```
======================Cliente.java=====================================
==========


public class Cliente extends Persona {

	private String codicefiscale;

	public Cliente(String cognome, String nome, String indirizzo,
String codicefiscale) {
		super(cognome, nome, indirizzo);
		this.codicefiscale = codicefiscale;
	}

	@Override
	public String toString() {
		return "Cliente ["+super.toString()+" codicefiscale=" +
codicefiscale + "]";
	}

}


==========================Fornitore.java===================================
==================


public class Fornitore extends Persona {

	private String partitaIVA;

	public Fornitore(String cognome, String nome, String indirizzo,
String partitaIVA) {
		super(cognome, nome, indirizzo);
		this.partitaIVA = partitaIVA;
	}

	@Override
	public String toString() {
		return "Fornitore ["+super.toString()+" partitaIVA=" +
partitaIVA + "]";
	}


}



=======================GUIPersone.java===================================
=====




import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.util.List;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class GUIPersone  extends JFrame implements Lockable{
      private boolean locked = true;
      private JButton buttonSubmit, buttonAddCliente, buttonAddFornitore;
      private JPanel centralPanel, cSxPanel, cDxPanel, ccSxPanel,
ccDxPanel;
      private JLabel nomeLab1, nomeLab2, cognomeLab1, cognomeLab2,
indirizzoLab1, indirizzoLab2, codFiscLab, pIvaLab;
      private JTextField nomeField1, nomeField2, cognomeField1,
cognomeField2, indirizzoField1, indirizzoField2,
                  codFiscField, pIvaField;

      public GUIPersone(List<Cliente> dbCliente, List<Fornitore>
dbFornitore) {
            super();
            // pannello sinistro
            cSxPanel = new JPanel(new BorderLayout());
            ccSxPanel = new JPanel(new GridLayout(4, 2));

            nomeLab1 = new JLabel("Nome Cliente");
            cognomeLab1 = new JLabel("Cognome Cliente");
            indirizzoLab1 = new JLabel("Indirizzo Cliente");
            codFiscLab = new JLabel("Codice Fiscale Cliente");

            nomeField1 = new JTextField(10);
            cognomeField1 = new JTextField(10);
            indirizzoField1 = new JTextField(10);
            codFiscField = new JTextField(10);

            ccSxPanel.add(nomeLab1);
            ccSxPanel.add(nomeField1);
            ccSxPanel.add(cognomeLab1);
            ccSxPanel.add(cognomeField1);
            ccSxPanel.add(indirizzoLab1);
            ccSxPanel.add(indirizzoField1);
            ccSxPanel.add(codFiscLab);
            ccSxPanel.add(codFiscField);

            buttonAddCliente = new JButton("Aggiungi Cliente");
            JPanel butCliContainer = new JPanel();
            butCliContainer.add(buttonAddCliente);

            cSxPanel.add(ccSxPanel, BorderLayout.CENTER);
            cSxPanel.add(butCliContainer, BorderLayout.SOUTH);
            cSxPanel.setBorder(BorderFactory.createLineBorder(Color.BLACK)
);

            // pannello Destro
            cDxPanel = new JPanel(new BorderLayout());
            ccDxPanel = new JPanel(new GridLayout(4, 2));

            nomeLab2 = new JLabel("Nome Fornitore");
```

```
            cognomeLab2 = new JLabel("Cognome Fornitore");
            indirizzoLab2 = new JLabel("Indirizzo Fornitore");
            pIvaLab = new JLabel("Partita Iva Fornitore");

            nomeField2 = new JTextField(10);
            cognomeField2 = new JTextField(10);
            indirizzoField2 = new JTextField(10);
            pIvaField = new JTextField(10);

            ccDxPanel.add(nomeLab2);
            ccDxPanel.add(nomeField2);
            ccDxPanel.add(cognomeLab2);
            ccDxPanel.add(cognomeField2);
            ccDxPanel.add(indirizzoLab2);
            ccDxPanel.add(indirizzoField2);
            ccDxPanel.add(pIvaLab);
            ccDxPanel.add(pIvaField);


            buttonAddFornitore = new JButton("Aggiungi Fornitore");
            JPanel butFornContainer = new JPanel();
            butFornContainer.add(buttonAddFornitore);

            cDxPanel.add(ccDxPanel, BorderLayout.CENTER);
            cDxPanel.add(butFornContainer, BorderLayout.SOUTH);
            cDxPanel.setBorder(BorderFactory.createLineBorder(Color.BLACK)
);

            centralPanel = new JPanel(new GridLayout(1, 2));
            centralPanel.add(cSxPanel);
            centralPanel.add(cDxPanel);

      centralPanel.setBorder(BorderFactory.createLineBorder(Color.BLACK)
);

            buttonSubmit = new JButton("Aggiunta Terminata");
            Container mainWindow = getContentPane();
            mainWindow.add(centralPanel, BorderLayout.CENTER);

            JPanel subContainer = new JPanel();
            subContainer.add(buttonSubmit);
            mainWindow.add(subContainer, BorderLayout.SOUTH);

            setSize(800, 200);
            setVisible(true);
            setDefaultCloseOperation(EXIT_ON_CLOSE);

            buttonAddCliente.setActionCommand("cliente");
            buttonAddFornitore.setActionCommand("fornitore");
            buttonSubmit.setActionCommand("submit");

            ListnerCliente listenerCliente = new ListnerCliente(dbCliente,
nomeField1, cognomeField1, indirizzoField1, codFiscField);
            buttonAddCliente.addActionListener(listenerCliente);

            ListnerFornitore lf = new ListnerFornitore(dbFornitore,
nomeField2, cognomeField2, indirizzoField2, pIvaField);
            buttonAddFornitore.addActionListener(lf);

            ListnerChangeStatus lcs = new ListnerChangeStatus(this);
```

```
            buttonSubmit.addActionListener(lcs);

      }

      @Override
      public boolean isLocked() {
            return locked;
      }

      @Override
      public void unlock(){
            this.locked = false;
      }
}

class ListnerChangeStatus implements ActionListener{
      private GUIPersone gui;
      public ListnerChangeStatus(GUIPersone gui) {
            this.gui = gui;
      }
      @Override
      public void actionPerformed(ActionEvent e) {

            JButton source = (JButton)e.getSource();
            if(source.getActionCommand().equals("submit")){
                  gui.unlock();
//                gui.setVisible(false);
                  gui.dispose();
            }

      }

}
```

==========ListenerCliente.java==========

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

import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class ListnerCliente extends ListnerPersona {
      private JTextField codiceFiscale;
      private List<Cliente> dbCliente;
      public ListnerCliente(List<Cliente> dbCliente, JTextField
nomeField, JTextField cognomeField, JTextField indirizzoField,
                  JTextField codiceFiscale) {
            super(nomeField, cognomeField, indirizzoField);
            this.codiceFiscale=codiceFiscale;
            this.dbCliente = dbCliente;
      }
```

```
	@Override
	public void actionPerformed(ActionEvent arg) {
		JButton source = (JButton)arg.getSource();
		if(source.getActionCommand().equals("cliente")){
			String nome = super.getNomeField().getText();
			String cognome = super.getCognomeField().getText();
			String indirizzo = super.getIndirizzoField().getText();
			String cf = codiceFiscale.getText().toUpperCase();
			if(nome.equals("")||
cognome.equals("")||indirizzo.equals("")||cf.equals("")||!controlloCF(cf)
){
				JOptionPane.showMessageDialog(source, "Errore
Inserimento", "Errore Inserimento", JOptionPane.ERROR_MESSAGE);
				return;
			} else {
				dbCliente.add(new Cliente(cognome, nome,
indirizzo, cf));
				JOptionPane.showMessageDialog(source, "Inserimento
OK", "Inserimento OK", JOptionPane.INFORMATION_MESSAGE);
				super.getNomeField().setText("");
				super.getCognomeField().setText("");
				super.getIndirizzoField().setText("");
				codiceFiscale.setText("");
			}

		}
	}

	private boolean controlloCF(String cf) {
		boolean check = true;
		if(cf.length() != 16)
			return false;
		check = check && isAlphabetical(cf.substring(0, 6));
		check = check && isNumeric(cf.substring(6, 8));
		check = check && isAlphabetical(cf.substring(8,9));
		check = check && isNumeric(cf.substring(9,11));
		check = check && isAlphabetical(cf.substring(11,12));
		check = check && isNumeric(cf.substring(12,15));
		check = check && isAlphabetical(cf.substring(15,16));

		return check;
	}

	private boolean isNumeric(String x){
		try{
			Long.parseLong(x);
			return true;
		}catch(NumberFormatException e){
			return false;
		}
	}

	private boolean isAlphabetical(String x){
		for(int i = 0; i < x.length();i++){
			char c = x.charAt(i);
			int cValue = (int)c;
			if(cValue<65||cValue>90){
				return false;
			}
```

```
            }
            return true;
      }


}
```

==========ListenerFornitore.java==========

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

import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class ListnerFornitore extends ListnerPersona {

      private JTextField pIva;
      private List<Fornitore> dbFornitore;

      public ListnerFornitore(List<Fornitore> dbFornitore, JTextField
nomeField, JTextField cognomeField, JTextField indirizzoField, JTextField
pIva) {
            super(nomeField, cognomeField, indirizzoField);
            this.pIva = pIva;
            this.dbFornitore = dbFornitore;
      }

      @Override
      public void actionPerformed(ActionEvent arg) {
            JButton source = (JButton)arg.getSource();
            if(source.getActionCommand().equals("fornitore")){
                  String nome = super.getNomeField().getText();
                  String cognome = super.getCognomeField().getText();
                  String indirizzo = super.getIndirizzoField().getText();
                  String pI = pIva.getText().toUpperCase();
                  if(nome.equals("")||
cognome.equals("")||indirizzo.equals("")||pI.equals("")||!controlloPIva(p
I)){
                        JOptionPane.showMessageDialog(source, "Errore
Inserimento", "Errore Inserimento", JOptionPane.ERROR_MESSAGE);
                        return;
                  } else {
                        dbFornitore.add(new Fornitore(cognome, nome,
indirizzo, pI));
                        JOptionPane.showMessageDialog(source, "Inserimento
OK", "Inserimento OK", JOptionPane.INFORMATION_MESSAGE);
                        super.getNomeField().setText("");
                        super.getCognomeField().setText("");
                        super.getIndirizzoField().setText("");
                        pIva.setText("");
```

```
                }

            }

      }

      private boolean controlloPIva(String pI) {
            if(pI.length() != 11)
                  return false;
            try{
                  Long.parseLong(pI);
                  return true;
            }catch(NumberFormatException e){
                  return false;
            }
      }

}
```

==========ListenerPersona.java==========

```
import java.awt.event.ActionListener;

import javax.swing.JTextField;

public abstract class ListnerPersona implements ActionListener {

      private JTextField nomeField, cognomeField, indirizzoField;

      public ListnerPersona( JTextField nomeField, JTextField
cognomeField, JTextField indirizzoField) {
            this.nomeField = nomeField;
            this.cognomeField = cognomeField;
            this.indirizzoField = indirizzoField;
      }

      protected JTextField getNomeField() {
            return nomeField;
      }

      protected JTextField getCognomeField() {
            return cognomeField;
      }

      protected JTextField getIndirizzoField() {
            return indirizzoField;
      }

}
```

==========Lockable.java==========

```
public interface Lockable {

	public boolean isLocked();
	public void unlock();

}
```

==========Main.java==========

```
import java.util.*;

public class Main {

	final static String[] cognomi = { "Rossi", "Verdi", "Bianchi",
"Neri", "Gialli", "Viola" };
	final static String[] nomi = { "Maria", "Giulia", "Antonio",
"Carlo", "Luisa", "Anna" };

	static List<Cliente> dbClienti = new LinkedList<Cliente>();
	static List<Fornitore> dbFornitori = new LinkedList<Fornitore>();
	static List<Prodotto> dbProdotti = new LinkedList<Prodotto>();
	static List<Ordine> dbOrdini = new LinkedList<Ordine>();

	public static Cliente newRandomCliente() {
		int kc = (int) (Math.random() * cognomi.length);
		int kn = (int) (Math.random() * nomi.length);
		return new Cliente(nomi[kn], cognomi[kc], "", "" +
nomi[kn].hashCode());
	}

	public static Fornitore newRandomFornitore() {
		int kc = (int) (Math.random() * cognomi.length);
		int kn = (int) (Math.random() * nomi.length);
		return new Fornitore(nomi[kn], cognomi[kc], "", "" +
cognomi[kn].hashCode());
	}

	public static Cliente chooseRandomCliente() {
		int k = (int) (Math.random() * dbClienti.size());
		return dbClienti.get(k);
	}

	public static Fornitore chooseRandomFornitore() {
		int k = (int) (Math.random() * dbFornitori.size());
		return dbFornitori.get(k);
	}

	public static Prodotto newRandomProdotto(int n, int q) {
		String codice = String.format("A%03d", n);
		double pr = Math.random() * 100;
		return new Prodotto(codice, "", chooseRandomFornitore(), pr,
q);
	}
```

```
public static Prodotto chooseRandomProdotto() {
	int k = (int) (Math.random() * dbProdotti.size());
	return dbProdotti.get(k);
}

public static Ordine newRandomOrdine(int n) {
	Ordine ord = new Ordine(n, chooseRandomCliente());
	int k = (int) (Math.random() * 10);
	for (int i = 0; i < k; i++) {
		Prodotto p = chooseRandomProdotto();
		int q = (int) (Math.random() * 10);
		ord.addProdottoQuantita(p, q);
	}
	return ord;
}

public static Ordine chooseRandomOrdine() {
	int k = (int) (Math.random() * dbOrdini.size());
	return dbOrdini.get(k);
}

public static void creaDB() {

	GUIPersone gui= new GUIPersone(dbClienti, dbFornitori);
	while(gui.isLocked()){
		try {
			Thread.sleep(50);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}

	System.err.println(dbClienti.size());
	System.err.println(dbFornitori.size());

	for (int i = 0; i < 20; i++) {
		int q = (int) (Math.random() * 10);
		dbProdotti.add(newRandomProdotto(i, q));
	}
	for (int i = 0; i < 20; i++) {
		dbOrdini.add(newRandomOrdine(i));
	}
}

public static void main(String[] args) {
	creaDB();

	System.out.println("=== LISTA TUTTI PRODOTTI ===");
	Iterator<Prodotto> ip = dbProdotti.iterator();
	while (ip.hasNext()) {
		System.out.println(ip.next());
	}

	System.out.println("=== TOTALI DEGLI ORDINI ===");
	Iterator<Ordine> it = dbOrdini.iterator();
	while (it.hasNext()) {
		Ordine ord = it.next();
		double ptot = Operazioni.totaleOrdine(ord);
		System.out.println("Ordine: " + ord);
```

```
                System.out.println("Totale: " + ptot);
            }

            System.out.println("=== PRODOTTI NON DISPONIBILI NEGLI ORDINI
===");
            it = dbOrdini.iterator();
            while (it.hasNext()) {
                Ordine ord = it.next();
                List<Prodotto> pnd =
Operazioni.prodottiNonDisponibili(ord, dbProdotti);
                System.out.println("Ordine: " + ord);
                System.out.println("Prodotti non disponibili: " + pnd);
            }
      }

}
```

==========Operazioni.java==========

```
import java.util.*;

public class Operazioni {

      public static double totaleOrdine(Ordine ord) {
            double r=0.0;
            for (CoppiaProdottoQuantita c : ord.getProdotti()) {
                r += + c.prodotto.getPrezzo() * c.quantita;
            }
            return r;
      }

      public static List<Prodotto> prodottiNonDisponibili (Ordine ord,
List<Prodotto> lp) {
            List<Prodotto> r = new LinkedList<Prodotto>();
            for (CoppiaProdottoQuantita c : ord.getProdotti()) {
                if (c.quantita > c.prodotto.getDisponibili())
                    r.add(c.prodotto);
            }
            return r;
      }

}
```

==========Ordine.java==========

```
import java.util.*;

class CoppiaProdottoQuantita {

      public Prodotto prodotto;
      public int quantita;

      public CoppiaProdottoQuantita(Prodotto prodotto, int quantita) {
```

```
            super();
            this.prodotto = prodotto;
            this.quantita = quantita;
      }

      @Override
      public int hashCode() {
            final int prime = 31;
            int result = 1;
            result = prime * result + ((prodotto == null) ? 0 :
prodotto.hashCode());
            result = prime * result + quantita;
            return result;
      }

      @Override
      public boolean equals(Object obj) {
            if (this == obj)
                  return true;
            if (obj == null)
                  return false;
            if (this.getClass() != obj.getClass())
                  return false;
            CoppiaProdottoQuantita other = (CoppiaProdottoQuantita) obj;
            if (prodotto == null) {
                  if (other.prodotto != null)
                        return false;
            } else if (!prodotto.equals(other.prodotto))
                  return false;
            if (quantita != other.quantita)
                  return false;
            return true;
      }

      @Override
      public String toString() {
            return "<" + prodotto.getCodice() + ", " + quantita + ">";
      }


}

public class Ordine {

      private final int numero;
      private Cliente cliente;
      private List<CoppiaProdottoQuantita> prodotti;

      public Ordine(int numero, Cliente cliente) {
            super();
            this.numero = numero;
            this.cliente = cliente;
            this.prodotti = new LinkedList<CoppiaProdottoQuantita>();
      }

      public int getNumero() {
            return numero;
      }

      public Cliente getCliente() {
```

```
		return cliente;
	}

	public void setCliente(Cliente cliente) {
		this.cliente = cliente;
	}

	public List<CoppiaProdottoQuantita> getProdotti() {
		return prodotti;
	}

	private CoppiaProdottoQuantita prodottoPresente(Prodotto p) {
		for (CoppiaProdottoQuantita c: prodotti) {
			if (c.prodotto.equals(p)) return c;
		}
		return null;
	}

	public void addProdottoQuantita(Prodotto p, int q) {
		CoppiaProdottoQuantita c = prodottoPresente(p);
		if (c==null)
			prodotti.add(new CoppiaProdottoQuantita(p, q));
		else
			c.quantita += q;
	}

	@Override
	public String toString() {
		return "Ordine [numero=" + numero + ", cliente=" + cliente +
", prodotti=" + prodotti + "]";
	}

}
```

==========Persona.java==========

```
public abstract class Persona {

	private final String cognome, nome;
	private String indirizzo;

	public Persona(String cognome, String nome, String indirizzo) {
		super();
		this.cognome = cognome;
		this.nome = nome;
		this.indirizzo = indirizzo;
	}

	public String getIndirizzo() {
		return indirizzo;
	}

	public void setIndirizzo(String indirizzo) {
		this.indirizzo = indirizzo;
	}

	public String getCognome() {
```

```
            return cognome;
      }

      public String getNome() {
            return nome;
      }

      @Override
      public String toString() {
            return cognome + " " + nome + ", " + indirizzo;
      }

}
```

==========Prodotto.java==========

```
public class Prodotto {

      private final String codice, descrizione;
      private Fornitore fornitore;
      private double prezzo;
      private int disponibili;

      public Prodotto(String codice, String descrizione, Fornitore
fornitore, double prezzo, int disponibili) {
            super();
            this.codice = codice;
            this.descrizione = descrizione;
            this.fornitore = fornitore;
            this.prezzo = prezzo;
            this.disponibili = disponibili;
      }

      public Fornitore getFornitore() {
            return fornitore;
      }

      public void setFornitore(Fornitore fornitore) {
            this.fornitore = fornitore;
      }

      public double getPrezzo() {
            return prezzo;
      }

      public void setPrezzo(double prezzo) {
            this.prezzo = prezzo;
      }

      public int getDisponibili() {
            return disponibili;
      }

      public void setDisponibili(int disponibili) {
            this.disponibili = disponibili;
      }
```

```
	public String getCodice() {
		return codice;
	}

	public String getDescrizione() {
		return descrizione;
	}

	@Override
	public String toString() {
		return "Prodotto [codice=" + codice + ", descrizione=" +
descrizione + ", fornitore=" + fornitore + ", prezzo="
				+ prezzo + ", disponibili=" + disponibili + "]";
	}

}
```