

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 1

GUÍA DE LABORATORIO

(formato docente)

INFORMACIÓN BÁSICA					
ASIGNATURA:	PROGRAMACIÓN DE SISTEMAS				
TÍTULO DE LA PRÁCTICA:	Programación Orientado a Objetos				
NÚMERO DE PRÁCTICA:	02	AÑO LECTIVO:	2023- A	NRO. SEMESTRE:	V
TIPO DE PRÁCTICA:	INDIVIDUAL				
	GRUPAL	MÍNIMO DE ESTUDIANTES	03	MÁXIMO DE ESTUDIANTES	04
FECHA INICIO:	11/05/2023	FECHA FIN:	17/05/2023	DURACIÓN:	04
RECURSOS Y EQUIPOS A UTILIZAR:					
Linux					
DOCENTE(s):					
Magister Edith Giovanna Cano Mamani Magister Jorge Manuel Polanco Argüelles					

OBJETIVOS/TEMAS Y COMPETENCIAS	
OBJETIVOS:	
<ul style="list-style-type: none"> La presente práctica de laboratorio tiene como objetivo el uso del C++ para la creación de clases, atributos, métodos u operaciones. 	
TEMAS:	
✓ El lenguaje de programación C++: tipos de datos, estructuras, punteros.	
COMPETENCIAS A ALCANZAR	Diseña responsablemente sistemas componentes o procesos para satisfacer necesidades dentro de restricciones realista, económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad
	Construye responsablemente soluciones siguiendo un proceso adecuado llevando a cabo las pruebas ajustadas a los recursos disponibles del cliente
	Aplica la forma flexible técnicas, métodos, principios, normas, estándares y herramientas de ingeniería necesarias para la construcción de software e implementación de sistemas de información

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 2</p>

CONTENIDO DE LA GUÍA

I. MARCO CONCEPTUAL

INTRODUCCIÓN:

El lenguaje C++ fue creado por Bjarne Stroustrup, en el año 1979. Este programador danés empezó a trabajar en el lenguaje, que lo llamaba C con clases.

Stroustrup comparó varios lenguajes para la creación del predecesor de C++. Por ejemplo, vio que Simula (un lenguaje POO del 1962), tenía buenas características para programar, pero era muy lento, y el lenguaje BCPL era rápido, pero de bajo nivel.

Así que mejoró el lenguaje C con características de Simula. Se decidió por el lenguaje C porque:

- es de uso general.
- es rápido.
- es portable.
- es muy utilizado.

Aunque el lenguaje C++ se creó en 1979, tal y como hemos dicho, no es hasta 1983 que tiene su nombre definitivo, pasando de C con clases a C++.

En ese tiempo también se añadieron nuevas características, como la herencia, la sobrecarga de funciones, y las funciones virtuales.

C++ 2.0 se lanzó en 1989, con nuevas opciones como herencia múltiple (los lenguajes actuales de programación orientada a objetos solo permiten una única herencia), clases abstractas, funciones estáticas y muchas más.

La duración del laboratorio (actividades y ejercicios resueltos y propuestos) será de (seis) 6 horas académicas

ACTIVIDADES



- Consideraciones: Instalar un IDE para la programación de C++. Puede ser Zinjal, Visual Studio 2019, entre otros.

- **ACTIVIDAD 1.**

El estudiante debe entender los ejercicios resueltos

- **ACTIVIDAD 2.**

Revisar los ejercicios propuestos creando los archivos respectivos y un informe de todos los programas utilizados y explicados por cada archivo.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 3</p>

II. EJERCICIO RESUELTO

Ejercicio 1. Crear un archivo biblioteca.h que contenga la función que recepciona los parámetros de una suma, crear un archivo biblioteca.cpp que contenga la función suma de dos enteros y retorne su suma, crear un archivo principal que implemente la suma de dos números 1 (uno) utilizando los archivos anteriores

Archivo biblioteca1.h

```
int suma(int,int);
```

Archivo biblioteca1.cpp

```
#include "biblioteca1.h"
```

```
int suma(int a, int b){
    return a+b;
}
```

Archivo principal.cpp

```
#include <iostream>
```

```
#include "biblioteca1.cpp"
```

```
using namespace std;

int main(){
    int resultado = suma(1,1);
    cout << "El resultado es: "<< resultado << "\n";
    return 0;
}
```

Ejercicio 2. Crear un archivo cabecera biblioteca1.h que tenga la función de suma de dos enteros y crear un archivo Principal.cpp que sume los números uno mas uno utilizando dicha biblioteca

Archivo biblioteca1.h

```
int suma(int a, int b){
    return a+b;
}
```

Archivo Principal.cpp



```
#include <iostream>
```

```
#include "biblioteca1.h"
```

```
using namespace std;
```

```
int main(){
    int resultado = suma(1,1);
    cout << "El resultado es: "<< resultado << "\n";
    return 0;
}
```

Ejercicio 3. Crear una clase MyClass con un método foo y una dato entero bar de clase miembro, implementar esa clase en un archivo main.cpp en una variable a.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 4</p>

Archivo myclass.h

```
class MyClass
{
    public:
        void foo();
        int bar;
};
```

Archivo main.cpp

```
#include "myclass.h"
```

```
int main()
{
    MyClass a;
    return 0;
}
```

Ejercicio 4. Implementar una clase persona con los atributos de nombre, id y el método getdetails en un archivo biblioteca.h y en un archivo de implementación Principal.cpp

Archivo biblioteca.h

```
class person
{
    char name[20];
    int id;
    public:
        void getdetails(){}
};
```

Archivo Principal.cpp



```
#include "biblioteca.h"
```

```
int main()
{
    person p1;
    return 0;
}
```

Ejercicio 5. Implementar un programa en c++ que permita el acceso a los atributos de una clase Alumnos utilizando un archivo biblioteca.h para la creación de las clases y un archivo Principal.cpp para la implementación de la clase antes mencionada

Archivo biblioteca.h

```
// Programa en C++ para demostrar
// el acceso a los datos de la clase
#include <bits/stdc++.h>
using namespace std;
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 5</p>

```
class Alumnos
{
// especificando acceso
    public:
        //datos de la clase niembro
        string NombreAlumno;

        //Funciones niembros
        void imprimirnombro()
        {
            cout << "Nombre alumno es: " << NombreAlumno;
        }
};
```



```
Archivo Principal.cpp
#include "biblioteca.h"
int main(){
//Declara un objeto de la clase Alumnos
    Alumnos obj1;

    //accesando a los datos de la clase niembro
    obj1.NombreAlumno = "Abhi";

    //accesando a la funcion de la clase niembro
    obj1.imprimirnombro();
    return 0;
}
```

Ejercicio 6. Implementar la definición de los métodos de una clase en un programa en C++, tanto dentro como fuera de la clase Alumnos, utilizando un archivo biblioteca.h para las clases y un archivo Principal.cpp para las implementaciones

```
Archivo biblioteca.h
//Existen dos formas definir la función de la clase niembro
//Dentro de la definición de la clase
//Fuera de la definición de la clase
//Programa de C++ para demostrar la declaración
//de la función fuera de la clase
#include <bits/stdc++.h>
using namespace std;
class Alumnos
{
    public:
        string NombreAlumno;
        int id;
        //Imprimirnombro no esta definida dentro de la definici3n de la clase
        void Imprimirnombro();
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 6</p>

```

        //Imprimirid esta definida dentro de la definición de la clase
        void Imprimirid()
        {
            cout << "Alumno id es: " << id;
        }
};

```

```

Archivo Principal.cpp
#include "biblioteca.h"
//Definición de Imprimirnombre usando el alcance de operador de resolución
void Alumnos::Imprimirnombre()
{
    cout << "Nombre es: " << NombreAlumno;
}
int main() {
    Alumnos obj1;
    obj1.NombreAlumno = "xyz";
    obj1.id = 15;
    //llamando a Imprimirnombre()
    obj1.Imprimirnombre();
    cout << endl;
    //llamando a imprimirid()
    obj1.Imprimirid();
    return 0;
}

```

Ejercicio 7. Implementar un programa en C++ para demostrar el uso de constructores para la clase Alumnos utilizando un archivo biblioteca.h para la definición de clases y un archivo Principal.cpp para la implementación de dichas clases

```

Archivo biblioteca.h
//Programa en C++ para demostrar los constructores
#include <bits/stdc++.h>
using namespace std;
class Alumnos
{
    public:
    int id;
    //Constructor por defecto
    Alumnos()
    {
        cout << "Constructor por defecto llamado" << endl;
        id = -1;
    }
    //Constructor parametrizado
    Alumnos(int x)
    {

```

```
        cout << "Constructor parametrizado llamado" << endl;
        id = x;
    }
};
```

Archivo Principal.cpp

```
#include "biblioteca.h"
```

```
int main(){
    //el obj1 llamara al constructor por defecto
    Alumnos obj1;
    cout << "Alumno id es: " << obj1.id << endl;
    //obj1 llamara a un constructor parametrizado
    Alumnos obj2(21);
    cout << "Alumno id es: " << obj2.id << endl;
    return 0;
}
```



Ejercicio 8. Implementar un programa en C++ que utiliza destructores para la clase alumnos utilizando un archivo biblioteca.h para las clases y un archivo Principal.cpp para la implementación de las clases presente en el archivo biblioteca.h

Archivo biblioteca.h

```
//Programa en C++ para explicar los destructores
#include <bits/stdc++.h>
using namespace std;
class Alumnos
{
    public:
    int id;
    //Definición de destructor
    ~Alumnos()
    {
        cout << "Destructor llamado por id:" << id << endl;
    }
};
```

Archivo Principal.cpp

```
#include "biblioteca.h"
int main()
{
    Alumnos obj1;
    obj1.id = 7;
    int i = 0;
    while ( i < 5 )
    {
        Alumnos obj2;
        obj2.id=i;
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 8</p>

```

        i++;
    }//El alcance del obj2 finaliza aqui
    return 0;
} //El alcance del obj1 finaliza aqui

```

Ejercicio 9. Utilizando un programa de C++ crear una clase padre que le de herencia a una clase hijo, utilizando los archivos biblioteca.h para las clases y el archivo Principal.cpp para la implementación de las clases de la biblioteca.h

```

Archivo biblioteca.h
//Programa en C++ para demostrar la implementación de
//Herencia
#include <bits/stdc++.h>
using namespace std;
//Clase base
class Padre
{
    public:
        int id_p;
};
//Subclases herederas de la clase padre
class hijo : public Padre
{
    public:
        int id_c;
};

```

```

Archivo Principal.cpp
#include "biblioteca.h"
//función principal
int main()
{
    hijo obj1;
    //Un objeto de clase hijo tien todos los datos de la clase niembro
    //y todas las funciones de la clase niembro del padre
    obj1.id_c = 7;
    obj1.id_p = 91;
    cout << "El id del hijo es: " << obj1.id_c << endl;
    cout << "El id del padre es: " << obj1.id_p << endl;
    return 0;
}

```

Ejercicio 10. Implementar la herencia de clases de una clase Vehiculo hacia una clase carro, utilizando el archivo biblioteca.h para las clases y el archivo Principal.cpp para la implementación del programa

```

Archivo biblioteca.h
//Programa en C++ para explicar

```



```
//una simple herencia
#include <iostream>
using namespace std;
//clase base
class Vehiculo{
    public:
        Vehiculo()
        {
            cout << "Este es un vehÃ-culo" << endl;
        }
};
//La sub clase derivada desde una simple clase base
class Carro : public Vehiculo{
};
```

```
Archivo Principal.cpp
#include "biblioteca.h"
int main()
{
    //Creando un objeto de una subclase
    //que invocarÃ al constructor de la clase base
    Carro obj;
    return 0;
}
```

Ejercicio 11. Implementar un programa en C++ que permita la herencia mÃltiple utilizando la clase Vehiculo, la clase CuatroRuedas y la clase Carro como clase que hereda, en un archivo biblioteca.h escribir las clases a ser implementadas y un archivo Principal.cpp para la implementaci3n de las clases

```
Archivo biblioteca.h
//C++ programa para explicar
//Herencia Multiple
#include <iostream>
using namespace std;
//primera clase base
class Vehiculo{
    public:
        Vehiculo()
        {
            cout << "Este es un vehiculo" << endl;
        }
};
//segunda clase base
class CuatroRuedas {
    public:
        CuatroRuedas()
```

```
        {  
            cout << "Este es un vehiculo de cuatro ruedas" << endl;  
        }  
};  
class Carro : public Vehiculo, public CuatroRuedas {  
  
};
```



Archivo Principal.cpp
#include "biblioteca.h"
int main()
{

//Creando un objeto de la subclase
//que invocará el constructor de la clase base
 Carro obj;
 return 0;
}

Ejercicio 12. Implementar utilizando un programa en C++ la herencia multinivel usando la clase Vehiculo, la clase CuatroRuedas y la clase Carro, usando un archivo biblioteca.h donde se implementarán las clases y un archivo Principal.cpp donde se implementarán las clases

Archivo biblioteca.h

```
//Programa en C++ para implementar  
//La herencia Multinivel  
#include <iostream>  
using namespace std;  
//Clase base  
class Vehiculo  
{  
    public:  
        Vehiculo()  
        {  
            cout << "Esto es un vehiculo" << endl;  
        }  
};  
//Primera subclase derivada de la clase vehiculo  
class CuatroRuedas : public Vehiculo  
{  
    public:  
        CuatroRuedas()  
        {  
            cout << "Objeto con cuatro ruedas son vehiculos" << endl;  
        }  
};
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 11</p>

//Sub clase derivada de la clase base CuatroRuedas

```
class Carro : public CuatroRuedas {
    public:
        Carro()
        {
            cout << "Carro tiene 4 ruedas" << endl;
        }
};
```



Archivo Principal.cpp

```
#include "biblioteca.h"
int main()
{
    //creando objetos de la subclase que
    //invocará el constructor de la clase base
    Carro obj;
    return 0;
}
```

Ejercicio 13. Utilizando un programa en C++ implementar la herencia jerarquica utilizando la clase Vehiculo, la clase Bus y la clase Carro, utilizando el archivo biblioteca.h para las clases a ser implementadas y el archivo Principal.cpp para la implementación de dichas clases

Archivo biblioteca.h

```
//Programa en C++ para implementar
//La herencia jerarquica
#include <iostream>
using namespace std;
//Clase base
class Vehiculo
{
    public:
        Vehiculo()
        {
            cout << "Esto es un vehiculo" << endl;
        }
};
//Primera sub clase
class Carro : public Vehiculo
{
};
//Segunda sub clase
class Bus : public Vehiculo
{
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 12</p>

```
};
```

Archivo Principal.cpp

```
#include "biblioteca.h"
```

```
int main()
```

```
{
```

```
    //creando un objeto de la subclase
```

```
    // que invocara al constructor de la clase base
```

```
    Carro obj1;
```

```
    Bus obj2;
```

```
    return 0;
```

```
}
```

Ejercicio 14. Implementar la herencia hibrida utilizando un programa en C++ para la clase Vehiculo, clase Faro, la clase Carro y la clase Bus, utilizando un archivo biblioteca.h para las clases y un archivo Principal.cpp para la implementación de las clases creadas en biblioteca.h

Archivo biblioteca.h

```
//Programa en C++ para la herencia hibrida
```

```
#include <iostream>
```

```
using namespace std;
```

```
//clase base
```

```
class Vehiculo
```

```
{
```

```
    public:
```

```
        Vehiculo()
```

```
        {
```

```
            cout << "Este es un vehiculo" << endl;
```

```
        }
```

```
};
```

```
//clase base
```

```
class Faro
```

```
{
```

```
    public:
```

```
        Faro()
```

```
        {
```

```
            cout << "Faro del vehiculo\n";
```

```
        }
```

```
};
```

```
//primera sub clase
```

```
class Carro : public Vehiculo
```

```
{
```

```
};
```

```
//segunda sub clase
```

```
class Bus : public Vehiculo, public Faro
```

```
{
```

```
};
```

Archivo Principal.cpp

```
#include "biblioteca.h"
```

```
int main()
```

```
{
```

```
    //creando objeto de la sub clase que
```

```
    // invocar el constructor de la clase base
```

```
    Bus obj2;
```

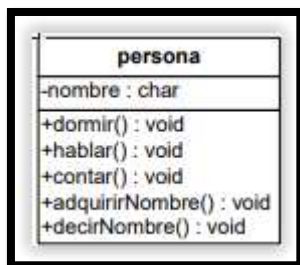
```
    return 0;
```

```
}
```

III. EJERCICIOS PROPUESTOS

- EJERCICIO 1.**

Implemente en C++ la siguiente clase. Considere en su implementación sus propiedades y métodos. La clase de ser implementada en un archivo cabecera (.h), los métodos en un archivo .cpp y el programa principal que contiene al main en otro archivo .cpp. Explicar cada línea de código en sus referencias.





La clase persona consta de una propiedad o dato miembro, y cinco métodos o funciones.

- EJERCICIO 2.**

Se tiene la siguiente definición de la clase Tanque:

```

class Tanque {
    double contenido;
public:
    static const int capacidad = 5000.0;
    Tanque();
    double getContenido() const;
    void agregar(double cantidad); // no superar la capacidad
  
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 14</p>

```
void sacar(double cantidad); // no sacar más de lo que hay
};
```



- a) Complete escribiendo el código de métodos y constructor.
- b) Añada un método sacaMitad() que elimina la mitad del contenido del tanque. Si el tanque está vacío no hace nada.
- c) Escriba una función main que instancie y utilice un objeto t de la clase Tanque. Agregue 100 unidades al tanque. Añada código que contenga un bucle de tipo while, con sentencias que utilicen repetidamente el método sacaMitad() para reducir el contenido del tanque. La iteración deberá detenerse cuando el contenido del tanque sea menor a 1.0.

• EJERCICIO 3.

- a) En una carrera de velocidad participan un cierto número de atletas, cada uno de ellos tiene un nombre, número, nacionalidad y el tiempo que le tomó correr la carrera. Cree una clase Atleta que represente a un atleta. Para ensayar esta clase, una función main(), que haga lo siguiente: provea información acerca de dos atletas, cree dos objetos Atleta e imprima el nombre del más rápido.
- b) Una carrera cubre una cierta distancia y cada carrera tiene un ganador. Cree una clase Carrera que maneje esta información. Ajuste la función main para crear un objeto de tipo Carrera y registre el nombre del atleta más rápido de la carrera.
- c) Suponga que desea tener acceso a más información acerca del ganador de la carrera (por ejemplo, el tiempo que tardó o su nacionalidad). No podemos hacer esto desde el objeto carrera directamente. Necesitamos cambiar la definición de la clase tal que, en lugar de almacenar el nombre del ganador, almacene un objeto Atleta. Haga este cambio muestre todos los detalles del ganador de una carrera.
- d) Ahora deseamos registrar la información de todos los participantes en la carrera, podemos hacer esto añadiendo un atributo competidor a la clase Carrera, que será un array de atletas. Edite la función que determina el ganador de tal forma que ahora lo haga mirando los tiempos de cada competidor. Pruebe estos cambios.

• EJERCICIO 4.

- a) Un vagón de un tren tiene 40 asientos, cada uno de ellos puede estar ocupado o vacante. El vagón puede ser de primera o segunda clase. Cree una clase Carriage para representar esta información. En el constructor se supondrá que todos los asientos inicialmente están vacantes. Escriba los métodos apropiados de acceso y actualización y un método que vaya ocupando los asientos de la siguiente forma: si el vagón es de primera hay un 10% de probabilidad que los asientos sean ocupados; si es de segunda clase hay un 40% de probabilidad que los asientos sean ocupados. Escriba una función main() que contenga un objeto Carriage, llénelo aleatoriamente e imprima el estado de cada asiento.
- b) Un tren consta de un cierto número de vagones, tiene una estación de partida y una de llegada, un cierto precio para los tickets de primera y otro para los de segunda. Cree una clase Train que contenga esta

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 15</p>

información. Añada una función que llene los asientos de los vagones aleatoriamente. Cree un método que calcule el total de ventas de tickets. Ajuste su función main para probar lo hecho.

• EJERCICIO 5.

Se desea modelar información acerca de salas para eventos. La capacidad de los salones depende de la superficie y el costo del alquiler depende de la superficie y si están equipados o no.

- a) Defina la clase Sala con campos miembro nombre, ancho, largo y equipado. Incluir constructores.
- b) Desarrolle una función que permita obtener la capacidad de la sala. Se considera necesario 1.5 m² para cada asistente.
- c) Desarrolle un método que calcule el valor del alquiler: \$45 el m² si la sala está equipada y \$32 el m² si no lo está.
- d) Instancie tres objetos de la clase Sala, con las siguientes características:
 1. Sala1: 12 m de largo por 8 m de ancho, equipada.
 2. Sala2: 20 m de largo por 18 m de ancho, sin equipar.
 3. Sala3: 15 m de largo por 12 m de ancho, equipada.
- e) Imprima en pantalla los datos de la primera sala (campos miembros, superficie, capacidad y costo).
- f) Para los próximos 4 eventos se esperan 200, 50, 100 y 150 asistentes respectivamente. Para ello instancie un arreglo y recórralo con un bucle for imprimiendo en cada iteración el nombre de las posibles salas para realizar el evento según su capacidad, independientemente de que si están equipadas o no.

• EJERCICIO 6.

Escriba el código de una clase Candado que modele un candado con combinación numérica (como los usados para equipaje de viaje). Al utilizar por primera vez dicho candado, puede programársele un número de 3 dígitos del 0 al 9 (deberá almacenarlos en un array) que, será la combinación de seguridad.

La clase Candado deberá almacenar también información (en forma de array) del estado actual de los tres dígitos de la combinación (que en un dado instante podrá o no coincidir con los 3 dígitos programados). Agregar un constructor que inicialice ambos arrays.

Deberá añadir además los siguientes métodos:

- Un método que permita alterar alguno de los 3 dígitos (indicar posición) de la combinación actual.
- Un método, puedeAbrir, que retornará una variable booleana de valor true en caso que la cerradura pueda abrirse y, false en caso contrario.

- Un método `mismaCombinacionActual` que retornará una variable booleana de valor `true` en el caso que la combinación actual del objeto con el cual se invoca este método coincida con la combinación actual de otro Candado (deberá pasarlo como argumento al invocarlo).

Deberá escribir una función `main` donde creará dos objetos de tipo Candado, `c1` y `c2`. Deberá cambiar uno de los tres dígitos de la combinación actual de `c1`, mostrar por pantalla: un mensaje que indique si con la combinación actual de `c2` que programó se puede abrir o no y otro indicando si `c1` y `c2` tienen (o no) programados las mismas combinaciones actuales.

IV. CUESTIONARIO



1. ¿La programación orientada a objetos utiliza abstracción?
2. ¿Las clases son elementos abstractos?

V. REFERENCIAS Y BIBLIOGRAFÍA RECOMENDADAS:

- [1] P.J. Deitel and H.M. Deitel, "Cómo Programar en C++", México, Ed. Pearson Educación, 2009
- [2] B. Stroustrup, "El Lenguaje de Programación C++", Madrid, Adisson Pearson Educación, 2002
- [3] B. Eckel, "Thinking in C++", Prentice Hall, 2000

TÉCNICAS E INSTRUMENTOS DE EVALUACIÓN

TÉCNICAS: <i>Trabajo</i>	INSTRUMENTOS: <i>Rubricas</i>			
CRITERIOS DE EVALUACIÓN Y LOGROS ALCANZADOS				
Criterios	Aprendizaje Alto	Aprendizaje Bueno	Aprendizaje Medio	Aprendizaje Bajo
Planificar el tratamiento del problema	Se planifica completa y adecuadamente el tratamiento del problema	Se planifica completamente el tratamiento del problema con observaciones en acápites	Se planifica parcialmente el tratamiento del problema	No planifica el tratamiento del problema
Organizar el trabajo del equipo/grupo	Se organiza el trabajo del equipo completa y adecuadamente	Se organiza el trabajo del equipo completamente	Se organiza el trabajo del equipo parcialmente	No se organiza el trabajo del equipo
Problema	Se describe, identifica, enuncia, determinando las causas y efectos de manera muy claro y puntual del problema	Se describe, identifica, enuncia, determinando las causas y efectos de manera bien claro y puntual el problema	Se describe, y/o identifica, y/o enuncia, y/o determina las causas y efectos de manera parcial el	No se trata el problema que se va a investigar

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 17

Marco Teórico	Los estudiantes saben cómo buscar, organizar y compartir la información nueva, muy bien sobre el problema	Los estudiantes saben cómo buscar, organizar y compartir la información nueva, bien sobre el problema	Los estudiantes saben cómo buscar, organizar y compartir la información nueva, regular sobre el problema	Los estudiantes no saben cómo buscar, organizar y compartir la información nueva, sobre el problema	
Comparativa de la selección de aspectos	Se compara muy bien los aspectos de productos u otros elementos	Se compara bien los aspectos de productos u otros elementos	Se compara parcialmente los aspectos de productos u otros elementos	No se compara los aspectos de productos u otros elementos	
Trabajar en grupo, colaborativamente con compañeros evitando trabajar solo	Los estudiantes trabajan muy bien colaborativamente con sus compañeros reflejándose en la lista de actividades realizadas compartiendo responsabilidades	Los estudiantes trabajan bien colaborativamente con sus compañeros reflejándose en la lista de actividades realizadas compartiendo responsabilidades	Los estudiantes trabajan regular colaborativamente con sus compañeros reflejándose en la lista de actividades realizadas compartiendo responsabilidades	Los estudiantes No trabajan colaborativamente con sus compañeros reflejándose en la pobre lista de actividades realizadas	
Generación de posibles soluciones (Alternativas)	Se genera muy bien propuestas de posibles soluciones	Se genera bien propuestas de posibles soluciones	Se genera regular propuestas de soluciones	No se plantea propuestas de soluciones	
Presentación de la Solución	La presentación de la solución está muy bien elaborada con calidad en su contenido, en el medio adecuado, exponiendo los resultados con la expresión oral apropiada	La presentación de la solución está bien elaborada con calidad en su contenido, en el medio adecuado, exponiendo los resultados con la expresión oral apropiada	La presentación de la solución está regularmente elaborada con calidad media en su contenido, en el medio adecuado, exponiendo los resultados con la expresión oral media	La presentación de la solución No está elaborada de acuerdo a lo esperado de tal	
Prototipo o Análisis Situacional	El prototipo o análisis situacional como resultado de resolver un problema, evidencia muy bien la alternativa de solución del problema	El prototipo o análisis situacional como resultado de resolver un problema, evidencia bien la alternativa de solución del problema	El prototipo o análisis situacional como resultado de resolver un problema, evidencia de forma regular la alternativa de solución del problema	El prototipo o análisis situacional como resultado de resolver un problema, No evidencia la alternativa de solución del problema	
Lecciones Aprendidas	Las lecciones aprendidas están muy bien redactadas guardando relación entre el resultado y	Las lecciones aprendidas están bien redactadas guardando relación entre el	Las lecciones aprendidas están de forma regular redactadas guardando regular	No se redactan las lecciones aprendidas guardando relación entre el	

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p align="center">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 18</p>

Conclusiones	Se redactan muy bien de manera clara y en concordancia a los objetivos planteados del problema	Se redactan bien de manera clara y en concordancia a los objetivos planteados del problema	Se redactan de forma regular, con poca claridad y en concordancia a los objetivos planteados del problema	Las conclusiones No se redactan de forma clara y en concordancia a los objetivos planteados del problema.	
Referencias	Las referencias se redactan muy bien utilizando uno de los estilos propuestos	Las referencias se redactan bien utilizando uno de los estilos propuestos	Las referencias se redactan de forma regular utilizando uno de los estilos propuestos	Las referencias No se redactan utilizando uno de los estilos propuestos	
Anexos	Se incluyen los anexos pertinentes que sustentan muy bien el informe	Se incluyen los anexos pertinentes que sustentan bien el informe	Se incluyen los anexos pertinentes que sustentan de forma regular el informe	No se incluyen los anexos que sustenten el informe	
Informe	El informe está muy bien elaborado, comprensible, reflejando el trabajo y resultados alcanzados con coherencia y estructura	El informe está bien elaborado, y/o comprensible, y/o reflejando el trabajo y/o resultados alcanzados con coherencia y/o estructura	El informe está regularmente elaborado, y/o poco comprensible, y/o poco refleja el trabajo y/o poco resultados alcanzados con poca coherencia y/o poca estructura	El informe No está elaborado conforme a los planteamientos establecidos	
Autoevaluación	La autoevaluación difiere en el rango de desviación del 5%	La autoevaluación difiere en el rango de desviación del 10%	La autoevaluación difiere en el rango de desviación del 15%	La autoevaluación difiere en el rango de desviación del 20% ó más	