Diseño y Programación Orientados a Objetos
1er. Semestre 2024

Certamen II ELO-329 Diseño y Programación Orientada a Objetos (1s2024)

Nombre: _____
Paralelo: _____

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 (30 minutos):
Primera pregunta (30 pts.)

A. (5pts) Mencione 2 desafíos o aspectos a tomar en cuenta al desarrollar una aplicación Android para un móvilque no son igualmente críticos o no están presentes al desarrollar una aplicación Java de escritorio.

R:
Las aplicaciones para un móvil deberían **poder adaptarse a tamaños de pantalla y resoluciones variadas**. Las aplicaciones de escritorio varían menos en las resoluciones de pantalla.
El cambio de las capacidades de los equipos móviles nuevos en hardware y software demandan **desarrollos compatibles con plataformas antiguas** para llegar a un alto porcentaje de los usuarios.

B. (5pts) Para cada segmento de código indique si es válido (compila sin errores) o no. En caso de tener error,indique el error.

int a = 8;
int * const p1;
p1 = &a;
	int b;
	const int * p2;
	p2 = &b;
		const int c=10;
		const int * p3;
		p3 = &c;

No es válido, no es correcto declarar un putero constante sin darle un valor inicial inmediatamente. Es un error asignar un nuevo valor a un puntero contante en la última sentencia. Las sentencias 2 y 3 deberían fundirse en una. int * const p1 = &a;	Correcto	Correcto

C. (5pts) Considera el siguiente código en C++

```
class Cuenta {
   private:
      double saldo;
    public:
      Cuenta(double s) : saldo(s) {}
      double obtenerSaldo() {return saldo;}
      friend void actualizarSaldo(Cuenta &cuenta, double nuevoSaldo); // (1)
};
```

```
void actualizarSaldo(Cuenta &cuenta, double nuevoSaldo) {
    cuenta.saldo = nuevoSaldo;
}

int main() {
   Cuenta miCuenta(100.0);
   actualizarSaldo(miCuenta, 150.0);
   std::cout << "Saldo actualizado: " << miCuenta.obtenerSaldo() << std::endl;
   return 0;
}
```

Indique el rol que cumple la línea (1) marcada en este código. ¿Qué ocurre si se omite esa línea?
R:

La línea (1) declara la función global actualizarSaldo(…) como amiga de la clase Cuenta. Esto permite a esta función acceder a miembros privado y protegidos de la clase.

Si se omite esta línea, este programa no compila y arroja error por acceso a miembro privado de Cuenta en función actualizarSaldo(…).

5 puntos: Identificación Correcta del Rol de la Línea (1), Explicación Clara de las Consecuencias de Omitir la Línea (1).

2 puntos: Identificación Parcial del Rol de la Línea (1), Explicación Incompleta de las Consecuencias de Omitir la Línea (1).

D. (5pts) Considera el siguiente código en C++

```
class Recurso {
    private:
      static int contadorRecursos;
      int *datos;
    public:
      Recurso() {
          datos = new int[100];
          contadorRecursos++;
      }
      ~Recurso() {
          delete[] datos;
          contadorRecursos--;
      }
      static int obtenerContadorRecursos() {
          return contadorRecursos;
      }
};

int Recurso::contadorRecursos = 0;
```

```
int main() {
   std::cout << "Contador de recursos: " << Recurso::obtenerContadorRecursos() << std::endl;
   Recurso r1;
   std::cout << "Contador de recursos: " << Recurso::obtenerContadorRecursos() << std::endl;
   {
      Recurso r2;
      std::cout << "Contador de recursos: " << Recurso::obtenerContadorRecursos() << std::endl;
   }
   std::cout << "Contador de recursos: " << Recurso::obtenerContadorRecursos() << std::endl;
   return 0;
}
```

Indique qué valores se muestran por consola luego de ejecutar la función **main**.

R:
Contador de recursos: 0
Contador de recursos: 1
Contador de recursos: 2
Contador de recursos: 1
5 puntos: Identificación Correcta de Todos los Valores Mostrados por Consola, Explicación Clara del Flujo del Programa:
2 puntos: Identificación Parcial de los Valores Mostrados por Consola, Explicación Incompleta del Flujo del Programa:

E. (5pts) En el contexto de manejo de excepciones en C++, explique qué es el método **what()**, dónde está definido y cuál es su uso.

R:
El método what() es una función miembro de la clase base std::exception en C++.
Su propósito es proporcionar una descripción del error que ocurrió. Este método puede ser sobrescrito por clases derivadas para proporcionar mensajes de error más específicos y detallados.

F. (5pts) En el contexto de Ingeniería de Software, describa qué es un caso de uso y qué es una historia deusuario. Indique al menos dos diferencias entre casos de uso e historias de usuarios.

R:
Un caso de uso detalla la interacción entre un actor (usuario o sistema) y el sistema para alcanzar un objetivo específico, con un enfoque en qué hace el sistema.
Por otro lado, una historia de usuario es una breve descripción de una funcionalidad deseada desde la perspectiva del usuario final, centrada en el valor que aporta.
Diferencias:
1) Los casos de uso son más detallados y estructurados, mientras que las historias de usuario son concisas y enfocadas en el valor.
2) Los casos de uso representan situaciones de funcionalidad del programa que deben implementarse, mientras que algunas historias de usuario pueden no implementarse si hay otras que ofrecen más valor.