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 en hoja con su nombre. (4 puntos cada respuesta)

a) ¿Qué diferencia hay entre programación imperativa y declarativa? ¿A cuál de ellas corresponde la programación orientada a objetos? Justifique.

En la programación declarativa se especifica el resultado deseado y no la forma de obtenerlo, en cambio en programación imperativa se indican los pasos para obtener el resultado.
La programación orientada a objetos es imperativa.

b) Indique qué respondería si en una entrevista de trabajo le preguntan ¿Qué es ligado dinámico? ¿Existe o no relación entre ligado dinámico y la programación genérica? ¿Si existe, cuál sería esa relación; si no, explique?

Ligado dinámico es una propiedad de los lenguajes orientados a objetos que consiste en la determinación en tiempo de ejecución del código a ejecutar ante la invocación de un método. Este código queda definido por el objeto que recibe la invocación y no por la referencia usada para invocar el método.
Sí existe relación. Gracias al ligado dinámico es posible tener métodos genéricos, lo cual hace posible la reutilización de código.
Nota: Ejemplo: el método para ordenar datos dentro de un arreglo sólo requiere que sus elementos sean comparables según cada clase de objeto lo defina.

c) En un archivo Java se ocupa la instancia t1 de la clase Timer del paquete javax.swing, y la instancia t2 de la clase Timer del paquete java.util. Suponga que éstos son los únicos objetos del archivo. Muestre qué pondría usted en la sección "import" y las instrucciones usadas por usted para crear las instancias t1 y t2. Para t1 use constructor Timer(int delay, ActionListener listener), pata t2 use constructor por omisión.

Bajo inport no iría código.

```
javax.swing.Timer t1 = new javax.swing.Timer( "delay", myListener); /* siendo myListener un objeto subtipo de ActionListener */
java.util.Timer t2 = new java.util.Timer();
```

d) Un compañero le pregunta: "¿Cómo sé si debo o no redefinir el método clone de una clase para generar copias de instancias de esa clase?" ¿Cuál sería su respuesta?

Se debe redefinir el método clone si la clase tiene objetos no instancias de String como atributos y el problema requiere usar semántica equivalente a copia profunda.

e) ¿Cuál es la función de los layout manager? ¿En qué clase se especifica cuál usar?

Su función es facilitar la organización del espacio en una interfaz gráfica de manera que ante cambios de tamaño de la ventana la posición de los objetos sea la deseada.
Se especifica en la instancia de la clase panel.

f) Mencione dos ventajas de la incorporación de manejo de excepciones en lenguajes orientados a objetos, respecto de lenguajes no orientados a objetos como C.

* El código que implementa la lógica del programa se separa del código para atender las condiciones de excepción. Esto facilita la claridad y mantención del código.
* Varias situaciones de error pueden ser agrupadas y manejadas en forma conjunta.

g) ¿Qué diferencia existe entre una clase anidada estática de aquella anidada no estática?
Si tenemos la clase A y la clase B anidada dentro de A. Dé un ejemplo de creación de una instancia de B cuando ésta está definida como estática.

La clase anidada no estática puede acceder a todos los métodos y atributos de la clase anfitriona. Las clases anidadas estáticas sólo pueden acceder a métodos estáticos de la clase anfitriona.
Ejemplo: A.B b = new A.B();

h) El archivo html contiene la clase MyApplet que deriva de JApplet se ve como sigue. MyApplet depende de varias clases.
<applet code="MyApplet.class" width="300" height="100"> </applet>
¿Funciona? Justifique. ¿Haría usted algún cambio para hacerla funcionar eficientemente? Explique.

Sí funciona. Si bien MyApplet depende de varias clases, el navegador las solicita al servidor conforme éstas son requeridas.
Sí. Generar un archivo jar con todas las clases relacionadas con la aplicación y ponerlas asociadas al atributo archive del rótulo applet. Así con un acceso al servidor, el navegador baja todas las clases relacionadas.

Segunda Parte, con apuntes (68 minutos)
2.- (34 puntos) Considerando las clases involucradas en un sistemas de bolas, resortes y osciladores como la tarea 2, se desea crear clases para el elemento físico Elástico (Elastic) y su vista (ElasticView). Los elásticos se pueden modelar como resortes con constante de elasticidad no nula cuando está estirado más allá de su largo en reposo, y con constante de elasticidad nula si la distancia entre sus extremos es inferior a su largo en reposo. La representación gráfica de un elástico será la de una trazo recto (línea recta). Diseñe e implemente las clases de modo que permitan su integración en la tarea 2:

a) Elastic (18 puntos)
b) ElasticView (16 puntos)

```
import java.awt.*;

public class Elastic extends PhysicsElement implements Connector{
  private static int id=0;
  protected final double restLength;
  private final double stiffness;
  protected Attachable a_end, b_end;
  protected ElasticView view;

  private Elastic(){
    this(0,0);
  }
  public Elastic(double restLength, double stiffness){
   super(id++);
    this.restLength = restLength;
    this.stiffness = stiffness;
    a_end = b_end = null;
   view = new ElasticView(this);
  }
  public void attachEnd (Attachable sa) {
    if(a_end==null)
      a_end = sa;
    else
      b_end = sa;
    sa.attach(this);
  }
  private Vector2D getStrech() {
    if ((a_end == null) || (b_end == null))
```

```
        return new Vector2D();
      Vector2D b_a = b_end.getPosition().minus(a_end.getPosition());
      Vector2D b_aUnitary = b_a.unitary();
      return b_a.minus(b_aUnitary.times(restLength));
    }
    public Vector2D getForce(Ball ball) {
      if ((ball != a_end) && (ball != b_end))
        return new Vector2D();
      Vector2D strech = getStrech();
      if (strech.module() <= restLength)
        return new Vector2D();
      if (ball == a_end)
        return strech.times(stiffness);
      else
        return strech.times(-stiffness);
    }
    public Vector2D getAendPosition() {
      return a_end.getPosition();
    }
    public Vector2D getBendPosition() {
      return b_end.getPosition();
    }
    public double getRestLength() {
      return restLength;
    }
    public void paintView (Graphics2D g) {
      view.paint(g);
    }
}

import java.awt.geom.*;
import java.awt.*;
class ElasticView {
    private static final double xPoints[]={0,1.0};
    private static final double yPoints[]={0,0};
    private static final Path2D.Double polyline =
                    new Path2D.Double(Path2D.WIND_EVEN_ODD,xPoints.length);
    private Stroke stroke;
    private Elastic elastic;

    static {
      polyline.moveTo (xPoints[0], yPoints[0]);
      for (int index = 1; index < xPoints.length;index++)
        polyline.lineTo(xPoints[index], yPoints[index]);
    }
    public ElasticView(Elastic elastic) {
      this.elastic = elastic;
      AffineTransform at = AffineTransform.getTranslateInstance(0,0);
      stroke = new BasicStroke(0.01f);
    }
    public Vector2D getAendPosition() {
      return elastic.getAendPosition();
    }
    public Vector2D getBendPosition() {
      return elastic.getBendPosition();
    }
    public void paint (Graphics2D g){
      Vector2D a=elastic.getAendPosition();
      double restLength = elastic.getRestLength();
      double ax = a.getX();
      double ay = a.getY();
      Vector2D v = elastic.getBendPosition().minus(a);
      AffineTransform at = AffineTransform.getTranslateInstance(ax, ay);
      at.rotate(v.getX(), v.getY());
      at.scale(v.module(),  restLength);
      Path2D.Double shape = (Path2D.Double) at.createTransformedShape(polyline);
      if (v.module() < restLength)
```

```
        g.setColor(Color.BLACK);
      else
        g.setColor(Color.RED);
      g.setStroke(stroke);
      g.draw(shape);
    }
}
```

3.- (34 puntos)

a) (20 puntos) Desarrolle la aplicación gráfica Titulo.java que muestra un campo de texto (JTextField) y un JInternalFrame. La aplicación espera el ingreso de un texto en el campo de texto y lo usa para cambiar el título del JInternalFrame.

b) (14 puntos) Cree la clase AppletTitulo.java. Ésta tiene igual funcionalidad que la aplicación pedida en a) y puede ser mostrada como applet en una página web.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Title {
  public static void main(String argv[]) {   3 puntos
    new TitleFrame();
  }
}

class TitleFrame extends JFrame {   4 puntos
  public TitleFrame(){
    TitleGUI gui = new TitleGUI();
    add(gui);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 300);
    setVisible(true);
  }
}
class TitleGUI extends JPanel implements ActionListener {  3 puntos
  private JTextField quote = new JTextField( 20);
  private JInternalFrame iFrame;  // por atributos y su asignación en constructor 3 puntos.

  public TitleGUI() {
    add(quote);
    iFrame = new JInternalFrame("title");
    iFrame.setVisible(true);
    iFrame.add(new JLabel("                                    "));
    add(iFrame);
    quote.addActionListener(this);       3 puntos
  }

  public void actionPerformed( ActionEvent e) {   4 puntos
    iFrame.setTitle(quote.getText());
  }
}
================================================
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class AppletTitle extends JApplet {   Hereda de JApplet 3 puntos
  public void init(){           4 puntos
    TitleGUI gui = new TitleGUI();
    add(gui);
  }
```

```
}
/* Esta clase es la misma que la parte a) */   // reutilizar código 7 puntos
class TitleGUI extends JPanel implements ActionListener { // panel: parte interna de la ventana
  private JTextField quote = new JTextField( 20);
  private JInternalFrame iFrame;

  public TitleGUI() {
    add(quote);
    iFrame = new JInternalFrame("title");
    iFrame.setVisible(true);
    iFrame.add(new JLabel("                                    "));
    add(iFrame);
    quote.addActionListener(this);
  }

  public void actionPerformed( ActionEvent e) {
    iFrame.setTitle(quote.getText());
  }
}
```