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) ¿Cuál es la diferencia entre un objeto real y un modelo de un objeto real? ¿Qué debemos incluir en el modelo?
El modelo de un objeto real imita o representa algunas de las características del objeto real. (La realidad es muy compleja para ser representada en su totalidad.)
El modelo debe incluir lo necesario para obtener resultados que semejen la realidad dentro del grado de precisión deseado.

b) Indique qué respondería si en una entrevista de trabajo el preguntan ¿Qué es herencia?
Herencia es un mecanismo que permite definir una clase extendiendo la definición de otra.
Para que haya herencia debe existir la relación "es-un" y la propiedad de subtipo entre ambas clases.

c) Si en la creación de una clase, un programador opta por un nombre de archivo distinto al de la clase ¿se genera algún problema? Refiérase al caso en que la compilación es por consola y considere dos casos: i) la clase incluye el método main, ii) la clase es usada por la aplicación pero no incluye el método main.
Sí se genera un problema. Casos:
i) El comando para compilar debe usar el nombre del archivo, pero el comando para ejecutar debe usar el nombre de la clase.
ii) En este caso el problema es mayor pues la compilación de la aplicación arroja error al no encontrar la implementación de la clase por usar otro nombre en el archivo. El usuario deberá compilar esa clase primeroo y luego otras que la ocupen.
(Nota: En ambos casos no es algo recomendado.)

d) Un amigo le pide ayuda para superar el siguiente error de compilación:

```
agustin@agustin-Satellite-P55-A:~/teaching/elo329/1s14/Assignments/T2/source$ javac Etapa1/PhysicsLab.java
Etapa1/PhysicsLab.java:25: error: cannot find symbol
  private void createConfiguration(MyWorld world) {
                                   ^
 symbol:   class MyWorld
 location: class PhysicsLab_GUI
```

Si su amigo no puede cambiar el comando para compilar ¿qué consejo de da usted?
Debe configurar la variable de ambiente CLASSPATH parra incluir en ella el directorio donde se encuentre el código fuente.

e) ¿Qué valor tiene asignado cada atributo de **b** después de ejecutar class B **b**= new B(6);?

```
class A {
  protected int i=2;
  public A() {
     i=3;
}}
```

```
class B extends A {
  private Sring nombre;
  private int j = 4;
  public B(int k){
    i=5;  j=k;
}}
```

b.Sring = null , b.i = 5, b.j = 6;

f) ¿Cuándo debemos poner la palabra reservada "**throws**" en la declaración de un método? Por ejemplo en:
public void doio (InputStream in, OutputStream out) **throws** IOException {...}
Se debe poner cuando el método invoca código que puede generar excepciones y no utiliza la sentencia try-catch. También se debe utilizar cuando el código del método lanza expeciones con sentencia **throw**.

g) Un programador afirma: "Cuando existen clases anónimas en una aplicación, no existirá archivo separado

.class para esa clase" ¿Está usted de acuerdo? Justifique.
No. El compilador genera un archivo .class por cada clase anónima o no del código. La clase anónima aparecerá con el nombre de la clase que la contiene seguido de un punto y un número.

h) Explique y muestre comandos para preparar una aplicación Java de varios archivos .class para que ésta sea transportada como un único archivo. ¿Cómo se ejecutaría desde consola?
Se debe crear un archivo "manifiesto" que indique cuál es la clase que incluye el método main (2 punto), luego de compilar la aplicación se debe ejecutar:
$ jar cmf "archivo manifiesto" app.jar *.class (1 punto)
Para ejecutar la aplicación se debe ejecutar:
$ java -jar app.jar (1 punto)
Nota: Para su ejecución con doble click, poner permisos de ejecución al archivo. $ chmod 700 app.jar

Segunda Parte, con apuntes (68 minutos)
2.- (18 puntos) Considerando las clases involucradas en un sistemas de bolas, resortes y puntos fijos como la tarea 2, se desea crear una clase para el elemento físico Oscilador. Éste se representa visualmente por un cuadrado amarillo con un círculo negro en su interior. Cuando éste es seleccionado la zona amarilla pasa a rojo . Un Oscilador se caracteriza por oscilar horizontalmente en torno a un punto central con una amplitud y frecuencia dada. Su punto central, su amplitud y frecuencia de oscilación se definen en su constructor. Como los puntos fijos (FixedHoook) y bolas (Ball), resortes se pueden enganchar a un oscilador.
Diseñe e implemente las clases de modo que permitan su integración en la tarea 2:
a) Oscilador (18 puntos)

```
import java.util.*;
import java.awt.*;

public class Oscillator extends PhysicsElement implements Simulateable, SpringAttachable {
  private static int id=0;  // Oscillator's identification number
  private double center;     // oscillator's center
  private double amplitude;     // oscillation's amplitude       // 2 puntos
  private double w;     // oscillator's frequency [rad]
  private double time;
  private double pos_t;
  private double pos_tPlusDelta;
  private OscillatorView view;
  private ArrayList<Spring> springs;

  private Oscillator(){
   this(1.0,0.3,0.5);
  }
  public Oscillator(double c, double a, double f){      // 3 puntos
    super(id++);
    pos_t = pos_tPlusDelta = center = c;
    amplitude = a;
    w = 2*Math.PI*f;
    time=0;
    view = new OscillatorView(this);
    springs = new ArrayList<Spring>();
  }
  public double getPosition() {
    return pos_t;
  }
  public void computeNextState(double delta_t, MyWorld world) {     // 4 puntos
    time+=delta_t;
    pos_tPlusDelta = center + amplitude*Math.sin(w*time);
  }
  public void updateState(){                                        // 2 puntos
   pos_t = pos_tPlusDelta;
  }
  public void updateView (Graphics2D g) {
   view.updateView(g);
  }
  public boolean contains(double x, double y) {
    return view.contains(x,y);
  }
  public void setSelected(){
```

```
    view.setSelected();
  }
  public void setReleased(){
    view.setReleased();
  }
  public void dragTo(double x){  // pos_t = center +dx                  // 4 puntos
    center+= (x-pos_t);        // x = new_center + dx
    pos_t=x;                  // new_center= x - dx = x - (pos_t-center)
  }                          // new_center=center +x-pos_t

  public String getDescription() {
   return "Oscillator_" + getId()+":x";
  }
  public String getState() {
   return getPosition()+"";
  }
  public void attachSpring(Spring s){
   springs.add(s);
  }
  public void detachSpring(Spring s){                    // 3 otros métodos
   springs.remove(s);
  }
}
```

b) OsciladorView (16 puntos)

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

public class OscillatorView {
  private Color color = Color.YELLOW;                          // 2+2=4 puntos
  private Rectangle2D.Double outerShape=null;
  private Ellipse2D.Double centerShape = null;
  private double size;  // width/2
  private Oscillator osc;

  public OscillatorView (Oscillator o){                      // 4 puntos
    osc = o;
    size = 0.1;
    outerShape = new Rectangle2D.Double(o.getPosition()-size,-size, 2*size, 2*size);
    centerShape = new Ellipse2D.Double(o.getPosition()-size/2, -size/2, size, size);
  }
  public boolean contains (double x, double y){           // 2 puntos
    return outerShape.contains(x,y);
  }
  public void setSelected (){                           // 2 otros métodos
    color = Color.RED;
  }
  public void setReleased() {
    color = Color.YELLOW;
 }
  void updateView(Graphics2D g) {                          // 4 puntos
    outerShape.setFrame(osc.getPosition()-size, -size, 2*size, 2*size);
    g.setColor(color);
    g.fill(outerShape);
    centerShape.setFrame(osc.getPosition()-size/2, -size/2, size, size);
    g.setColor(Color.BLACK);
    g.fill(centerShape);
  }
}
```

3.- (34 puntos)

Desarrolle una aplicación gráfica en Java que muestre un texto indicando el estado del botón izquierdo del mouse (presionado o no) y otro texto para indicar si el mouse se ha arrastrado (drag). El arrastrado concluye cuando el botón izquierdo es liberado.

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

public class MouseTest extends JFrame {                    //  5 puntos constructor
  MouseTestGUI gui = new MouseTestGUI(); // pudo ir en constructor
  public MouseTest() {
    setTitle("Mouse Test");
    setSize(200,50);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    getContentPane().add(gui);
    setVisible(true);
  }
  public static void main( String[] args) {                //  5 puntos
    MouseTest mimic = new MouseTest();
  }
}

class MouseTestGUI extends JPanel {
  private JLabel pressedState = new JLabel("Mouse State");    //  5 puntos atributos
  private JLabel draggedState = new JLabel("Dragging State");

  public MouseTestGUI() {                                    //  5 puntos  constructor
    super(new BorderLayout());   // it's OK defaut layout manager
    add(pressedState,BorderLayout.PAGE_START);
    add(draggedState, BorderLayout.CENTER);

    MouseAdapter listener = new MouseAdapter(){      //  9 puntos implementación de listeners
      public void mousePressed(MouseEvent e) {
        if (e.getButton()==MouseEvent.BUTTON1) // it's Ok if missing.
          pressedState.setText("Mouse Pressed");
      }
      public void mouseDragged(MouseEvent e) {
        draggedState.setText("Dragging Mouse");
      }
      public void mouseReleased(MouseEvent e) {
        draggedState.setText("Not Dragging");
        pressedState.setText("Mouse Released");
      }
    };

    addMouseMotionListener(listener);          //  5 puntos registro de listeners
    addMouseListener(listener);
  }
}
```