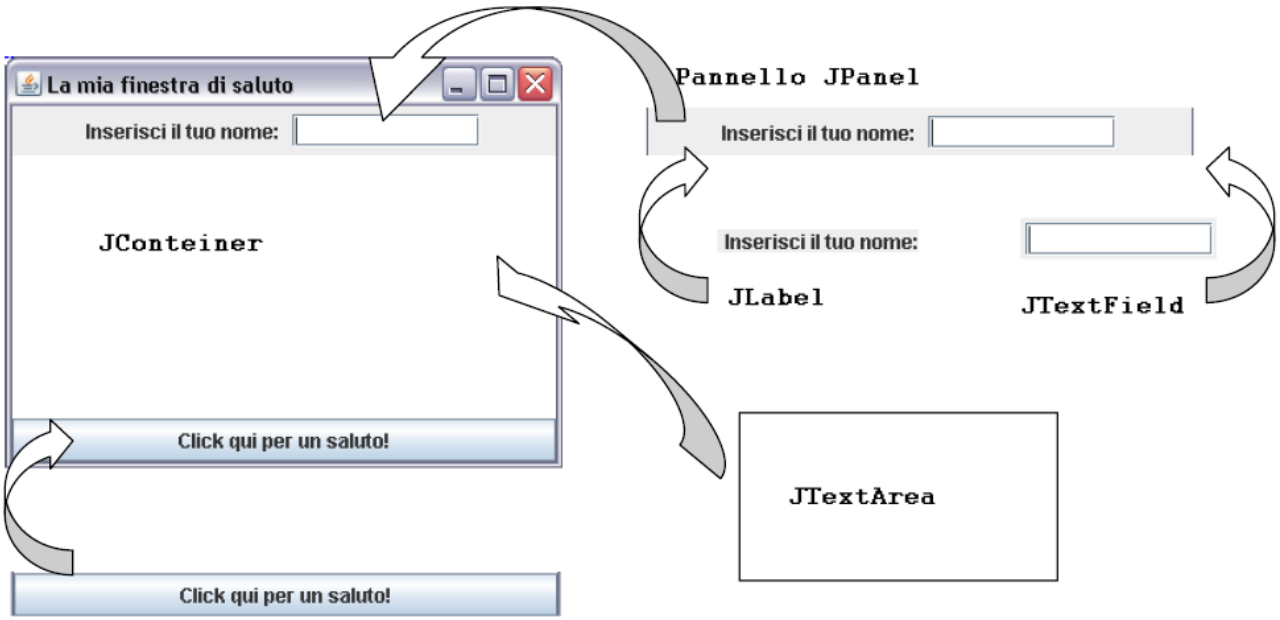
```
getContentPane().add (BorderLayout.CENTER, txaDisplay);
```

Il problema di questo layout è che noi possiamo aggiungere solo un componente per ogni area per cui non possiamo avere nella zona North le due componenti **Label** e **TextField** che rappresentano il prompt per l'utente e la casella di testo per fargli inserire il nome. Per sopperire a questa limitazione si utilizzano i pannelli (**JPanel)** con i quali il metodo di costruzione è rappresentato dalle operazioni seguenti:

- si costruisce un pannello
- si aggiungono componenti al pannello
- si aggiunge l'intero pannello al conteiner

```
JPanel pnlInput = new JPanel();// Creo il pannello
pnlInput.add(prompt); // Gli aggiungo la label
pnlInput.add(txfNome); // Glia ggiungo la casella di testo
// Aggiungo il panello alla finestra nella regione Nord
getContentPane().add(BorderLayout.North", pnlInput);
```

Il Layout di default per un **JPanel** è il **FlowLayout** in pratica le componenti vengono aggiunte una di seguito all'altra. Va benissimo per la nostra applicazione.
La seguente figura illustra graficamente il concetto di costruzione dell'aspetto della finestra:



Il codice seguente svolge le azioni precedentemente descritte e dovrebbe essere abbastanza semplice interpretarlo.

```
pnlInput.add(prompt);
pnlInput.add(txfNome);
frame.getContentPane().add(BorderLayout.NORTH, pnlInput);
frame.getContentPane().add(BorderLayout.CENTER, txaDisplay);
frame.getContentPane().add(BorderLayout.SOUTH, btnSaluto);
frame.pack();//dimensiona la finestra esattamente al suo contenuto
```

Ora che sappiamo come aggiungere componenti ad una finestra vediamo come possiamo gestirne il funzionamento. Come abbiamo già visto in un ambiente GUI si utilizza una programmazione guidata dagli eventi. Quindi creiamo l'oggetto ActionListener da aggiungere al bottone
Quando un utente pigia un bottone java crea un oggetto **ActionEvent** contenente svariate informazioni sull'evento accaduto come il nome del componente, il tempo la posizione ecc.
Per la nostra applicazione quindi quando l'utente fa click sul bottone **btnSaluto** l'applicazione deve copiare il contenuto della casella di input aggiungerci un saluto e visualizzarlo nella **TextArea**. Il tutto potrebbe essere effettuato con il frammento di codice seguente:

```
String name = txfNome.getText();
txaDisplay.append("Ciao " + name+" è un vero piacere conoscerti\n");
```

La domanda è dove scrivere questo codice e quando richiamarlo? La risposta è abbastanza semplice questo codice deve essere inserito in un metodo speciale che viene richiamato direttamente da java nel momento in cui accade un determinato evento. In pratica questo metodo noi lo conosciamo già è il famoso metodo actionPerformed dell'interfaccia ActionListener che noi dobbiamo implementare per gestire l'evento click sul bottone.

```
public abstract interface ActionListener extends EventListener
{ public abstract void actionPerformed(ActionEvent e);
}
```

Quindi per il nostro listener andremo ad implementare il metodo actionPerformed come segue:

```
public void actionPerformed(ActionEvent e)
{
String name = txfNome.getText();
txaDisplay.append("Ciao " + name+" è un vero piacere conoscerti\n");
}
```

e poi aggiungere al bottone un listener con il solito metodo addActionListener:

btnSaluto = new JButton("Clicca qui per un saluto!");
btnSaluto.addActionListener(new btnSaluto_Click());

In questo esercizio implementiamo la classe ascoltatore direttamente come classe interna alla classe finestra, il codice della finestra e la classe programma sono simili alle esercitazioni precedenti:

```
private void inizializza()
{
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    pnlInput.setBackground(new Color(0,0,255));
    prompt.setFont(new Font("Tahoma", Font.BOLD, 18));
    prompt.setForeground(new Color(255,0,0));
    pnlInput.add(prompt);
    pnlInput.add(txfNome);
    txaDisplay.setFont(new Font("Tahoma", Font.ITALIC, 16));
    this.getContentPane().add(BorderLayout.NORTH, pnlInput);
    this.getContentPane().add(BorderLayout.CENTER, txaDisplay);
    this.getContentPane().add(BorderLayout.SOUTH, btnSaluto);
    btnSaluto.addActionListener(new btnSaluto_Click());
    this.pack();//dimensiona la finestra esattamente al suo contenuto

 }

/*
 * Classe interna alla finestra che implementa l'ascolatatore di eventi
 * la scelta della classe interna semplifica l'accesso alle componenti della finestra
 */

public class btnSaluto_Click implements ActionListener
{
 public void actionPerformed(ActionEvent ae)
  {
     String name = txfNome.getText();
     txaDisplay.append("Ciao " + name+" è un vero piacere conoscerti\n");
 }
}
```