Primer Certamen

En este certamen usted no podrá hacer preguntas. Si algo no está claro, indíquelo en su respuesta, haga una suposición razonable y resuelva conforme a ella.

Primera parte, **sin apuntes** (32 minutos; 32 puntos):

1.- Responda brevemente y entregue esta hoja con su nombre. (Cuide su caligrafía, 4 puntos cada respuesta)

a) En el contexto de la Programación Orientada a Objetos defina: clase, objeto, atributo, constructor.

Clase: es la definición de las características comunes a un conjunto de objetos. Las característica comunes son los atributos a considerar y el comportamiento de esos objetos.

Objeto: en POO un objeto es la representación en un programa de un objeto real. Esta representación define un nombre, un estado y un comportamiento para el objeto.

Un objeto es una instancia de una Clase.

Atributo: es una un dato que permite almacenar parte del estado de un objeto. El conjunto de atributos de un objeto almacenan su estado.

Constructor: Código que define el valor inicial de un objeto al ser creado.

b) Dada la clase A de la izquierda, ¿Qué imprime el segmento de código de la derecha?

```
public class A {
  public A( ) {
    System.out.println("En constructor");
  }
  {
    System.out.println("En bloque no estático");
  }
  static {
    System.out.println("En bloque estático");
  }
}
```

```
  :
  A primero = new A();
  A segundo = new A();
  :
```

Ponga aquí su respuesta:

```
En bloque estático
En bloque no estático
En constructor
En bloque no estático
En constructor
```

c) Implemente el método equals de la clase B.

```
public class B {
   private int edad;
   public boolean equals(Object obj) {
      //  ??
   }
}
```

Ayuda: En clase Object tiene el método:
getClass(),
Returns the runtime class of this Object.

Ponga aquí su respuesta:

```
 public boolean equals(Object obj) {
      B b;
      if (this == obj) return true;
      if (obj == null) return false;
      if (getClass() != obj.getClass()) return false;
      b = (B) obj;
      return edad == b.edad;
   }
```

d) Se tiene la definición de clases anidadas como se muestra a la izquierda. Ponga a la derecha el código para crear el objeto de nombre "anidado" como instancia de la clase AnidadaEstatica y el objeto "interno" como instancia de la clase Interna.

```
public class Externa {
    public static class AnidadaEstatica {
      // algo
    }
    public class Interna {
      // otro algo
    }
}
```

```
Externa.AnidadaEstatica anidado = new Externa.AnidadaEstatica();

Externa externo = new Externa();
Externa.Interna interno = externo.new Interna();
```

e) Considere que existen las clases **Silla** y **Estante** las cuales derivan de la clase *abstracta* **Mueble**. Suponiendo que existen los constructores por defecto, determine si las siguientes secciones de código son correctas o no y explique el motivo.

Mueble m = new Mueble(); Incorrecto No es posible crear una instancia de una clase abstracta.	Mueble m = new Silla(); Es correcto si la clase Silla define los métodos abstractos de la clase Mueble. En otro caso esto también es errado.	ArrayList<Mueble> M = New ArrayList<Mueble>; M.add((Mueble) new Estante()); Esto es correcto si la clase Estante define los métodos abstractos de la clase Mueble. En otro caso esto también es errado. El casteo es innecesario.

f) Usted cuenta con la lista dinámica **pers**, la cual contiene objetos tipo **Personas** y **Ayudantes** (esta última extiende de **Personas**). Defina el método **cuentaAyudantes**, tal que retorne la cantidad de ayudantes existentes en la lista (responda en el espacio de la derecha).

…
int cuentaAyudantes (ArrayList<Personas> pers) {
 	
// COMPLETAR
 	
}
 	
…
	int cuentaAyudantes (ArrayList<Personas> pers) {
	int i=0;
	for (Persona p: pers)
	if (p instanceof Ayudante) i++;
	return i;
	}

g) Si bien todas las excepciones deben retornar elementos subclases de **Throwable**, java prohíbe crear subclases directas de ésta. Comente si es posible crear excepciones propias, y si es así, cómo se resuelve el no poder extender **Throwable**.

Sí es posible crear excepciones propias.
La situación se resuelve porque existe una clase (Exception), que deriva de Throwable, de la cual sí es posible crear subclases directas.

h) En relación a los archivos Java Archive (jar):

h.1) Comente 2 ventajas de su uso.

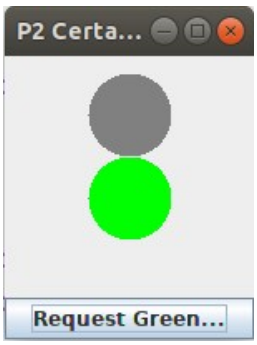
Se facilita el transporte del código. En lugar de copiar o enviar múltiples archivos .class, imágenes, audios, etc. se copia o envía solo un archivo.
La ejecución se simplifica pues se puede configurar su ejecución con doble click sobre ellos.

h.2) Describa la funcionalidad del documento de *manifiesto*.

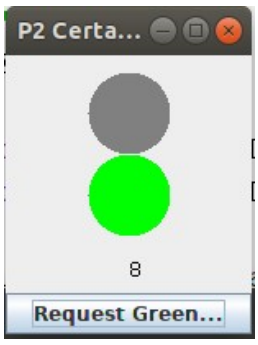
El documento manifiesto permite indicar a la máquina virtual qué clase dentro de un jar contiene el método main de inicio.

Segunda Parte, con apuntes (68 minutos) **Responda en hojas separadas.**

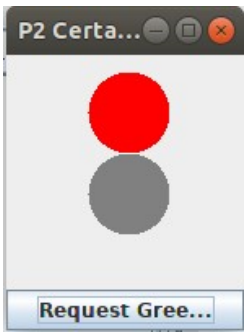
2.- (34 Puntos) Se adjunta el código correspondiente a un semáforo peatonal como el indicado a la izquierda.



Original)



a)



b)

Usted puede bajar los archivos originales desde: http://profesores.elo.utfsm.cl/~agv/elo329/1s19/C1/

a) Modifique el código dado para mostrar en segundos el tiempo restante antes de volver a rojo. (14 pts)

b) Modifique el código dado para que al presionar el botón mientras esté en verde o parpadeo, el semáforo vuelva inmediatamente a rojo. (10 pts)

c) Modifique el código original para que la aplicación solo termine su ejecución si se presiona la (x) en extremo superior derecho cuando el semáforo está en rojo. En otro caso ignora la acción de término o cierre. (10 pts) Ayuda: Previo a una configuración específica, es posible remover cualquier operación previa asociada al cierre de un JFrame invocando:

unaInstanciaDeJFrame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

```
/*********   Código proporcionado   *******/
/* P2.java */
import java.awt.*;
import javax.swing.*;
import java.util.*;

public class P2 {
   public static void main(String[] args) {
      SwingUtilities.invokeLater(new Runnable() {
         public void run() {
            MyOwnPanel tlPanel = new MyOwnPanel();
            CrosswalkTrafficLight crosswalk = new CrosswalkTrafficLight(6,4);
            tlPanel.setTrafficLight(crosswalk);
            MyOwnFrame frame = new MyOwnFrame();
            frame.add(tlPanel, BorderLayout.CENTER);
            MyButton button = new MyButton(crosswalk);
            frame.add(button, BorderLayout.SOUTH);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
         }
      });
   }
}
class MyOwnFrame extends JFrame  {
   public MyOwnFrame() {
      setTitle("P2 Certamen ELO329 1s19");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
   }
   public static final int DEFAULT_WIDTH = 150;
   public static final int DEFAULT_HEIGHT = 200;
}
class MyOwnPanel extends JPanel  {
   public MyOwnPanel() {
      setFocusable(true);
   }
   public void setTrafficLight(CrosswalkTrafficLight light) {
```

```
      tl = light;
      tl.setPanel(this);
   }
   public void paintComponent(Graphics g)   {
      super.paintComponent(g);
      Graphics2D g2 = (Graphics2D)g;
      tl.paint_view(g2);
   }
   private CrosswalkTrafficLight tl;
}

/* TraafficLightState.java  */
public enum TrafficLightState {STOP, TRANSITION, FOLLOW}

/* MyButton.java  */
import javax.swing.JButton;
import java.awt.event.*;
public class MyButton extends JButton implements ActionListener {
   public MyButton (CrosswalkTrafficLight light) {
      super("Request Green light");
      tl = light;
      addActionListener(this);
   }
   public void actionPerformed (ActionEvent event) {
      if (tl.getState() == TrafficLightState.STOP)
         tl.turnFollow();
   }
   private CrosswalkTrafficLight tl;
}

/* TrafficLight.java  */
import java.awt.Graphics2D;
import javax.swing.JPanel;
public class TrafficLight {
   public TrafficLight (int ft, int tt){
      followTime = ft;
      transitionTime = tt;
      state = TrafficLightState.STOP;
   }
   public void turnStop() {
      state = TrafficLightState.STOP;
      panel.repaint();
   }
   public void turnTransition() {
      state = TrafficLightState.TRANSITION;
      panel.repaint();
   }
   public void turnFollow() {
      state = TrafficLightState.FOLLOW;
      panel.repaint();
   }
   public TrafficLightState getState() {
      return state;
   }
   public void setPanel(JPanel p) {
      panel = p;
   }
   protected JPanel panel;
   protected int followTime;
   protected int transitionTime;
   private TrafficLightState state;
}

/* CrosswalkTrafficLight.java   */
import java.awt.*;
import java.awt.geom.*;
```

```
import java.awt.event.*;
import javax.swing.*;
public class CrosswalkTrafficLight extends TrafficLight implements ActionListener{
   public CrosswalkTrafficLight (int ft, int tt) {
      super(ft, tt);
      red_view = new Ellipse2D.Double(origen_x, origen_y, DIAMETER, DIAMETER);
      green_view = new Ellipse2D.Double(origen_x, origen_y+DIAMETER, DIAMETER, DIAMETER);
      timer = new Timer(ON_OFF_TIME, this);
   }
   public void paint_view (Graphics2D g2d) {
      switch (getState()) {
         case STOP: if (timer.isRunning()) timer.stop();
                    g2d.setColor(Color.RED);
                    g2d.fill(red_view);
                    g2d.setColor(Color.GRAY);
                    g2d.fill(green_view);
                    break;
         case TRANSITION: g2d.setColor(Color.GRAY);
                          g2d.fill(red_view);
                          if(toggle)
                             g2d.setColor(Color.GREEN);
                          g2d.fill(green_view);
                          break;
         case FOLLOW: if (!timer.isRunning()) {
                        timer.start();
                        time=followTime+transitionTime;
                      }
                      g2d.setColor(Color.GRAY);
                      g2d.fill(red_view);
                      g2d.setColor(Color.GREEN);
                      g2d.fill(green_view);
      }
   }
   public void actionPerformed (ActionEvent event) {
      toggle = !toggle;
      if (toggle) {
         time--;
         if (time==transitionTime) turnTransition();
         if (time == 0) turnStop();
      }
      panel.repaint();
   }
   private Ellipse2D red_view, green_view;
   private Timer timer;
   private int time;
   private boolean toggle=false;
   private int origen_x=50;
   private int origen_y=10;
   private static int DIAMETER=50;
   private static int ON_OFF_TIME = 500; //[ms]
}
/*********   FIN Código proporcionado   *******/
```

a) Solo se modifica clase /* CrosswalkTrafficLight.java */

```
import java.awt.*;
import java.awt.geom.*;
import java.awt.event.*;
import javax.swing.*;
public class CrosswalkTrafficLight extends TrafficLight implements ActionListener{
   public CrosswalkTrafficLight (int ft, int tt) {
      super(ft, tt);
      red_view = new Ellipse2D.Double(origen_x, origen_y, DIAMETER, DIAMETER);
      green_view = new Ellipse2D.Double(origen_x, origen_y+DIAMETER, DIAMETER, DIAMETER);
      timer = new Timer(ON_OFF_TIME, this);
   }
   public void paint_view (Graphics2D g2d) {
      switch (getState()) {
         case STOP: if (timer.isRunning()) timer.stop();
                    g2d.setColor(Color.RED);
                    g2d.fill(red_view);
                    g2d.setColor(Color.GRAY);
                    g2d.fill(green_view);
                    break;
         case TRANSITION: g2d.setColor(Color.GRAY);
                          g2d.fill(red_view);
                          if(toggle)
                             g2d.setColor(Color.GREEN);
                          g2d.fill(green_view);
          // Código necesario para pregunta a)
                      g2d.setColor(Color.BLACK);
                      g2d.drawString(""+time, origen_x+DIAMETER/2, origen_y+(int)(2.5*DIAMETER));
          // fin código pregunta a
                          break;
         case FOLLOW: if (!timer.isRunning()) {
                        timer.start();
                        time=followTime+transitionTime;
                      }
                      g2d.setColor(Color.GRAY);
                      g2d.fill(red_view);
                      g2d.setColor(Color.GREEN);
                      g2d.fill(green_view);
            // Código necesario para pregunta a)
                      g2d.setColor(Color.BLACK);   // a)
                      g2d.drawString(""+time, origen_x+DIAMETER/2, origen_y+(int)(2.5*DIAMETER));
            // fin código pregunta a
      }
   }
   public void actionPerformed (ActionEvent event) {
      toggle = !toggle;
      if (toggle) {
         time--;
         if (time==transitionTime) turnTransition();
         if (time == 0) turnStop();
      }
      panel.repaint();
   }
   private Ellipse2D red_view, green_view;
   private Timer timer;
   private int time;
   private boolean toggle=false;
   private int origen_x=50;
   private int origen_y=10;
   private static int DIAMETER=50;
   private static int ON_OFF_TIME = 500; //[ms]
}
```

b) Solo se modifica clase /* MyButton.java */

```
import javax.swing.JButton;
import java.awt.event.*;
public class MyButton extends JButton implements ActionListener {
   public MyButton (CrosswalkTrafficLight light) {
      super("Request Green light");
      tl = light;
      addActionListener(this);
   }
   public void actionPerformed (ActionEvent event) {
      if (tl.getState() == TrafficLightState.STOP)
         tl.turnFollow();

     // Inicio código respuesta pregunta b)
      else
         tl.turnStop();
     // Fin código respuesta pregunta b)
   }
   private CrosswalkTrafficLight tl;
}
```

c) Solo se requiere modificar la clase P2

```
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;    //Requerido por pregunta c)
import javax.swing.event.*; //Requerido por pregunta c)

public class P2 {
   public static void main(String[] args) {
      SwingUtilities.invokeLater(new Runnable() {
         public void run() {
            MyOwnPanel tlPanel = new MyOwnPanel();
            CrosswalkTrafficLight crosswalk = new CrosswalkTrafficLight(6,4);
            tlPanel.setTrafficLight(crosswalk);
            MyOwnFrame frame = new MyOwnFrame();
            frame.add(tlPanel, BorderLayout.CENTER);
            MyButton button = new MyButton(crosswalk);
            frame.add(button, BorderLayout.SOUTH);
      // Inicio código pregunta c)
      //    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
            frame.addWindowListener( new WindowAdapter() { /* Clase anónima */
                  public void windowClosing(WindowEvent e) {
                    if (crosswalk.getState()==TrafficLightState.STOP)
                       System. exit( 0);
                  }
               }
            );
      // Fin código pregunta c)
            frame.setVisible(true);
         }
      });
   }
}
```

3.- (34 puntos) Cree una clase “Comunidad” para almacenar un conjunto de personas. Cada persona tiene un nombre (un String) y su edad en años (un int). Una instancia de Comunidad debe permitir:
- Ingresar una nueva persona,
- Obtener la edad de alguna persona a partir de su nombre (si no está retorna -1),
- Mostrar por la salida estándar la lista de personas ordenadas por su edad (de menor a mayor) y para igual edad, ordenadas por su nombre.

Un ejemplo de su uso es (http://profesores.elo.utfsm.cl/~agv/elo329/1s19/C1/):

```
class POO2 {
      public static void main (String[] args) {
            Comunidad vecinos = new Comunidad();
            vecinos.add("Francisco", 19); // Nombre y edad
            vecinos.add("Jorge", 25);
            vecinos.add("Camila", 19);
            System.out.println("La edad de Camila es "+vecinos.getAge("Camila"));
            vecinos.listarOrdenados();
      }
}
```

Ayuda: Revise el método estático sort de la clase Collections.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.*;

class Comunidad {

   public int getAge(String name) {
      for (Person p : pers)
         if (p.getName().equals(name))
            return p.getAge();

      return -1;
   }

   public void add(String n, int e) {
      pers.add(new Person(n,e));
   }

   public void listarOrdenados() {
      System.out.println("\nComunidad:");

      Collections.sort(pers);
      for (Person p : pers)
         System.out.println(p);
   }

   private ArrayList <Person> pers = new ArrayList<Person>();;
}

class Person implements Comparable<Person>{
   private String name;
   private int age;

   Person(String n, int a) {
      name = n;
      age = a;
   }
   public int getAge() {
      return age;
   }
   public String getName() {
```

```
      return name;
   }
   public int compareTo(Person p) {
      if (age<p.age)
         return -1;
      else if (age>p.age)
         return 1;
      else return name.compareToIgnoreCase(p.name);
   }

   public String toString(){
      return name + ", " + age;
   }
}
```