Personalizzazione della classe PannelloGrafico

1. INTRODUZIONE

La classe PannelloGrafico viene fornita come base da personalizzare per i propri homework. La Sezione 2 descrive le parti della classe che devono essere personalizzate. La Sezione 3 elenca i metodi di utilità messi a disposizione dalla classe.

2. PARTI DA PERSONALIZZARE

La classe contiene:

1. Gli statement di importazione dei package della libreria standard di Java necessari alla classe per gestire gli elementi grafici (*javax.swing* e *java.awt.event*). A questa parte devono ovviamente essere aggiunti evenutali altri statement di importazione di package necessari all'applicazione che si vuole realizzare.
2. La dichiarazione degli attributi della classe. A questa parte devono essere aggiunti gli attributi specifici dell'applicazione che si vuole realizzare.

```
/* INIZIO PARTE DA PERSONALIZZARE */

// Attributi dell'applicazione

/* FINE PARTE DA PERSONALIZZARE */
```

3. Un costruttore che imposta le caratteristiche del pannello (titolo della finestra, dimensioni delle matrici e numero di bottoni). Questa parte dev'essere personalizzata con i dati specifici dell'applicazione che si vuole realizzare. Si noti che, nel caso in cui si utilizzino entrambe le matrici (A e B), la dimensione massima delle matrici è 10 righe per 10 colonne. Nel caso in cui si utilizzi la sola matrice A (impostando 0 righe e 0 colonne per la matrice B), la dimensione massima della matrice è 10 righe per 23 colonne. Il costruttore successivamente invoca il metodo "inizializza" che si occupa di inizializzare gli elementi grafici.

```
public PannelloGrafico()

{     /* INIZIO PARTE DA PERSONALIZZARE */

      titoloFinestra = "Titolo finestra";

      numeroRigheA = 10;
```

```
        numeroColonneA = 10;
        numeroRigheB = 10;
        numeroColonneB = 10;
        numeroBottoni = 3;
        /* FINE PARTE DA PERSONALIZZARE */

        inizializza();
    }
```

4. Un metodo “bottonePremuto” che viene *invocato automaticamente* nel momento in cui l’utente preme un bottone. Il metodo riceve come parametro il numero del bottone che è stato premuto. Si noti che la numerazione dei bottoni parte da 1.

```
    private void bottonePremuto(int numeroBottone)
    {   /* INIZIO PARTE DA PERSONALIZZARE */
        setMessaggio("Premuto il bottone "+numeroBottone);
        /* FINE PARTE DA PERSONALIZZARE */
    }
```

   Questo metodo dev’essere personalizzato con le istruzioni da eseguire quando l’utente preme i vari bottoni.

5. Un metodo “main” che crea un’istanza della classe e imposta le etichette dei bottoni. Questa parte dev’essere personalizzata con le etichette specifiche per l’applicazione che si vuole realizzare.

```
    public static void main(String[] args)
    {   PannelloGrafico p = new PannelloGrafico();
        /* INIZIO PARTE DA PERSONALIZZARE */
        p.setEtichettaBottone(1, "Bottone 1");
        p.setEtichettaBottone(2, "Bottone 2");
        p.setEtichettaBottone(3, "Bottone 3");
        /* FINE PARTE DA PERSONALIZZARE */
    }
```

3. METODI DI UTILITÀ

La classe contiene:

- Un metodo **`void setMessaggio(String m)`** che imposta il messaggio da visualizzare all'utente.
- Un metodo **`void setEtichettaBottone(int numeroBottone, String etichetta)`** che imposta l'etichetta di un bottone.
- Un metodo **`String[][] getMatriceA()`** che restituisce il contenuto attuale della matrice A.
- Un metodo **`String[][] getMatriceB()`** che restituisce il contenuto attuale della matrice B.
- Un metodo **`void setMatriceA(String[][] A)`** che imposta il contenuto della matrice A.
- Un metodo **`void setMatriceB(String[][] B)`** che imposta il contenuto della matrice B.
- Un metodo **`void bloccaMatriceA()`** che rende la matrice A non modificabile.
- Un metodo **`void sbloccaMatriceA()`** che rende la matrice A modificabile.
- Un metodo **`void bloccaMatriceB()`** che rende la matrice B non modificabile.
- Un metodo **`void sbloccaMatriceB()`** che rende la matrice B modificabile.
- Un metodo **`int[][] convertiInMatriceIntera(String[][] m)`** che converte una matrice di stringhe in una matrice di interi.
- Un metodo **`String[][] convertiDaMatriceIntera(int[][] m)`** che converte una matrice di interi in una matrice di stringhe.