


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

INFORME DE LABORATORIO

(formato estudiante)



INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Fundamentos de la programación 2</i>				
TÍTULO DE LA PRÁCTICA:					
NÚMERO DE PRÁCTICA:	<i>04</i>	AÑO LECTIVO:	<i>2024</i>	NRO. SEMESTRE:	<i>2</i>
FECHA DE PRESENTACIÓN	<i>12/10/2024</i>	HORA DE PRESENTACIÓN	<i>17/00/00</i>		
INTEGRANTE (s)	<i>Leonardo Juan José Baca Calsin</i>			NOTA (0-20)	
DOCENTE(s):				<i>Pinto Oppe Lino José</i>	

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p><i>El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.</i></p> <p><i>El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado</i></p> <ol style="list-style-type: none"> 1. Cree un Proyecto llamado Laboratorio4 2. Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio 3 3. Completar el Código de la clase DemoBatalla

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

Clase Nave:

```
public class Nave {  
  
    private String nombre;  
    private int fila;  
    private String columna;  
    private boolean estado;  
    private int puntos;  
  
    // Metodos mutadores  
    public void setNombre( String n){  
        nombre = n;  
    }  
  
    public void setFila(int f){  
        fila = f;  
    }  
  
    public void setColumna(String c){  
        columna = c;  
    }  
  
    public void setEstado(boolean e){  
        estado = e;  
    }  
  
    public void setPuntos(int p){  
        puntos = p;  
    }  
    // Metodos accesorios  
    public String getNombre(){  
        return nombre;  
    }  
  
    public int getFila(){  
        return fila;  
    }  
}
```

	<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-PRLE-001	Página: 4

```

public String getColumna(){
    return columna;
}

public boolean getEstado(){
    return estado;
}

public int getPuntos(){
    return puntos;
}

public String toString(){
    return "\nNombre: "+getNombre()+"\nFila: "+getColumna()+"\nColumna: "+getColumna()+"\nEstado: "+getEstado()+"\nPuntos: "+getPuntos();
}

// Completar con otros métodos necesarios
}

```

Clase DemoBatalla:

```
import java.util.*;
public class DemoBatalla {
    Run | Debug
    public static void main(String[] args) {
        Nave[] misNaves = new Nave[8];
        Scanner nay = new Scanner(System.in);
        String nombre;
        String col;
        int fil, punt;
        boolean est;
        for (int i = 0; i < misNaves.length; i++) {
            System.out.println("Nave " + (i + 1));
            System.out.print(s:"Nombre: ");
            nombre = nay.next();
            System.out.print(s:"Fila: ");
            fil = nay.nextInt();
            System.out.print(s:"Columna: ");
            col = nay.next();
            System.out.print(s:"Estado (true/false): ");
            est = nay.nextBoolean();
            System.out.print(s:"Puntos: ");
            punt = nay.nextInt();

            misNaves[i] = new Nave();
            misNaves[i].setNombre(nombre);
            misNaves[i].setFila(fil);
            misNaves[i].setColumna(col);
            misNaves[i].setEstado(est);
            misNaves[i].setPuntos(punt);
        }
    }
}
```

```
System.out.println(x:"\nNaves creadas:");
mostrarNaves(misNaves);
mostrarPorNombre(misNaves);
mostrarPorPuntos(misNaves);
System.out.println("\nNave con mayor número de puntos: " + mostrarMayorPuntos(misNaves));

System.out.print(s:"Ingrese el nombre de la nave a buscar: ");
nombre = nay.next();
int pos = busquedaLinealNombre(misNaves, nombre);
if (pos == -1) {
    System.out.println(x:"Nave no encontrada");
} else {
    System.out.println("Nave encontrada: " + misNaves[pos]);
}

ordenarPorPuntosBurbuja(misNaves);
mostrarNaves(misNaves);
ordenarPorNombreBurbuja(misNaves);
mostrarNaves(misNaves);

System.out.print(s:"Ingrese el nombre de la nave para búsqueda binaria: ");
String nombreBuscado = nay.next();
pos = busquedaBinariaNombre(misNaves, nombreBuscado);
if (pos == -1) {
    System.out.println(x:"Nave no encontrada");
} else {
    System.out.println("Nave encontrada: " + misNaves[pos]);
}
```

```
ordenarPorPuntosSeleccion(misNaves);
mostrarNaves(misNaves);
ordenarPorNombreSeleccion(misNaves);
mostrarNaves(misNaves);
ordenarPorPuntosInsercion(misNaves);
mostrarNaves(misNaves);
ordenarPorNombreInsercion(misNaves);
mostrarNaves(misNaves);
```

```
}
```

Métodos:

```
// Método para mostrar todas las naves
public static void mostrarNaves(Nave[] flota) {
    for (Nave nave : flota) {
        System.out.println("Nave: " + nave.getNombre() + ", Puntos: " + nave.getPuntos());
    }
}
```

```
// Método para mostrar todas las naves de un nombre que se pide por teclado
public static void mostrarPorNombre(Nave[] flota) {
    Scanner nay = new Scanner(System.in);
    System.out.print(s:"Ingrese el nombre de una Nave que desea buscar en la flota: ");
    String nombreBuscar = nay.next();
    boolean encontrado = false;

    for (Nave nave : flota) {
        if (nave.getNombre().equals(nombreBuscar)) {
            System.out.println("Nave encontrada: " + nave);
            encontrado = true;
            break; // Salir del bucle si se encuentra
        }
    }

    if (!encontrado) {
        System.out.println(x:"Nave no encontrada");
    }
}
```

```
// Método para mostrar todas las naves con un número de puntos inferior o igual
// al número de puntos que se pide por teclado
public static void mostrarPorPuntos(Nave[] flota) {
    Scanner nay = new Scanner(System.in);
    System.out.print(s:"Ingrese una cantidad de puntos para buscar en la flota: ");
    int puntosBuscar = nay.nextInt();

    boolean encontrado = false;
    for (Nave nave : flota) {
        if (nave.getPuntos() <= puntosBuscar) {
            System.out.println("Nave: " + nave.getNombre() + ", Puntos: " + nave.getPuntos());
            encontrado = true;
        }
    }

    if (!encontrado) {
        System.out.println("No hay naves con puntos menores o iguales a " + puntosBuscar);
    }
}
```

```
// Método que devuelve la Nave con mayor número de Puntos
public static Nave mostrarMayorPuntos(Nave[] flota) {
    Nave mayorNave = flota[0];
    for (Nave nave : flota) {
        if (nave.getPuntos() > mayorNave.getPuntos()) {
            mayorNave = nave;
        }
    }
    return mayorNave;
}
```

```
// Método para buscar la primera nave con un nombre que se pidió por teclado
public static int busquedaLinealNombre(Nave[] flota, String s) {
    for (int i = 0; i < flota.length; i++) {
        if (flota[i].getNombre().equals(s)) {
            return i; // Retorna la posición de la nave encontrada
        }
    }
    return -1; // Si no se encuentra, retorna -1
}
```

```
// Método que ordena por número de puntos de menor a mayor
public static void ordenarPorPuntosBurbuja(Nave[] flota) {
    Nave aux;
    for (int i = 0; i < flota.length - 1; i++) {
        for (int j = 0; j < flota.length - 1 - i; j++) {
            if (flota[j].getPuntos() > flota[j + 1].getPuntos()) {
                // Intercambiar las naves completas
                aux = flota[j];
                flota[j] = flota[j + 1];
                flota[j + 1] = aux;
            }
        }
    }
}
```



```
// Método que ordena por nombre de A a Z
public static void ordenarPorNombreBurbuja(Nave[] flota) {
    Nave aux;
    for (int i = 0; i < flota.length - 1; i++) {
        for (int j = 0; j < flota.length - 1 - i; j++) {
            if (flota[j].getNombre().compareTo(flota[j + 1].getNombre()) > 0) {
                // Intercambiar las naves completas
                aux = flota[j];
                flota[j] = flota[j + 1];
                flota[j + 1] = aux;
            }
        }
    }
}
```

```
// Método para buscar la primera nave con un nombre que se pidió por teclado
public static int busquedaBinariaNombre(Nave[] flota, String nombreBuscado) {
    ordenarPorNombreBurbuja(flota); // Asegurarse de que el arreglo está ordenado
    int izquierda = 0;
    int derecha = flota.length - 1;

    while (izquierda <= derecha) {
        int medio = (izquierda + derecha) / 2;
        int comparacion = flota[medio].getNombre().compareTo(nombreBuscado);

        if (comparacion == 0) {
            return medio; // Se encontró el nombre
        } else if (comparacion < 0) {
            izquierda = medio + 1; // Buscar en la mitad derecha
        } else {
            derecha = medio - 1; // Buscar en la mitad izquierda
        }
    }

    return -1; // No se encontró el nombre
}
```

```
// Método que ordena por número de puntos de menor a mayor
public static void ordenarPorPuntosSeleccion(Nave[] flota) {
    int n = flota.length;

    for (int i = 0; i < n - 1; i++) {
        int indiceMin = i;
        for (int j = i + 1; j < n; j++) {
            if (flota[j].getPuntos() < flota[indiceMin].getPuntos()) {
                indiceMin = j;
            }

            // Intercambiar la nave con el mínimo valor con la nave en la posición i
            if (i != indiceMin) {
                Nave temp = flota[i];
                flota[i] = flota[indiceMin];
                flota[indiceMin] = temp;
            }
        }
    }
}
```

```
// Método que ordena por nombre de A a Z
public static void ordenarPorNombreSeleccion(Nave[] flota) {
    int n = flota.length;

    for (int i = 0; i < n - 1; i++) {
        int indiceMin = i;
        for (int j = i + 1; j < n; j++) {
            if (flota[j].getNombre().compareTo(flota[indiceMin].getNombre()) < 0) {
                indiceMin = j;
            }
        }

        // Intercambiar la nave con el mínimo valor con la nave en la posición i
        if (i != indiceMin) {
            Nave temp = flota[i];
            flota[i] = flota[indiceMin];
            flota[indiceMin] = temp;
        }
    }
}
```

```
// Método que ordena por número de puntos de menor a mayor
public static void ordenarPorPuntosInsercion(Nave[] flota) {
    for (int i = 1; i < flota.length; i++) {
        Nave clave = flota[i];
        int j = i - 1;

        while (j >= 0 && flota[j].getPuntos() > clave.getPuntos()) {
            flota[j + 1] = flota[j];
            j--;
        }
        flota[j + 1] = clave;
    }
}
```

```
// Método que ordena por nombre de A a Z
public static void ordenarPorNombreInsercion(Nave[] flota) {
    for (int i = 1; i < flota.length; i++) {
        Nave clave = flota[i];
        int j = i - 1;

        while (j >= 0 && flota[j].getNombre().compareTo(clave.getNombre()) > 0) {
            flota[j + 1] = flota[j];
            j--;
        }
        flota[j + 1] = clave;
    }
}
```

EJECUCIÓN DEL PROGRAMA

Nave 1
Nombre: Juan
Fila: 1
Columna: A
Estado (true/false): true
Puntos: 1
Nave 2
Nombre: Jose
Fila: 2
Columna: B
Estado (true/false): false
Puntos: 2



Nave 3
Nombre: Leo
Fila: 3
Columna: C
Estado (true/false): true
Puntos: 3
Nave 4
Nombre: Pepe
Fila: 4
Columna: D
Estado (true/false): true
Puntos: 4

Nave 5
Nombre: Fausto
Fila: 5
Columna: E
Estado (true/false): false
Puntos: 5
Nave 6
Nombre: Isaac
Fila: 6
Columna: F
Estado (true/false): true
Puntos: 6

```
Nave 7
Nombre: Jacob
Fila: 7
Columna: G
Estado (true/false): false
Puntos: 7
Nave 8
Nombre: Job
Fila: 8
Columna: H
Estado (true/false): true
Puntos: 8
```

```
Naves creadas:
Nave: Juan, Puntos: 1
Nave: Jose, Puntos: 2
Nave: Leo, Puntos: 3
Nave: Pepe, Puntos: 4
Nave: Fausto, Puntos: 5
Nave: Isaac, Puntos: 6
Nave: Jacob, Puntos: 7
Nave: Job, Puntos: 8
```

```
Ingrese el nombre de una Nave que desea buscar en la flota: Pepe
Nave encontrada:
Nombre: Pepe
Fila: D
Columna: D
Estado: true
Puntos: 4
```

	<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-PRLE-001</p>	<p>Página: 14</p>

```

Ingrese una cantidad de puntos para buscar en la flota: 3
Nave: Juan, Puntos: 1
Nave: Jose, Puntos: 2
Nave: Leo, Puntos: 3



Nave con mayor número de puntos:
Nombre: Job
Fila: H
Columna: H
Estado: true
Puntos: 8

```

```

Ingrese el nombre de la nave a buscar: Juan
Nave encontrada:
Nombre: Juan
Fila: A
Columna: A
Estado: true
Puntos: 1
Nave: Juan, Puntos: 1
Nave: Jose, Puntos: 2
Nave: Leo, Puntos: 3
Nave: Pepe, Puntos: 4
Nave: Fausto, Puntos: 5
Nave: Isaac, Puntos: 6
Nave: Jacob, Puntos: 7
Nave: Job, Puntos: 8
Nave: Fausto, Puntos: 5
Nave: Isaac, Puntos: 6
Nave: Jacob, Puntos: 7
Nave: Job, Puntos: 8
Nave: Jose, Puntos: 2
Nave: Juan, Puntos: 1
Nave: Leo, Puntos: 3
Nave: Pepe, Puntos: 4



```

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

```

Ingrese el nombre de la nave para búsqueda binaria: Leo
Nave encontrada:
Nombre: Leo
Fila: C
Columna: C
Estado: true
Puntos: 3
Nave: Juan, Puntos: 1
Nave: Jose, Puntos: 2
Nave: Leo, Puntos: 3
Nave: Pepe, Puntos: 4
Nave: Fausto, Puntos: 5
Nave: Isaac, Puntos: 6
Nave: Jacob, Puntos: 7
Nave: Job, Puntos: 8
Nave: Fausto, Puntos: 5
Nave: Isaac, Puntos: 6
Nave: Jacob, Puntos: 7
Nave: Job, Puntos: 8
Nave: Jose, Puntos: 2
Nave: Juan, Puntos: 1
Nave: Leo, Puntos: 3
Nave: Pepe, Puntos: 4
Nave: Juan, Puntos: 1
Nave: Jose, Puntos: 2
Nave: Leo, Puntos: 3
Nave: Pepe, Puntos: 4
Nave: Fausto, Puntos: 5
Nave: Isaac, Puntos: 6
Nave: Jacob, Puntos: 7
Nave: Job, Puntos: 8
Nave: Fausto, Puntos: 5
Nave: Isaac, Puntos: 6
Nave: Jacob, Puntos: 7
Nave: Job, Puntos: 8
Nave: Jose, Puntos: 2
Nave: Juan, Puntos: 1

```

	<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-PRLE-001</p>	<p>Página: 16</p>

Nave: Leo, Puntos: 3
Nave: Pepe, Puntos: 4

COMMIT:

```
LENOVO@DESKTOP-MAP07Q4 MINGW64 ~ (master)
$ ls
'3D Objects'      Desktop          'Menú Inicio'
Alice3            Documents       'Mis documentos'
AppData           Downloads       Music
AppMods           'Entorno de red' NTUSER.DAT
'Configuración local' Favorites        NTUSER.DAT{d2e18ee0-a5c2-11ee-af53-fc8268c6
Contacts          IdeaProjects    NTUSER.DAT{d2e18ee0-a5c2-11ee-af53-fc8268c6
Cookies           Impresoras      NTUSER.DAT{d2e18ee0-a5c2-11ee-af53-fc8268c6
'Datos de programa' Links            OneDrive

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~ (master)
$ cd Contacts

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts (master)
$ ls
LABORATORIO3/  LABORATORIO4/  desktop.ini

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts (master)
$ cd LABORATORIO4

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ ls
DemoBatalla.java  Nave.java



LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git init
Initialized empty Git repository in C:/Users/LENOVO/Contacts/LABORATORIO4/.git/

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    DemoBatalla.java
    Nave.java

nothing added to commit but untracked files present (use "git add" to track)
```


	<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-PRLE-001</p>	<p>Página: 17</p>

```
nothing added to commit but untracked files present (use "git add" to track)
LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git add DemoBatalla.java

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git add Nave.java

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   DemoBatalla.java
        new file:   Nave.java

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git config user.name "LeoMotoMoto200"

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git config user.email "lbacac@unsa.edu.pe"

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git commit -m "Mi primer commit en lab04"
[master (root-commit) 2a307cb] Mi primer commit en lab04
 2 files changed, 311 insertions(+)
 create mode 100644 DemoBatalla.java
 create mode 100644 Nave.java

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git remote add origin https://github.com/LeoMotoMoto200/Laboratorios.git

LENOVO@DESKTOP-MAP07Q4 MINGW64 ~/Contacts/LABORATORIO4 (master)
$ git push -u origin master
```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?



Comprobé mi código, ingresando cadenas de nombres, booleanos y números para los puntos y fila.

¿Qué resultado esperabas obtener para cada valor de entrada?

Esperaba que el Usuario vea la Nave a la cual le está asignando los atributos de la clase.

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

Para cada uno obtuve la respuesta esperada, es decir, me imprimía el nombre de la Nave, su fila, sus puntos, su estado y su columna.

	<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-PRLE-001	Página: 18

III. RUBRICA:

Colocar la evidencia de las respuestas realizadas al cuestionario enunciado en la guía práctica de laboratorio.

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	

6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
TOTAL		20		17	

CONCLUSIONES

Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.

Me gustaron mucho los arreglos con Objetos, ya que con estos podemos almacenar información de objetos, como personas, alumnos, soldados, naves, etc. Esto es beneficioso al momento de controlar los datos, para una aplicación o para un sitio web, gracias a esto, en un futuro, se nos hará más fácil almacenar datos, de una manera ordenada, como en el caso d los arreglos.

METODOLOGÍA DE TRABAJO

Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.

Para este trabajo, la metodología de trabajo que se ha utilizado es la siguiente: Primero entendimos el enunciado del problema, después comenzamos a pasar la lógica a un lenguaje de programación, en este caso, al lenguaje Java, y después de terminar el programa, comenzamos con las pruebas de compilación.

REFERENCIAS Y BIBLIOGRAFÍA

Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE

<https://repositorio.unsa.edu.pe/bitstreams/c4eb8421-6910-46fd-9746-e6c07c0b59bb/download>