Segundo Certamen

En este certamen usted no podrá hacer preguntas. Si algo no está claro, haga una suposición razonable, la anota y resuelva conforme a ello.

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

1.- Responda brevemente y entregue en una hoja con su nombre.

a) ¿Cuál es el propósito de crear paquetes en Java?
Los paquetes son un mecanismo para encapsular nombres. Así es posible crear clases cuyos nombres puedan coincidir con el de otras clases pero se diferencian por le paquete al que pertenecen.

b) ¿Qué problema de los lenguajes no orientados a objetos explica la incorporación del manejo de excepciones de los lenguajes orientados a objetos?
En los lenguajes no orientados a objetos las situaciones de error de ejecución deben ser tratadas en cada parte donde éstas puedan ocurrir. Esto extiende el código que cumple la funcionalidades deseadas en muchas líneas preocupadas de las condiciones de error. Como consecuencia, aumenta la dificultad para depurar y mantener el código. Las excepciones permiten separar el código fundamental del tratamiento de todas las condiciones de error.

c) Mencione 3 ventajas de las metodologías de desarrollo de software iterativas e incrementales.
**** Permite generar versiones ejecutables parciales para evaluación del cliente.***
**** Permite identificar prontamente condiciones no especificadas en los requerimientos.***
**** Permite una mejor gestión de proyecto identificando la proporción de casos de uso cubiertos en el tiempo.***

d) En orden mencione las actividades desarrolladas en cada iteración de la metodología iterativa e incremental.
Definición de requerimientos, análisis, diseño, implementación, pruebas y distribución.

e) ¿Cuál es el propósito de los diagramas de secuencia definidos en UML?
Su propósito es mostrar las relaciones dinámicas entre objetos que interactúan.

f) Presente alguna situación donde la sobrecarga de operadores debe hacerse o conviene hacerla vía funciones globales y no vía métodos de una clase.
Cuando no podemos modificar la clase donde se debe implementar la sobrecarga; por ejemplo, cuando deseamos enviar a hacia una salida serial del tipo cout una objeto instancia de una de nuestras clases.

g) Mencione y explique una diferencia de las siguientes dos declaraciones:

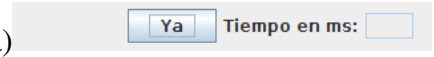
Lenguaje Java:	Lenguaje C++:
Student a;	Student b;

En Java a representa una referencia hacia un objeto Student, en cambio en C++ b es un objeto Student. En Java aún no se ha creado el objeto, en C++ ésta ya es creado usando el constructor sin parámetros.

h) En C++ explique cómo la implementación de un constructor de una clase hija puede invocar a un constructor específico de la clase padre.
Se debe hacer en la lista de inicialización, la cual se pone a continuación de los argumentos del constructor y antes del inicio del método.

Segunda Parte, **con apuntes** (68 minutos)

2.- **Desarrolle un applet** para medir el tiempo de reacción vista – mano.

a)  b) 	Inicialmente el applet debe mostrar algo similar a la figura a). Luego de un tiempo aleatorio el applet muestra la palabra "Alerta!!" ante lo cual el usuario debe presionar el botón "Ya". El applet muestra el tiempo en milisegundos desde que se muestra la Alerta hasta que presiona Ya como se muestra en b)

Nota: Si es de utilidad, considere el uso de:

En clase System: *static long currentTimeMillis()*	*Returns the current time in milliseconds since January 1st, 1970*
En clase Timer: *Timer(int delay, ActionListener listener)*	*Creates a Timer and initializes both the initial delay and between-event delay to delay milliseconds.*
En clase Math *static double random()* *static long round(double a)*	 *Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.* *Returns the closest long to the argument.*

Se presenta aquí una solución sencilla, usando los métodos vistos en clases. La clase Timer permite configurar el disparo del evento sólo una vez. El listener asociado al botón también se pudo configurar al momento de recibir la alerta. Así se consigue inactividad del botón previo a la aparición de "Alerta!!". Por simplicidad estos cambios no están programados en esta solución.

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

public class ReactionApplet extends JApplet {
   public void init() {
      tiempo=new JTextField("       ");
      tiempo.setEditable(false);
      JLabel text = new JLabel ("Tiempo en ms:");
      alerta= new JLabel("       ");
      new Timer(Math.round((int)(Math.random()*5000)+3000),
            new ActionListener (){
               public void actionPerformed(ActionEvent event) {
                  inicio = System.currentTimeMillis();
                  alerta.setText("Alerta!!");
               }
            }).start();
      botonReaccion= new JButton("  Ya  ");
      botonReaccion.addActionListener( new ActionListener() {
            public void actionPerformed(ActionEvent event) {
               tiempo.setText(""+(System.currentTimeMillis()-inicio));
            }
         });
      JPanel panel= new JPanel();
      panel.add(alerta);
```

```
        panel.add(botonReaccion);
        panel.add(text);
        panel.add(tiempo);
        getContentPane().add(panel);
    }

    private JLabel alerta;
    private JButton botonReaccion;
    private JTextField tiempo;
    private long inicio;
}
```

3.- En esta pregunta usted debe desarrollar una clase en C++ para permitir el manejo de dinero. Supongamos que interesa saber la cantidad de dinero en billetes recaudada en cada una de las cajas de un supermercado distinguiendo la cantidad por tipo de billetes. Luego se desea reunir el dinero de todas las cajas y se desea seguir sabiendo cuánto dinero hay por cada tipo de billete.
Cree la clase Dinero en C++ medido en miles de pesos. Esta clase debe distinguir los billetes de 20 mil, 10 mil, 5 mil, 2 mil y un mil pesos. La idea es que programas que usen su clase puedan hacer cosas del tipo:

```
Dinero a (0, 30, 15, 0, 0) ; // a tiene 30 mil pesos en billetes de 10 mil, y 15 mil en de 5 mil.
                             //  a no tiene billetes de 20 mil, 2 mil o 1 mil.
Dinero b (40, 0, 0, 4, 1);  // b tiene 40 mil en billetes de 20 mil, 4 mil en de 2 mil, y un billete de 1 mil.
                            // b no tiene billetes de 10 mil o 5 mil.
Dinero c;  // c no tiene dinero, 0 pesos.
 c = a+b;  // permite acumular en c la cantidad de cada billete de a y b.
 cout << c  // permite mostrar por pantalla la cantidad de billetes de cada tipo. Usted elige el formato.
```

```
#include <iostream>
using namespace std;

class Dinero {
public:
    Dinero () { veinteM = diezM = cincoM = dosM = m = 0; };
    Dinero(int a, int b, int c, int d, int e);
    Dinero operator+ (const Dinero &d) const;
    friend ostream& operator<< (ostream &os, const Dinero &d);
private:
    int veinteM, diezM, cincoM, dosM, m;
};
Dinero::Dinero(int a, int b, int c, int d, int e) {
    veinteM = a;
    diezM = b;
    cincoM = c;
    dosM = d;
    m = e;
}
Dinero Dinero::operator+ (const Dinero &d) const {
```

```
   Dinero tmp (veinteM+d.veinteM, diezM+d.diezM, cincoM+d.cincoM,
            dosM+d.dosM, m+d.m);
   return tmp;
}

ostream& operator<< (ostream &os, const Dinero &d) {
   os << "Veinte mil:" << d.veinteM << "  Diez mil:" << d.diezM;
   os << "  Cinco mil:" << d.cincoM << "  Dos mil:" << d.dosM;
   os << "  Mil:" << d.m << endl;
   return os;
}

int main (void) {
   Dinero a (0, 20, 15, 0, 0) ; // a tiene 20 mil pesos en billetes de 10 mil, y 15 mil en de 5 mil.
     //  a no tiene billetes de 20 mil, 2 mil o 1 mil.
   Dinero b(40, 0, 0, 4, 1);  // b tiene 40 mil en billetes de 20 mil, 4 mil en de 2 mil, y un billete de 1 mil.
     // b no tiene billetes de 10 mil o 5 mil.
   Dinero c;  // c y d no tiene dinero, 0 pesos.
   c = a+b;  // permite acumular en c la cantidad de cada billete de a y b.
   cout << c;
}
```