Esame di Programmazione III, Programmazione in Rete e Laboratorio, Ist. di Programmazione in Rete e Laboratorio, a.a. 2022/23 - Appello del 10 febbraio 2023

Esercizio 1 (10 punti)

Si simuli un'esecuzione del programma riportato sotto supponendo che **le liste di lock e di wait** siano gestite con una **politica First In First Out**. Per la simulazione si riportino tutti i cambiamenti di stato del programma, assumendo che lo stato sia descritto come segue:
{[lista di lock], [lista di wait], content, available}. Si assuma che lo **stato iniziale** del programma sia: **{[c1v, c2p, c3g, p1v, p2p, p3g], [ ], "", false}**, dove c1v è il primo elemento della lista e p3g l'ultimo.

```
public class EsercizioSenzaPrint {
   public static void main(String[] args) {
      CubbyHole c   = new CubbyHole();
      Consumer c1v = new Consumer("vegano", c);
      Producer p1v  = new Producer("vegano", c);
      Consumer c2p = new Consumer("pescetariano", c);
      Producer p3g  = new Producer("generico", c);
      Producer p2p  = new Producer("pescetariano", c);
      Consumer c3g = new Consumer("generico", c);
   }
}
class Producer extends Thread {
   private CubbyHole cubbyhole; private String type;
   public Producer(String type, CubbyHole c) {
      cubbyhole = c; this.type = type; start(); }
   public void run() {
      cubbyhole.put(type); }
}
class Consumer extends Thread {
   private CubbyHole cubbyhole;
   private String type;
   public Consumer(String type, CubbyHole c) {
      cubbyhole = c; this.type = type; start(); }
   public void run() { cubbyhole.get(type); }
}
class CubbyHole {
   private String content = "";
   private boolean available = false;
   public synchronized String get(String type) {
      while (!available || !content.equals(type)) {
         try { wait(); } catch (InterruptedException e) {} }
      available = false; String ris = content; content = "";
      notify();
      return ris;
   }
```

```
    public synchronized void put(String value) {
        while (available) {
            try { wait(); } catch (InterruptedException e) {} }
        content = value; available = true;
        notify();
    }
}
```

Esercizio 2 (10 punti)
Si sviluppino i seguenti punti:

- Descrivere in dettaglio il modello di programmazione guidata dagli eventi, spiegando su quali concetti (ed elementi architetturali) si basa e in cosa differisce dalla programmazione sequenziale.
- Descrivere brevemente come tale modello viene implementato nelle GUI Java, facendo un semplice esempio che mostri come si gestiscono gli eventi generati da un bottone (bottone SWING o JavaFX).

Esercizio 3 (10 punti)
Si consideri il codice riportato sotto:

1. Spiegare che operazione svolge il programma quando l'utente clicca il bottone "Cliccami".
2. Modificare il codice per inserire l'ActionListener
    a. come classe innestata anonima, e
    b. come lambda expression

Non è necessario riscrivere tutto il codice, basta riportare la parte che viene modificata.

```
public class Esame extends JFrame {
  private JButton button;

  private Esame() {
    button = new JButton("Cliccami");
    add(button);
    pack();
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    button.addActionListener(new Ascoltatore());
    setVisible(true);
  }

  public static void main(String[] args) {
    Esame beep = new Esame();
  }
}

class Ascoltatore implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    System.out.println("Beep!");
  }
}
```

POSSIBILI SOLUZIONI

Esercizio 1

{[c1v, c2p, c3g, p1v, p2p, p3g], [ ], “”, false}
… compatto 3 passi di esecuzione …
{[p1v, p2p, p3g], [c1v, c2p, c3g], “”, false}
{[p2p, p3g, c1v], [c2p, c3g], “vegano”, true}
…
{[c1v], [c2p, c3g, p2p, p3g], “vegano”, true}
{[c2p], [c3g, p2p, p3g], “”, false}
{[ ], [c3g, p2p, p3g c2p], “”, false}

-- deadlock

Esercizio 3

```
///// classe innestata anonima
public class EsameSvolto1 extends JFrame {
   private JButton button;

   public EsameSvolto1() {
      button = new JButton("Cliccami");
      add(button);
      pack();
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      button.addActionListener(new ActionListener() {
                                    public void actionPerformed(ActionEvent e) {
                                    System.out.println("Beep!");
                                    }
                              });
      setVisible(true);
   }
…
}
```

```
///// lambda

public class EsameSvolto2 extends JFrame {
   private JButton button;

   public EsameSvolto2() {
      button = new JButton("Cliccami");
      add(button);
      pack();
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            button.addActionListener(e -> {System.out.println("BEEP!");});
      setVisible(true);
   }
   …
}
```