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. (4 puntos cada respuesta)

a) Defina método, atributo y referencia en un lenguaje orientado a objetos como Java.

Método: es el miembro función de una clase. Es el que permite modelar el comportamiento de una clase.
Atributo: es un miembro dato de una clase. El conjunto de atributos permiten almacenar el estado de un objeto.
Referencia: es el nombre con el cual identificamos un objeto específico. Usamos la referencia para acceder a los métodos y atributos de un objeto específico.

b) ¿Qué condición (es) se debe(n) cumplir para que haya una relación de herencia entre dos clases? Mencione dos clases donde haya relación de herencia y explique cómo se cumple esta(s) condición(es).

Se deben satisfacer dos condiciones: debe existir una relación es-un entre ellas y además debe haber una relación de subtipo entre ambas.
La clase Bola hereda de ElementoFisico. Se cumple relación es-un debido a que una Bola es un Elemento Físico. Además, si se requiere un elemento físico, una bola puede cumplir ese rol de buena forma al tener las propiedades esperadas para un elemento físico.

c) Explique dos características de Java que no estaban presentes en C++ y cómo la Máquina Virtual Java ayuda en ello.

Java es un lenguaje interpretado y multiplataforma.
Java no se generan “fugas de memoria” (Memory leaks).
La JVM (Java Virtual Machine) permite la multiplataforma al disponer de una específica para cada plataforma y como las aplicaciones Java son interpretadas por la JVM, se logra que cualquier aplicación pueda correr en cualquier plataforma.
La JVM efectúa una recolección de basura (Garbage collection) en la que recupera los espacios de memoria luego que éstos son inaccesibles para la aplicación.

d) Indique si el siguiente código tiene algún problema. Muestre los comandos que usaría para compilar y luego para ejecutar este programa.

```
public class FirstSample {
  private course = "ELO329";
  public static void main(String[ ] args){
    System.out.println("Hello " + course);
  }
}
```

Sí tiene problema. El atributo course no tiene clase asociada (String) y es propio de una instancia de la clase FirstSample y no puede ser accedido desde el método estático main. Hasta ese momento no se ha creado ninguna instancia de FirstSample. Los métodos estáticos solo pueden acceder a atributos estáticos.
Para compilar: $ javac FirstSample.java
Para ejecutar: $ java FirstSample

e) ¿En qué situaciones se utiliza el bloque de inicialización estática? Un ejemplo de ello es:

```
static {
  Random generator = new Random();
  nextId = generator.nextInt(10000);
}
```

Se ocupa para dar el valor inicial a variables estáticas y en especial cuando ello requiere la ejecución de varias instrucciones del lenguaje.

f) Para el código: ClassA a = new ClassB();
i) Indique qué relación debe existir entre ClassA y ClassB para que la instrucción sea válida.
La clase ClassB debe heredar de la clase ClassA en forma directa o indirecta a través de otras clases intermedias.

ii) ¿Qué métodos y de qué clase puedo invocar usando la referencia **a**?
Con la referencia a se puede invocar cualquier método de la clase ClassA o de cualquiera de las clases superiores de las cuales la clase ClassA hereda.

iii) ¿Si un mismo método público aparece implementado en ambas clases, el código de qué clase se ejecuta?
Se ejecuta el código de la clase ClassB, excepto que el método sea privado en ClassA, en cuyo caso se ejecuta la implementación de ClassA.

g) ¿Qué es la programación basada, dirigida o conducida por eventos? ¿Podrá su programa atender varios eventos si al generarse uno de ellos, el método invocado se queda en un loop infinito?

La programación conducida por eventos es aquella donde el programador accede a un mecanismo (creado por él o el lenguaje) para registrar objetos asociados a eventos de interés. Luego ante la llegada de un evento, ese mecanismo invoca el método que corresponda del objeto registrado para ese evento.
No. En un loop infinito no se pueden atender, en general, otros eventos (a menos que usemos hebras).

h) Un amigo le hace entrega del archivo Final.jar que contiene todas las clases de una aplicación; sin embargo, no incluye el archivo manifiesto. Además le indica que la clase "Principal" contiene el método main. Muestre el comando que le permite ejecutar la aplicación.
Suponga ahora que su jefe le pide que la aplicación se pueda ejecutar usando $java -jar Principal.jar
Explique los pasos para generaría el archivo Principal.jar

$java -cp Final.jar Principal
Una opción para generar el archivo Principal.jar se:
* Expandir todos los archivos .class presentes en Final.jar
($ jar xf Final.jar)
* Crear un archivo de manifiesto que especifique cuál es la clase que contiene el main.
($ echo "Main-class: Principal" > manifiesto.mf)
* Volver a generar ell archivo jar incluyendo el archivo de manifiesto.
($ jar cmf manifiesto.mf Principal.jar *.class)
Nota: los comandos entre paréntesis no son pedidos en la pregunta.

Segunda Parte, con apuntes (68 minutos)

2.- (34 puntos) Considerando las clases involucradas en un sistema de bolas y resortes como en la tarea 1. En lugar de tener contenedores rectangulares, se pide ahora que sean circulares (circunferencias). Considere que las bolas siempre serán puestas al interior de un contenedor circular.
Diseñe e implemente la clase ContenedorCircular de modo que permita su uso en la tarea 1.

Uso adecuado de modificadores de acceso: 5 pts
Definición de Atributos: 5 pts.
Métodos para acceder a atributos: 5 pts..
Método collide: 14 pts
Dos métodos finales para no ser abstracta: 5 pts.

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

public class ContenedorCircular extends PhysicsElement  {
   private static int id=0;  // Class identification number
   private Vector2D center;
   private double radius;

   private ContenedorCirdular(){   // nobody can create an objet without state
     super(id++);
   }
   public CoontenedorCircular(Vector2D c, double r){
      super(id++);
      center = c;
      radius = r;
   }
   public Vector2D getCenter() {
      return center;
   }
   public double getRadious() {
      return radius;
   }
   public PhysicsElement collide(Ball b){
      if (b == null) return null;
      Vector2D center2center = b.getPosition().minus(center);
      boolean isNotTouching = center2center.module()+b.getRadius() < radius;
      if (isNotTouching) return null;
      boolean isApproaching = center2center.dot(b.getSpeed())>0;
      if (isApproaching) return this;
      else return null;
   }
   public String getDescription() {
      return "ContenedorCircular_" + getId()+":"+c.getDescription()+", center";
   }
   public String getState() {
      return c+", "+radius;
   }
}
```

3.- (34 puntos)
Desarrolle la aplicación Temporizador.java. Al ejecutar esta aplicación el usuario debe especificar un número entero positivo en la línea de comandos. Durante la ejecución, La GUI de la aplicación muestra el número ingresado inicialmente y lo decrementa en una unidad por cada segundo. Cuando el número llega a cero, en lugar del número la aplicación muestra el mensaje FIN!. La aplicación termina al cerrar su venta principal.

Estructura de aplicación: clase que extiende de JFrame y otra de JPanel: 5 pts.
Main adecuado y buen paso de parámetro: 5 pts
Constructor de Temporizador: 3 pts.
Atributos de clase GUI o equivalente: 3 pts.
Constructor de GUI, excepto timer: 3 pts.
Manejo de timer: su creación, creación de listener: 8 pts.
función actionPerformed: 7 pts (si no detiene timer -4 pts)

```
import java.awt.event.*; //ActionListener, ActionEvent
import javax.swing.*;  // JFrame, JPanel, Timer

public class Temporizador extends JFrame {
  public Temporizador(int time) {
    setTitle("Temporizador 1° Certamen ELO329 1s2016");
    setSize(100, 40);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    add(new TemporizadorGUI(time));
    setVisible(true);
  }
  public static void main( String[] args) {
    new Temporizador(Integer.parseInt(args[0]));
  }
}

class TemporizadorGUI extends JPanel implements ActionListener {
  private JLabel label;
  private int time;
  private Timer passingTime;
  public TemporizadorGUI(int t) {
    time = t;
    label = new JLabel(" "+time);
    passingTime = new Timer(1000, this);
    add(label);
    passingTime.start();
  }

  public void actionPerformed( ActionEvent e) {
    time--;
    if (time > 0)
      label.setText(" "+ time);
    else  {
      label.setText("FIN!");
      passingTime.stop();
    }
  }
}
```