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.

a) ¿Qué pasos involucra el diseño e implementación orientados a objetos?
**** Identificar los objetos relevantes del problema a resolver. Cada categoría de objetos da origen a una clase.***
**** Identificar la relación entre las clases.***
**** Identificar los atributos y comportamientos o servicios que éstas deben cumplir.***
**** Implementar cada uno de los comportamientos o servicios, eventualmente haciendo uso de los servicios ofrecidos por otras clases de objetos.***

b) Además de mantener el nombre del método y sus parámetros ¿En qué otra(s) cosa(s) debemos poner atención al redefinir un método presente en una clase base?
**** Debemos fijarnos que si retornamos un objeto, éste debe ser instancia de la misma clase del método a redefinir o bien instancia de una clase derivada de ésta.***
**** También debemos fijarnos en mantener las excepciones, si éstas no son lanzadas, en la redefinición no podemos lanzar excepciones.***

c) ¿Es posible tener un programa con implementaciones para el método main en dos clases? Explique por qué no o cómo se usaría tal cosa.
Sí es posible.
Cómo se usaría: Se crean métodos main en dos clases distintas, luego al compliar se generarán los dos .class correspondientes. Para correr cada versión de main, usamos el nombre de la clase correspondiente como parámetro de la máquina virtual Java.

d) ¿Qué debemos hacer para dar sentido a la salida: System.out.print(persona); cuando persona es un objeto?
Basta redefinir el método toString en la clase correspondiente a persona.

e) Como atributo de una clase se tiene static final double valorDolar, cuyo valor es leído desde teclado ¿cómo se debe programar tal inicialización? (puede usar pseudo código para la lectura de teclado).
En la clase donde esta constante está incluir una inicialización estática, del tipo:

```
static final double valorDolar;
static {
      Scanner s = new Scanner(System.in);
      valorDolar = s.nextDouble();
}
```

f) Al diseñar una solución de software ¿qué lo hace decidir por crear una clase abstracta en lugar de una clase en propiedad? ¿qué lo hace optar por crear una interfaz?
Creamos una clase abstracta cuando no es posible dar una implementación a alguno de los métodos de la clase. Optamos por una interfaz cuando no se cuenta con lo necesario para implementar cada uno de los métodos no estáticos. También optamos por una interfaz cuando las clases que la implementan ya derivan de alguna otra clase.

g) Un segmento de código Java muestra:
String texto = “UTFSM”;
JLabel label = new JLabel(texto);
…..….. /*Aquí se muestra el label en pantalla y luego*/
texto = “USM”; repaint();
¿Al actualizar texto en la línea previa, cambia lo desplegado por label en pantalla? Justifique.
No cambia. El string pasado en el constructor de JLabel no cambia por la asignación de un nuevo string a texto. En java se pasan referencias. Además los string son inmutables, cualquier invocación de método sobre texto, no cambiará el string originalmente asignado.

h) ¿Qué crítica se hizo a los lenguajes tradicionales no orientados a objetos que condujo a incorporar excepciones en los lenguajes Orientados a Objetos como Java y C++?
Que mezclaban el tratamiento de errores (casos poco frecuentes) con la lógica principal del código (objetivo principal del código), dificultando su claridad y mantención futura.

Segunda Parte, con apuntes (68 minutos)

2.- a) (18 puntos) Complete el código adjunto y las clases que corresponda.

```
import java.util.*;
public abstract class TestFormas {
       public static void main( String[] args) {
         double perimetro =0;
         ArrayList<Forma> formas = new ArrayList<Forma>();
         formas.add(new /* creamos un rectángulo */ );
         formas.add(new / creamos un círculo */ );
         for(Forma f: formas)
            perimetro += f.perimetro();
         System.out.println ("Perímetro = "+perimetro);
       }
}
```

¿Da lo mismo definir Forma como clase o interfaz? Explique

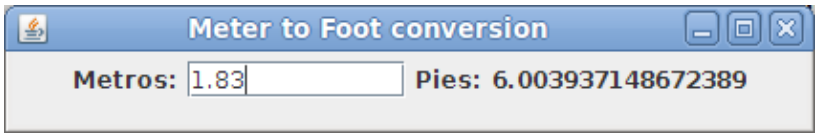
```
import java.util.*;
public abstract class TestFormas {
       public static void main( String[] args) {
             double perimetro =0;
             ArrayList<Forma> formas = new ArrayList<Forma>();
             formas.add(new Rectangulo(0,0, 5,4) );
             formas.add(new Circulo(0,0,4) );
             for(Forma f: formas)
                perimetro += f.perimetro();
             System.out.println ("Perímetro = "+perimetro);
       }
abstract class Forma {
      public abstract double perimetro();
}
class Circulo extends Forma  {
      Circulo (double xx, double yy, double radio){
            x = xx;   y = yy;   r = radio;
      }
      public double perimetro(){
            return 2*Math.PI*r;
      }

      private double x,y,r;
}
class Rectangulo extends Forma {
      Rectangulo (double xx1, double yy1, double xx2, double yy2){
            x1 = xx1;    y1 = yy1;
            x2 = xx2;    y2 = yy2;
      }
      public double perimetro() {
            return 2*(Math.abs(x1-x2)+Math.abs(y1-y2));
      }
      private double x1,y1,x2,y2;
}
```

Sí, en este caso Forma pudo ser interfaz con sintaxis un tanto distinta para Forma y las clases que implementan pero con resultado equivalente. Esto porque el principio de sustitución se cumple entre clases heredadas y clases que implementan una interfaz.

b) (16 puntos) Cree la clase GrupoForma. Esta clase almacena un conjunto de formas, cada una de las cuales puede ser un rectángulo, un círculo o un conjunto de formas. En la clase GrupoForma considere algún método para incorporar formas al conjunto y los necesarios para que instancias de GrupoForma puedan ser incorporadas al ArrayList del código dado.

```
class GrupoForma extends Forma {
      public double perimetro(){
            double p=0;
            for(Forma f: grupo)
                  p += f.perimetro();
            return p;
      }
      void add(Forma f){
            grupo.add(f);
      }
      private ArrayList<Forma> grupo = new ArrayList<Forma>();
}
```

3.- (34 puntos) Desarrolle una aplicación que permita transformar una longitud expresada en metros a su magnitud expresada en pies. La aplicación debe verse como la mostrada.



1 [foot] = 0.3048 [m]

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

public class Meter2Foot extends JFrame {
      M2F_Panel gui = new M2F_Panel();
      public Meter2Foot() {
            setTitle("Meter to Foot conversion");
            setSize(500,80);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            getContentPane().add(gui);
      }
      public static void main( String[] args) {
            Meter2Foot m2f = new Meter2Foot();
            m2f.setVisible(true);
      }
}

class M2F_Panel extends JPanel {
      private static final double METER_TO_FOOT_FACTOR=1/0.3048;
      private JTextField meter = new JTextField(10);
      private JLabel foot = new JLabel();
      private M2F_Listener listener = new M2F_Listener(this);

      public M2F_Panel() {
            add(new JLabel("Metros:"));
            add(meter);
            add(new JLabel("Pies:"));
            add(foot);
            meter.addActionListener(listener);
      }

      public void convert() {
            float m = Float.parseFloat(meter.getText());
            foot.setText(m*METER_TO_FOOT_FACTOR+"");
      }
}

class M2F_Listener implements ActionListener {
      private M2F_Panel gui;
      public M2F_Listener( M2F_Panel guiref) {
            gui = guiref;
      }
      public void actionPerformed( ActionEvent e) {
            gui.convert();
      }
}
```