```java
// Esempio di programmazione funzionale con Java

/** Persona tiene alcuni dati di una persona che lavora in azienda */

public class Persona {
   private String nome;
   private int eta;
   private String ruolo;

   public Persona(String nom, int et, String ruol) {
      nome = nom;
      eta = et;
      ruolo = ruol;
   }

   public boolean giovane() {
      return eta < 28;
   }

   public int getEta() {
      return eta;
   }

   public String getNome() {
      return nome;
   }

   public String getRuolo() {
      return ruolo;
   }

   public boolean isRuolo(String r) {
      return ruolo.equals(r);
   }
}


/** BustaPaga calcola per ogni persona il suo costo */

public class BustaPaga implements Comparable < BustaPaga > {
   private Persona dipendente;
   private int totale;
   private int costoBase;

   public BustaPaga(Persona p, int costo) {
      dipendente = p;
      costoBase = costo;
   }

   public BustaPaga calcolaCostoBase() {
      totale = costoBase * 25;
```

```
        return this;
    }

    public int getCosto() {
        return totale;
    }

    public BustaPaga aggiungiBonus() {
        totale = (int) Math.round(totale * 1.1);
        return this;
    }

    public Persona getPersona() {
        return dipendente;
    }

    public BustaPaga stampa() {
        System.out.println(dipendente.getNome() + "\t" + totale);
        return this;
    }

    @Override
    public int compareTo(BustaPaga b) {
        return dipendente.getNome().compareTo(b.getPersona().getNome());
    }
}
```

```
import java.util.Comparator;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

/** ListaPersone tiene le persone che compongono il team di sviluppo
 * e ne determina il costo  */
public class ListaPersone {
    private List < Persona > team = List.of(new Persona("Kent Gray", 29, "CTO"),
            new Persona("Luigi Rossi", 25, "Programmer"),
            new Persona("Al White", 25, "Programmer"),
            new Persona("Andrea Verdi", 26, "GroupLeader"),
            new Persona("Joe Black", 26, "Programmer"),
            new Persona("Jill Purple", 26, "Tester"),
            new Persona("Bill Green", 26, "Analyst"),
            new Persona("Luc Brown", 26, "Tester"));

    private Map < String, Integer > roleCost = Map.of("Programmer", 50, "GroupLea
            "CTO", 65, "Tester", 56, "Analyst", 54);

    private List < BustaPaga > pagamenti;

    /** per le persone del team calcola l'ammontare da pagare */
```

```
    public void genListaPagamenti() {
        pagamenti = team.stream()
                .map(pers -> new BustaPaga(pers, roleCost.getOrDefault(pers.getRuolo
                .map(busta -> busta.calcolaCostoBase())
                .map(busta -> busta.aggiungiBonus())
                .sorted()
                .collect(Collectors.toList());
    }

    /** calcola il totale pagamenti e stampa nome persona del team e costo */
    public int stampaCalcolaSomma() {
        return pagamenti.stream()
                .map(BustaPaga::stampa)
                .mapToInt(BustaPaga::getCosto)
                .sum();
    }

    /**  stampa ciascun ruolo e l'elenco delle persone in quel ruolo */
    public void stampaRuoliPersone() {
        roleCost.keySet()
                .stream()
                .sorted()
                .peek(role -> System.out.print("\n" + role+": "))
                .forEach(role -> team.stream()
                        .filter(pers -> pers.getRuolo().equals(role))
                        .sorted(Comparator.comparing(Persona::getNome))
                        .forEach(pers -> System.out.print(pers.getNome() + ", ")));
    }
}



public class MainTeam {

    public static void main(String[] args) {
        ListaPersone persone = new ListaPersone();

        persone.genListaPagamenti();
        int somma = persone.stampaCalcolaSomma();
        System.out.println("Totale:\t"+somma);

        persone.stampaRuoliPersone();
        System.out.println();
    }
}



Output dell'esecuzione

Al White        1375
Andrea Verdi    1650
```

```
Bill Green      1485
Jill Purple     1540
Joe Black       1375
Kent Gray       1788
Luc Brown       1540
Luigi Rossi     1375
Totale: 12128

Analyst: Bill Green,
CTO: Kent Gray,
GroupLeader: Andrea Verdi,
Programmer: Al White, Joe Black, Luigi Rossi,
Tester: Jill Purple, Luc Brown,
```