(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- **MouseListener**
- **MouseAdapter**
- **TextListener**
- **KeyListener**
- **KeyAdapter**
- **ItemListener**
- **getKeyCode()**
- **getKeyChar()**

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Quali sono gli eventi che può generare il *mouse*?
2. Quali sono gli eventi che può generare la *tastiera*?
3. Quali sono gli eventi che può generare una *casella di testo*?
4. Quali sono gli eventi che può generare una *lista*??

B2) Competenza

1. Qual è la classe di ascolto per la *tastiera*?
2. Qual è la classe di ascolto per il *mouse*?
3. Qual è la classe di ascolto per le *caselle di testo*?
4. Qual è la classe di ascolto per le *liste*, i *menu* e le *chechbox*?
5. Come può essere *rimosso* l'ascoltatore di un evento?
6. Descrivere brevemente le *classi d'ascolto* illustrate, consultando i relativi *gestori di evento.*
7. A cosa servono gli *adapter*?

(C) ESERCIZI DI COMPRENSIONE

1. La classe di ascolto per il mouse è la …….., che consente di gestire i vari eventi sul mouse, come il ……, gli eventi di …….. e …… rispetto ad un certo controllo, gli eventi di ……. e di …… del tasto sinistro.
2. La classe di ascolto per le caselle di testo è la …….., che consente di rilevare il cambiamento del testo presente nella casella, mediante il metodo ……
3. La classe di ascolto per la tastiera è la …….., che consente di rilevare i due eventi singoli di …… e ……..di un tasto, oppure l'immissione di un carattere dalla tastiera.
4. La classe di ascolto per le liste, per le caselle di spunta, per le caselle combinate e per le voci di menu è la …….., che ha un unico metodo ascoltatore in grado di rilevare il click su uno di questi oggetti.
5. Scrivere i metodi dell'interfaccia **MouseListener**:

Evento	Metodo interfaccia MouseListener
Mouse entrato	
Mouse uscito	
Tasto premuto	
Tasto rilasciato	
Click	

6. Scrivere i metodi dell'interfaccia **TextListener**:

Evento	Metodo interfaccia TextListener
Testo cambiato	

7. Scrivere i metodi dell'interfaccia **KeyListener**:

Evento	Metodo interfaccia KeyListener
Tasto premuto	
Tasto rilasciato	
Tasto premuto e rilasciato	

8. Scrivere i metodi dell'interfaccia **ItemListener**:

Evento	Metodo interfaccia TextListener
Stato dell'elemento	

9. Completare il codice seguente, inserendo opportunamente le stampa indicate:

```
public class gestoreM extends MouseAdapter
{
  public void mouseClicked(MouseEvent e)
  {
    System.out.println(……);
  }
  public void mouseEntered(MouseEvent e)
  {
```

"Mantenere il mouse nell'area"
"Elemento selezionato"
"P("+e.getX()+","+e.getY()+")"

```
        System.out.println(……);
    }
        public void mouseExited(MouseEvent e)
    {
        System.out.println("Mouse uscito");
    }
}
```

(D) ESERCIZI DI APPLICAZIONE

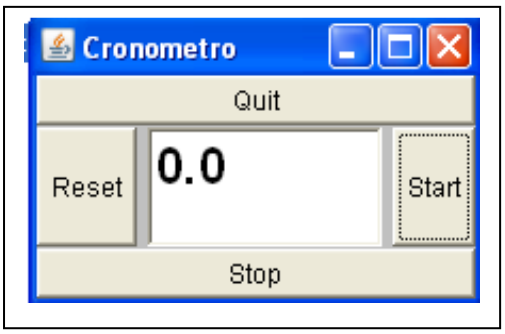
1. Creare un'interfaccia che simuli:
   a. una radiosveglia
   b. una calcolatrice
   c. un videoregistratore
   d. una lavatrice
   e. un distributore di bevande
   f. un distributore di carte telefoniche
2. Creare un'applicazione a finestre per inviare un numero di telefono ed un testo, fornendo una apposita casella spuntabile con etichetta "Invia una mail quando ricevuto". Il numero di telefono, formato da prefisso e numero, deve essere selezionabile da una lista di valori predefiniti.

(E) ESERCITAZIONI PRATICHE

1. Vogliamo realizzare un semplice cronometro.

Possiamo supporre che l'applicazione abbia un'interfaccia come quella mostrata nella seguente figura, dotata di quattro pulsanti con le seguenti funzioni:

- **Start**: avvia il cronometro
- **Stop**: blocca il cronometro e mostra il tempo trascorso dalla pressione del pulsante **Start**;
- **Reset**: azzera il display del tempo;
- **Quit**: esce dall'applicazione.



Consideriamo le seguenti classi:

- **gestoreF**: la classe contenente l'ascoltatore per la chiusura della finestra;
- **gestorePulsante**: la classe contenente l'ascoltatore che riconosce il pulsante premuto ed esegue l'azione richiesta;
- **Cronometro**: la classe ha i seguenti attributi privati:
  - **private double** *TempoIniziale*, che indica il tempo iniziale;
  - **private double** *TempoAccumulato*, il tempo conteggiato dal reset
  - **private boolean** *StatoConteggio*, dove assumiamo che fermo=falso, in funzione=vero

  e i seguenti metodi:
  - **public** Cronometro(), costruttore senza parametri
  - **public void** start(), avvio del cronometro
  - **public void** stop(), arresto del cronometro
  - **public void** reset(), reset cronometro
  - **public double** time(), calcolo tempo accumulato

Analisi della soluzione.
Iniziamo con la struttura della classe *Cronometro*.

```
public class Cronometro
{  private double tempoIniziale,            // tempo all'istante di avvio
                  tempoAccumulato;          // tempo via via crescente
   private boolean statoConteggio;          // stato del cronometro
   public Cronometro()  {…  }
            // costruttore che inizializza tempoAccumulato e statoConteggio
   public void start()  { ….. }
            // metodo start che avvia il cronometro e inizializza il tempo con …
            // System.currentTimeMillis()
   public void stop()
            // ferma il cronometro, pone in tempoFinale System.currentTimeMillis() …
            // e calcola il tempoAccumulato con:
            //  TempoAccumulato=TempoAccumulato + (TempoFinale - TempoIniziale);
   double TempoFinale
            // verifica se il conteggio è attivo; in caso affermativo, imposta tempoFinale …
            // con System.currentTimeMillis()  e disattiva il cronometro
   public void reset()
            // verifica se il conteggio è attivo; in caso negativo azzera tempoAccumulato
   public double time()
            //verifica se il conteggio è attivo ; in caso affermativo, pone in tempofinale …
            // System.currentTimeMillis() e restituisce il tempoAccumulato come:
            //        TempoAccumulato + (TempoFinale – TempoIniziale
            // se il conteggio non è attivo, invece, restituisce solo il tempoAccumulato
```

Lo schema dell'interfaccia invece è il seguente:

```
import java.awt.*;
import java.awt.event.*;
public class CronometroInterfaccia
{  static TextField tempo =new TextField();                    // casella per la mimsura del tempo
   public static void main(String args[])
      {  creazione oggetto cron  di classe cronometro;
         creazione del frame winCron con titolo;
         creazione del borderLayout bordLayout;
         creazione pannello p;
         creazione di 4 pulsanti con nomi ed etichette relative;
         impostazione del layout di p con bordLayout;
         impostazione del tempo  0.0 nela casella tempo;
         porre la casella tempo al centro di bordLayout;
         registrare l'ascoltatore per chiudere la winCron;
         registrare gli ascoltatori dei singoli pulsanti;
```

```
            impostare il font della casella tempo con "Times Roman" e "BOLD";
            impostare posizione, dimensioni e visibilità di winCron;
    } // end main
} // end class

class gestorePulsante implements ActionListener
{   Cronometro c;
    TextField t;
     public gestorePulsante(Cronometro c, TextField t)
     {   this.c=c;
         this.t=t;
     }
public void actionPerformed(ActionEvent e)
{   /*      predisporre una serie di if  nidificate per analizzare il pulsante premnuto
            mediante e.getActionCommand(). In ciascuno dei vari casi si istanzia su c
            il metodo relativo della classe Cronometro.
}
```

Implememtare infine, in modo consueto, la classe

```
class gestoreF implements WindowListener
{  ...... }
```

che consente la chiusura della finestra.

ESERCIZI CON CODICE DA RICONOSCERE

Completare le seguenti tabelle:

Individuare le proposizioni vere/false
Esercizi pratici

La numerazione è progressiva atttraverso le varie tipologie di esercizi

Completare le seguenti proposizioni

1. Una classe di problemi è formata da tutti i problemi aventi ……...

Associare le proposizioni di sinistra con le corrispondenti sulla destra:

1	L'analisi del testo…	**A**	elencare gli input e gli output
2	La tabella delle variabili di I/O…	**B**	descrivere le specifiche del problema
3	Il modello del problema…	**C**	descrivere sinteticamente la soluzione
4	Il procedimento risolutivo…	**D**	rappresentare il tipo di problema

Completare le seguenti tabelle:

IDClasse	Classe	Sezione	Specializzazione
11	3	A	Informatica
12	4	A	Informatica
13	5	A	Informatica
14	3	B	Elettronica
15	4	B	Elettronica
16	5	B	Elettronica
17	3	C	**NULL**

Domande vero/falso:

	Vero	Falso

Esercizi pratici

3. .