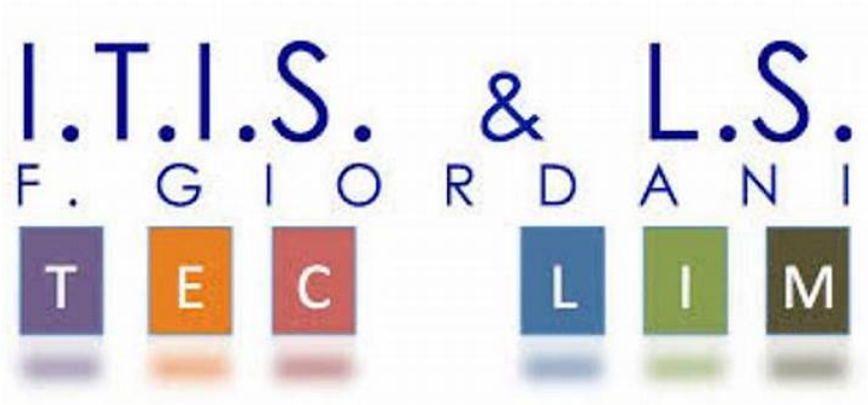


ITIS-LS "Francesco Giordani" Caserta

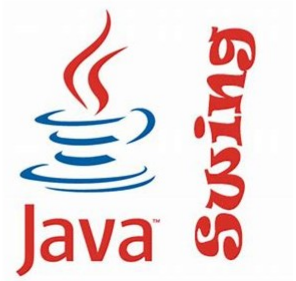
prof. Ennio Ranucci

a.s. 2019-2020

Visual java in ambiente Eclipse e framework Swing

Eclipse Version: 2020-03 (4.15.0)

Framework è una piattaforma che funge da strato intermedio tra un sistema operativo e il software che lo utilizza



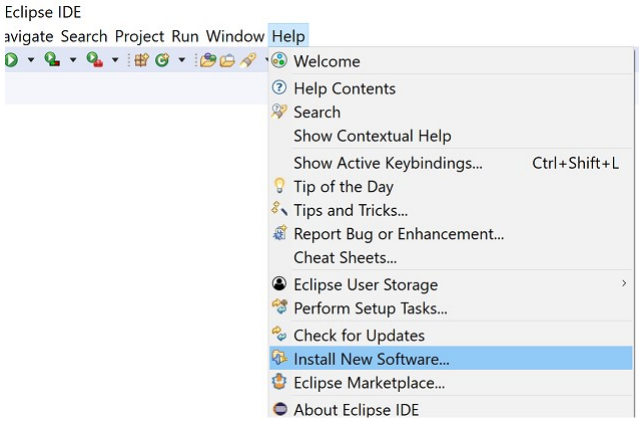
Per installare Eclipse 2020:

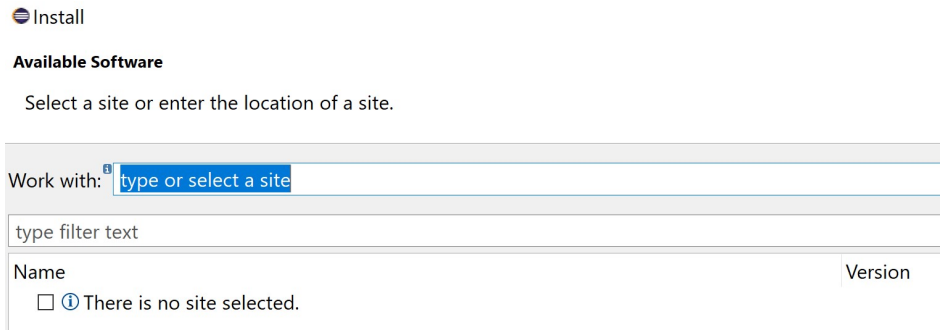
https://www.eclipse.org/downloads/packages/installer

Eclipse è un ambiente di sviluppo integrato per applicazioni open source e multi-piattaforma. Funziona principalmente come piattaforma di programmazione e può compilare ed eseguire il debug per molti linguaggi di programmazione diversi: benché noto per la programmazione in Java, la sua modularità consente di utilizzarlo per la programmazione in C, Python e molti altri.

Per installare WindowBuilder:

WindowBuilder è stato creato come plug-in per Eclipse





Copiare il link di windows Builder dalla pagina: https://www.eclipse.org/windowbuilder/download.php

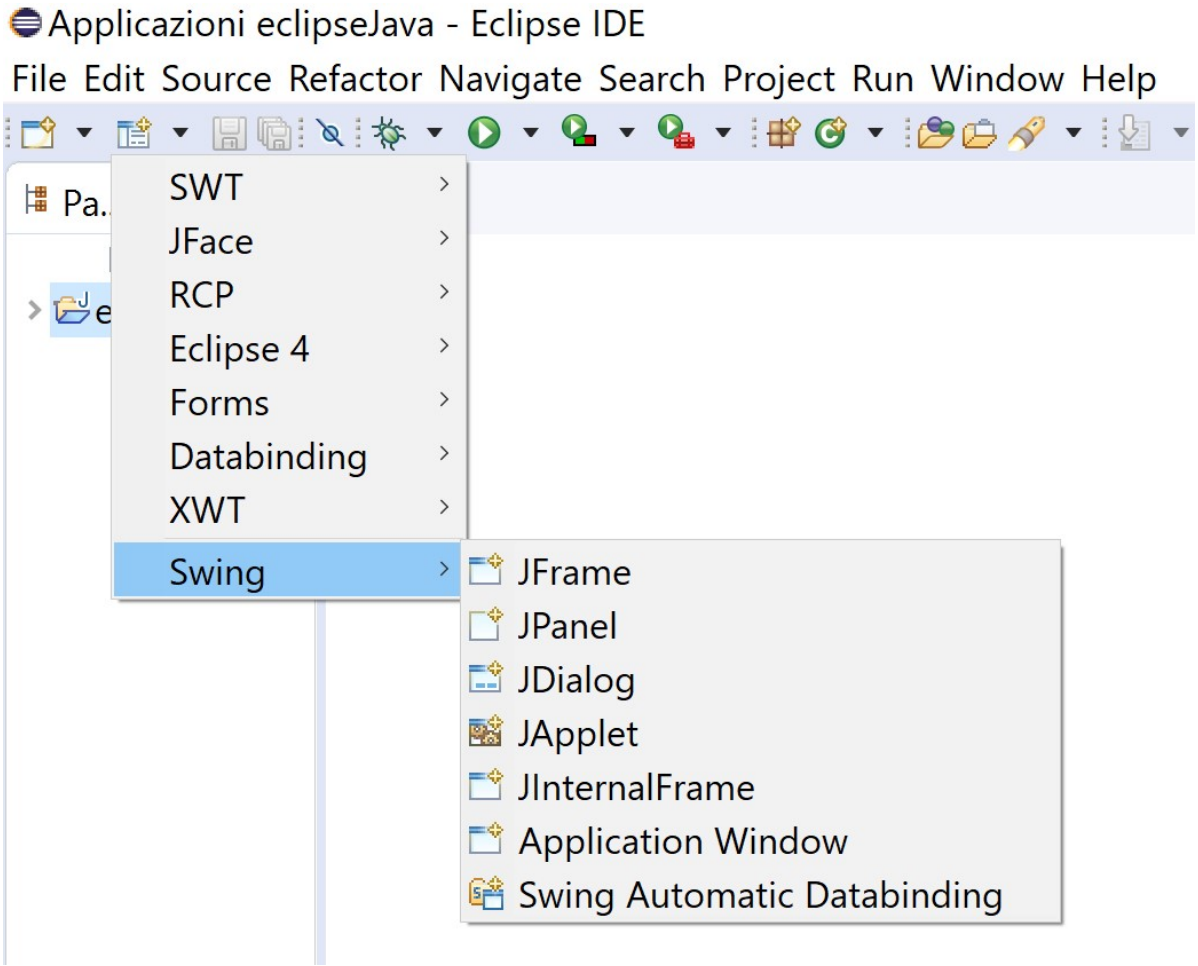
Incollare nella riga “Work with”: il link di windows Builder:

http://download.eclipse.org/windowbuilder/latest/ (latest version update site)

WindowBuilder è composto da SWT Designer e Swing Designer e semplifica la creazione di applicazioni GUI Java senza perdere molto tempo a scrivere codice. Consente di aggiungere facilmente controlli usando il trascinamento della selezione, di aggiungere gestori di eventi ai controlli, di modificare varie proprietà dei controlli utilizzando un editor di proprietà, ecc.
Il codice generato non richiede ulteriori librerie personalizzate per la compilazione e l'esecuzione: tutto il codice generato può essere utilizzato senza aver installato WindowBuilder Pro.

Per creare un nuovo pacchetto java:

Scegliere File-> new-> Java Project->Inserire il nome-> Finish
Selezionare la seconda icona della toolBar->Swing->JFrame



Scegliere JFrame -> inserire il nome (ad esempio esProvaJFrame**)->Finish**
e si ottiene il seguente codice:

```
package esProva;
import java.awt.BorderLayout;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
public class esProvaJFrame extends JFrame {
	private JPanel contentPane;
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					esProvaJFrame frame = new esProvaJFrame();
		frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	/**
	 * Create the frame.
	 */
	public esProvaJFrame() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		contentPane.setLayout(new BorderLayout(0, 0));
	setContentPane(contentPane);
	}
}
```

Se si sceglie Application Window invece che JFrame viene creata una classe principale che contiene un JFrame.
Viene generato il seguente codice:

```
package esProva;
import java.awt.EventQueue;
import javax.swing.JFrame;
public class esProvaApplicationWindow {
	private JFrame frame;	//differenza con la procedura JFrame del codice precedente
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
				  esProvaApplicationWindow window = new esProvaApplicationWindow();
				  window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
	}
		});
	}
	/**
	 * Create the application.
	 */
	public esProvaApplicationWindow() {
		initialize();
	}
	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frame = new JFrame();
		frame.setBounds(100, 100, 450, 300);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
}
```

Nota 1: Tra source e design viene utilizzato un parser e un traduttore che convertono il codice in interfaccia grafica GUI e viceversa. La programmazione visuale facilita lo sviluppo dell'applicazione. L'ingegneria del software è una branca del''informatica che si occupa di facilitare la produzione del software.
L'ingegneria del software può essere considerata come una delle principali discipline dell'informatica applicata. Si concentra sulla costruzione di sistemi software efficaci utilizzando un approccio rigoroso di ingegneria. Può essere descritto come un approccio sistematico, disciplinato e quantificabile per la progettazione, lo sviluppo, il funzionamento e la manutenzione del software e lo studio di questi approcci. La necessità di creare una disciplina teorico-pratica che si occupasse in toto della realizzazione dei software nasce, intorno alla fine degli anni sessanta, dall'esigenza di sviluppare prodotti sempre più complessi ed evoluti che rispondessero alle richieste delle grandi utenze, conferendo rigore e disciplina allo stato dell'arte dello sviluppo software nelle grandi aziende.
Più precisamente dal 1950 al 1965 lo sviluppo del software era alquanto limitato: molti programmi venivano sviluppati per batch, gli informatici erano pochi ed apprendevano sul campo. Ciò che veniva sviluppato era pensato per un unico cliente, inoltre ad ogni progetto lavorava ed avrebbe lavorato una sola persona, solitamente senza scrivere alcuna documentazione del software.
Nel 1968 la conferenza NATO tenuta a Garmisch, in Germania, rende chiaro il problema rappresentato dall'incapacità di produrre nei tempi previsti software affidabile e rispondente ai requisiti. A partire dal 1972 e fino al 1988 vengono introdotte nuove tecnologie, nascono i sistemi distribuiti e si afferma la figura del progettista del sistema informatico (quello che in seguito verrà chiamato architetto del software). Il costo dell'hardware si abbassa considerevolmente e di conseguenza la tecnologia informatica comincia a diffondersi rapidamente. Il livello qualitativo del software si eleva, tuttavia il suo sviluppo è ancora limitato a progetti scientifici e militari, e solo successivamente, dopo aver affrontato una lunga fase di collaudo, il software viene introdotto nelle industrie. Organizzazioni come il Pentagono spingono fortemente lo studio di modelli che permettano di minimizzare la quantità di errori all'interno dei software.
Con l'introduzione delle tecnologie informatiche anche nel settore industriale e commerciale, a partire dal 1988, bacini di utenze non più tecniche sentono l'esigenza di informatizzare le proprie strutture. In questo periodo nasce la programmazione orientata agli oggetti, si tende a controllare lo sviluppo del software, cercando di sviluppare prodotti di qualità, anche a causa della concorrenza affermatasi tra le software house. Si cerca di curare al massimo l'interfaccia grafica presentata all'utente, in quanto anche il tipo di utenza è cambiato. Da queste esigenze nasce l'incontro tra i requisiti dell'azienda cliente e le funzionalità che il programmatore deve realizzare.
Si sviluppa un concetto analogo alle ottimizzazioni da catena di montaggio nelle industrie del XX secolo, che avevano similmente stravolto il modo di produrre apparecchiature meccaniche (dalla produzione artigianale a quella industriale). Si cerca cioè di identificare i punti focali che devono governare la realizzazione di un buon prodotto software ma soprattutto si cerca di definire formalmente cosa possa descrivere un buon prodotto software.

Nota 2: Aggiungere controlli alla form e poi controllare il codice risultante dalla traduzione: i controlli sono stati definiti all'interno del costruttore. Dopo aver definito un evento, aggiungendo ulteriori controlli risultano definiti all'esterno del costruttore.

Per aprire un pacchetto java già creato:

Scegliere File-> import->Existing Project into workSpace->Next->Spuntare il pacchetto prescelto-> Finish
Cliccare sulla cartella con il nome del pacchetto(colonna sx)->Aprire le sottocartelle fino al file.java->pulsande dx del mouse->Open With->WindowBuilder Editor.

```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2019/2020
Classe 4^ sez.B spec. Informatica
Data: 15/05/20209
Numero es: 0
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Eclipse 2020 3
Obiettivo didattico: Utilizzare l'ambiente visuale di java
Obiettivo del programma: Inserire un pulsante, due etichette e una casella di testo. L'evento click del pulsante scrive buongiorno seguito dal nome acquisito dalla casella di testo;
*/
package es0;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import java.awt.Font;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.Color;
import java.awt.event.ActionListener;
```

```
import java.awt.event.ActionEvent;

public class es0 extends JFrame {

	private JPanel contentPane;
	private JTextField txtNome;
	private JLabel lblSaluto;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					es0 frame = new es0();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public es0() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 567, 514);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
```

```
		setContentPane(contentPane);
		contentPane.setLayout(null);
		JButton btnSaluta = new JButton("Saluta");
		btnSaluta.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				lblSaluto.setText("Buongiorno"+" "+txtNome.getText() );
			}
		});
		btnSaluta.setFont(new Font("Tahoma", Font.PLAIN, 30));
		btnSaluta.setBounds(317, 329, 203, 90);
		contentPane.add(btnSaluta);

		JLabel lblInserisciNome = new JLabel("Inserisci Nome");
		lblInserisciNome.setFont(new Font("Tahoma", Font.PLAIN, 30));
		lblInserisciNome.setBounds(21, 119, 209, 66);
		contentPane.add(lblInserisciNome);

		lblSaluto = new JLabel("");
		lblSaluto.setForeground(new Color(30, 144, 255));
		lblSaluto.setFont(new Font("Tahoma", Font.PLAIN, 40));
		lblSaluto.setBounds(45, 210, 475, 90);
		contentPane.add(lblSaluto);
		txtNome = new JTextField();
		txtNome.setFont(new Font("Tahoma", Font.PLAIN, 30));
		txtNome.setBounds(234, 117, 247, 72);
		contentPane.add(txtNome);
		txtNome.setColumns(10);
		}
}
```

Finestre modali e non modali

Ogni istanza di QWidget può essere mostrata a schermo invocando semplicemente il metodo show(). In questo caso, il **window manager** del sistema operativo prende in carico la richiesta di disegno aggiungendo il bordo ed i pulsanti della finestra prima di mostrarla a schermo come finestra non modale.
L'uso di una **finestra non modale** consente di preservare la reattività della finestra principale dell'applicazione, permettendo all'utente di interagire allo stesso tempo con entrambe. In altre parole, l'invocazione del metodo show() non è bloccante per l'event loop della finestra principale. A volte, però, questo comportamento deve essere modificato per forzare l'utente ad interagire momentaneamente soltanto con una finestra secondaria per motivi specifici. Per questa eventualità, il framework Qt predispone la classe QDialog che estende QWidget con il metodo exec(). Invocare exec() su una istanza di tipo QDialog o di una sua derivata consente di mostrarla come finestre modali.

```
String input = JOptionPane.showInputDialog("Messaggio");
```

Il metodo restituisce la stringa (come Object e quindi su cui è necessario fare il cast) inserita nellInputDialog oppure null se è stato premuto il bottone annulla.

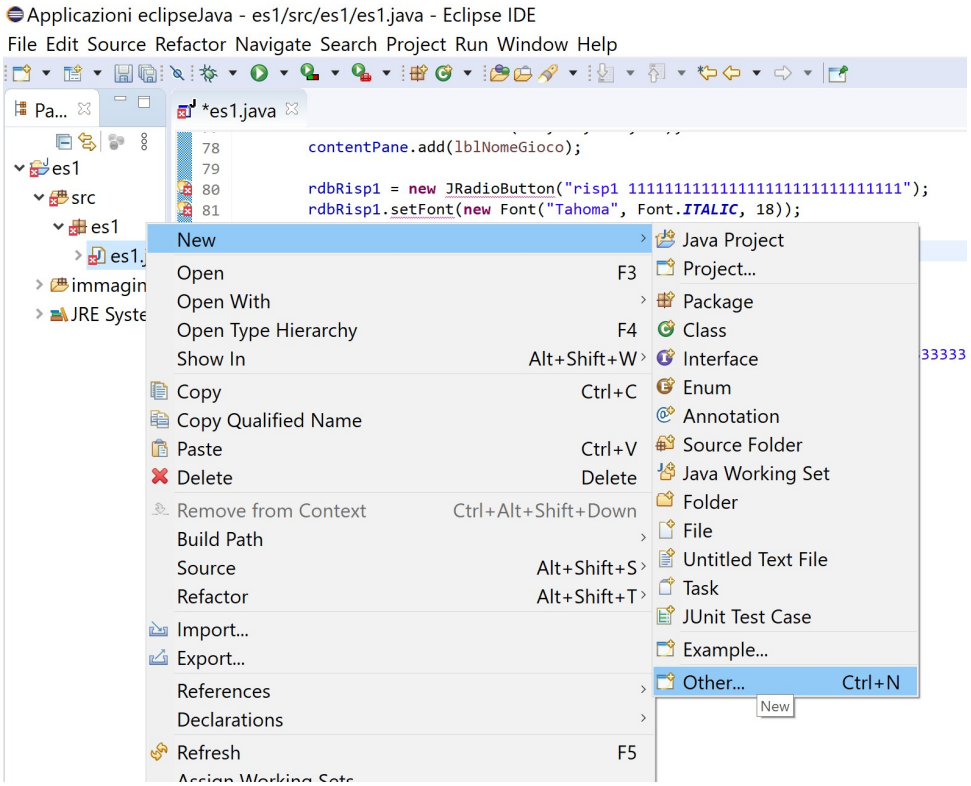
```
JOptionPane.showMessageDialog(null,"Ciao 4 Bnf", "Titolo finestra",
JOptionPane.INFORMATION_MESSAGE);
```

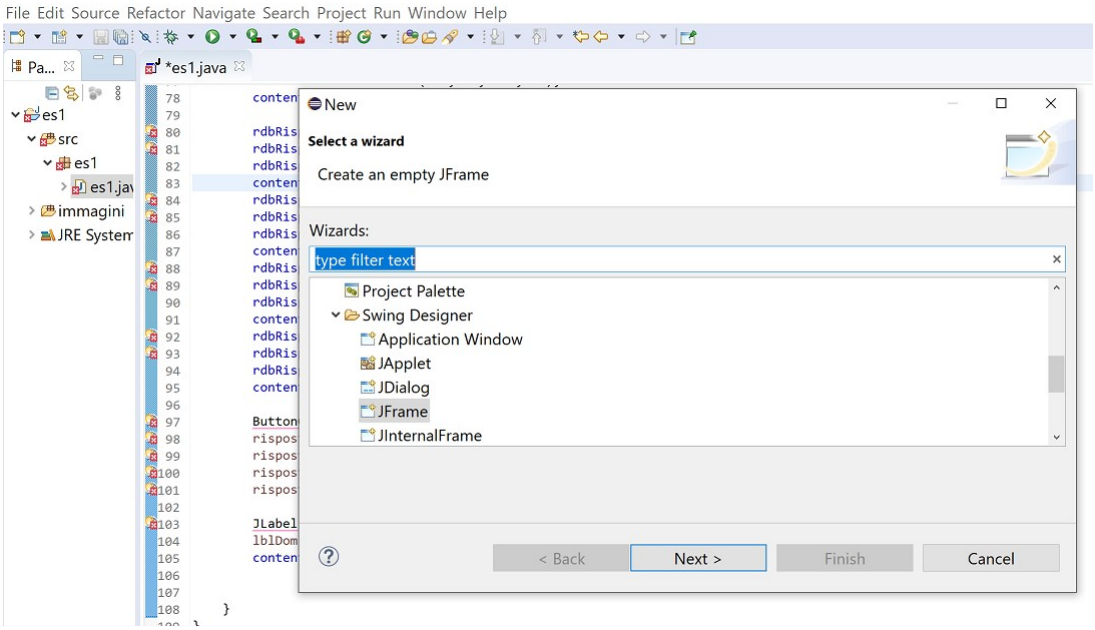
I metodi appena descritti richiedono di specificare i seguenti parametri:

parent: Questo parametro serve a specificare il frame principale; esso verrà bloccato fino al termine dell'interazione. Ponendo a null questo parametro la finestra verrà visualizzata al centro dello schermo e risulterà indipendente dal resto dell'applicazione(finestra modale).
Message: Questo campo permette di specificare una stringa da visualizzare come messaggio oppure, in alternativa, una qualunque sottoclasse di Component.
title: Questo parametro è utilizzato per specificare il titolo della finestra modale.
messageType: Attraverso questo parametro è possibile influenzare l'aspetto complessivo della finestra, per quanto riguarda il tipo di icona e il layout. Il parametro può assumere uno dei seguenti valori:
JOptionPane.ERROR_MESSAGE
JOptionPane.INFORMATION_MESSAGE
JOptionPane.WARNING_MESSAGE
JOptionPane.QUESTION_MESSAGE
JOptionPane.PLAIN_MESSAGE
optionType: I Confirm Dialog possono presentare diversi gruppi di opzioni per scegliere i bottoni da visualizzare nella finestra modale:
JOptionPane.YES_NO_OPTION
JOptionPane.YES_NO_CANCEL_OPTION
JOptionPane.OK_CANCEL_OPTION

icon: Ogni istanza della classe Icon rappresenta un'icona. In questo caso l'oggetto passato sarà visualizzato come icona di un Input Dialog. Nella maggior parte dei casi questo parametro può essere messo a null in modo tale che sia scelta l'icona di default.
SelectedValues: Questa parametro permette di impostare i possibili valori, tipicamente stringhe, che possono selezionati attraverso l'Input Dialog. I valori selezionabili sono inseriti in un ComboBox in modo tale che nessun altro valore è ammissibile. Se questo parametro viene messo a null allora il ComboBox è sostituito da un campo di testo e qualsiasi valore è ammesso.
initialValue: Questo parametro permette di stabilire il valore, tipicamente una stringa, che di default viene visualizzato.

Finestre multiple non modali





/*

ITIS-LS F.Giordani Caserta

Anno scolastico 2019/2020

Classe 4^ sez.B spec. Informatica

Data: 19/05/2020

Numero es: 1

Versione:1.0

Programmatore/i: Lezione collettiva

Sistema Operativo:Windows 10

Compilatore/Interprete: Eclipse 2020 3

Obiettivo didattico: Utilizzare l'ambiente visuale di java

Obiettivo del programma: Dare agli allievi gli strumenti per realizzare un gioco_apprendimento con il format prof Ranucci:
Il format prevede un questionario con una immagine, una domanda, tre distrattore ed una risposta corretta.Ad ogni risposta non corretta, il giocatore perde 1 punto dei 3 disponibili. Quando risponde correttamente viene mostrata una finestra con un approfondimento relativo all'argomento della domanda). In una zona riservata viene mostrato il numero della domanda e il numero totale delle domande, il punteggio della domanda, il punteggio parziale e, alla fine, il punteggio finale raggiunto.

*/

```
package es1;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import java.awt.Font;
import java.awt.Color;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JOptionPane;

public class es1 extends JFrame {

	private JPanel contentPane;
	private JRadioButton rdbRisp1;
	private JRadioButton rdbRisp2;
	private JRadioButton rdbRisp3;
	private JRadioButton rdbRisp4;
	private int corretta;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					es1 frame = new es1();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public es1() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 732, 603);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);
		corretta=2;
		JButton btnConferma = new JButton("Conferma");
		btnConferma.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent arg0) {
			if (rdbRisp1.isSelected() && corretta==1)
{System.out.println("risp1");}else{rdbRisp1.setEnabled(false);}   ;
```

```
                    if (rdbRisp2.isSelected())  System.out.println("risp2");
                    if (rdbRisp3.isSelected())  System.out.println("risp3");
                    if (rdbRisp4.isSelected())  System.out.println("risp4");
                    String input = JOptionPane.showInputDialog("Messaggio");
                    JOptionPane.showMessageDialog(null, "Ciao 4 B inf", "Titolo
finestra", JOptionPane.INFORMATION_MESSAGE);
                    finestraApprofondimentoModaleJDialog winModaleJDialog= new
finestraApprofondimentoModaleJDialog();
                   winModaleJDialog.setModal(true);
                    winModaleJDialog.show();
                    //finestraApprofondimentoModaleJFrame winModaleJFrame= new
finestraApprofondimentoModaleJFrame();
                    //winModaleJFrame.show();
                    }
            });
            btnConferma.setBounds(448, 124, 172, 48);
            contentPane.add(btnConferma);
            JLabel lblImmagine = new JLabel("");
            //file->new->source folder
            lblImmagine.setIcon(new
ImageIcon("C:\\Users\\ennio\\Desktop\\Scuola\\eclipse\\Applicazioni
eclipseJava\\es1\\immagini\\Campanile-visto-dalla-strada-per-San-Pietro-ad-
Montes.jpg"));
            lblImmagine.setBounds(0, 0, 395, 292);
            contentPane.add(lblImmagine);
            JLabel lblNomeGioco = new JLabel("Apprendere giocando");
            lblNomeGioco.setForeground(new Color(30, 144, 255));
            lblNomeGioco.setFont(new Font("Times New Roman", Font.PLAIN, 30));
            lblNomeGioco.setBounds(398, 35, 269, 54);
            contentPane.add(lblNomeGioco);

            rdbRisp1 = new JRadioButton("risp1 11111111111111111111111111111");
            rdbRisp1.setFont(new Font("Tahoma", Font.ITALIC, 18));
            rdbRisp1.setBounds(16, 350, 663, 30);
            contentPane.add(rdbRisp1);
            rdbRisp2 = new JRadioButton("risp2 22222222222222222222222222");
            rdbRisp2.setFont(new Font("Tahoma", Font.ITALIC, 18));
            rdbRisp2.setBounds(16, 380, 663, 30);
            contentPane.add(rdbRisp2);
            rdbRisp3 = new JRadioButton("risp3
333333333333333333333333333333333333333333333333333333333");
            rdbRisp3.setFont(new Font("Tahoma", Font.ITALIC, 18));
            rdbRisp3.setBounds(16, 410, 663, 30);
            contentPane.add(rdbRisp3);
            rdbRisp4 = new JRadioButton("risp4 ");
            rdbRisp4.setFont(new Font("Tahoma", Font.ITALIC, 18));
        rdbRisp4.setBounds(16, 440, 663, 30);
      contentPane.add(rdbRisp4);

      ButtonGroup risposteGruppo = new ButtonGroup();
            risposteGruppo.add(rdbRisp1);
            risposteGruppo.add(rdbRisp2);
            risposteGruppo.add(rdbRisp3);
            risposteGruppo.add(rdbRisp4);

            JLabel lblDom = new JLabel("domanda aaaaaaaaaaaaaaaa");
            lblDom.setBounds(31, 299, 636, 26);
            contentPane.add(lblDom);
      }
```

```
}
package es1;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

public class finestraApprofondimentoModaleJFrame extends JFrame {

	private JPanel contentPane;

	/**

	 * Launch the application.

	 */

	public static void main(String[] args) {

		EventQueue.invokeLater(new Runnable() {

			public void run() {

				try {

					finestraApprofondimentoModaleJFrame frame = new
finestraApprofondimentoModaleJFrame();

					frame.setVisible(true);

				} catch (Exception e) {

					e.printStackTrace();

				}

			}

		});

	}
```

```
	/**
	 * Create the frame.
	 */
	public finestraApprofondimentoModaleJFrame() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("Finestra JFrame Modale");
		lblNewLabel.setBounds(52, 55, 288, 48);
		contentPane.add(lblNewLabel);
	}
}
```

```
package es1;

import java.awt.BorderLayout;

import java.awt.FlowLayout;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.ImageIcon;

import javax.swing.SwingConstants;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class finestraApprofondimentoModaleJDialog extends JDialog {

	private final JPanel contentPanel = new JPanel();


	/**

	 * Launch the application.

	 */

	public static void main(String[] args) {

		try {

			finestraApprofondimentoModaleJDialog dialog = new
finestraApprofondimentoModaleJDialog();

			dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);

			dialog.setVisible(true);

		} catch (Exception e) {

			e.printStackTrace();
```

```
		}
	}

	/**
	 * Create the dialog.
	 */
	public finestraApprofondimentoModaleJDialog() {
		setBounds(100, 100, 579, 485);
		getContentPane().setLayout(new BorderLayout());
		contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
		getContentPane().add(contentPanel, BorderLayout.CENTER);
		contentPanel.setLayout(null);
		{
			JLabel lblNewLabel = new JLabel("New label");
			lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
			lblNewLabel.setIcon(new
ImageIcon("C:\\Users\\ennio\\Desktop\\Scuola\\eclipse\\Applicazioni
eclipseJava\\es1\\immagini\\Casertavecchia.jpg"));
			lblNewLabel.setBounds(37, 21, 258, 179);
			contentPanel.add(lblNewLabel);
		}
		{
			JLabel lblNewLabel_1 = new JLabel("Finestra di Dialogo Modale");
			lblNewLabel_1.setBounds(82, 246, 281, 39);
			contentPanel.add(lblNewLabel_1);
		}
		{
```

```
                JPanel buttonPane = new JPanel();
                buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
                getContentPane().add(buttonPane, BorderLayout.SOUTH);
                {
                        JButton okButton = new JButton("OK");
                        okButton.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent arg0) {
                                        dispose();
                                }
                        });
                        okButton.setActionCommand("OK");
                        buttonPane.add(okButton);
                        getRootPane().setDefaultButton(okButton);
                }
                {
                        JButton cancelButton = new JButton("Cancel");
                        cancelButton.setActionCommand("Cancel");
                        buttonPane.add(cancelButton);
                }
        }
    }
}
```