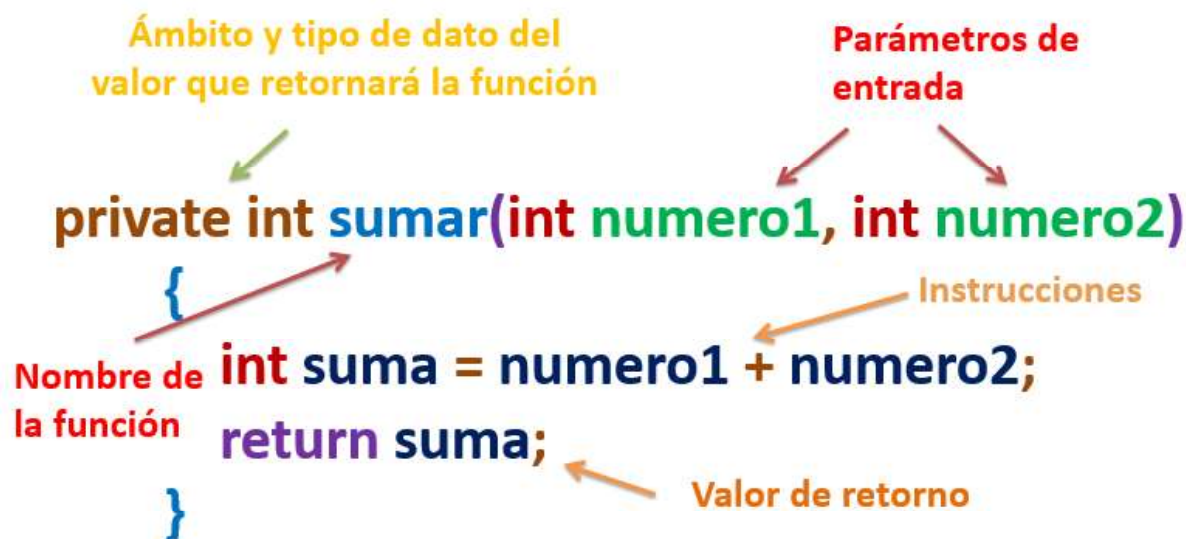


Lenguajes de Programación 1

Unidad 1

1.1 Funciones con retorno

- Retornan un determinado tipo de dato en el nombre de la función.
- Esta funcionalidad se realiza a través de la sentencia **return**.
- Es el tipo de función más comúnmente empleado.



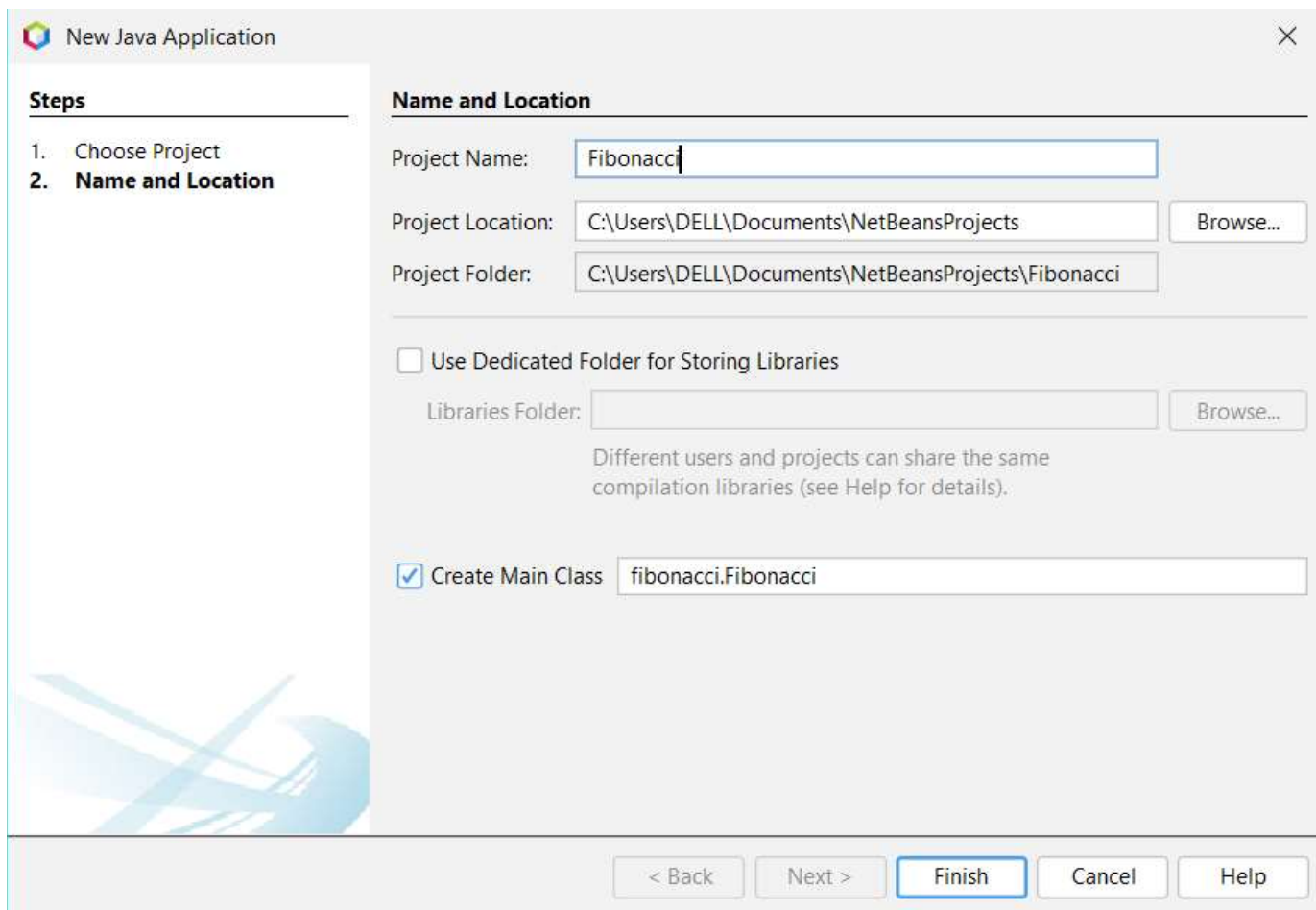
```
private int sumar(int numero1, int numero2)
{
    int suma = numero1 + numero2;
    return suma;
}
```

Diagram illustrating the components of a function definition:

- Ámbito y tipo de dato del valor que retornará la función**: Points to `private int`.
- Parámetros de entrada**: Points to `int numero1, int numero2`.
- Nombre de la función**: Points to `sumar`.
- Instrucciones**: Points to `int suma = numero1 + numero2;`.
- Valor de retorno**: Points to `return suma;`.



Serie de Fibonacci



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location: Browse...

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

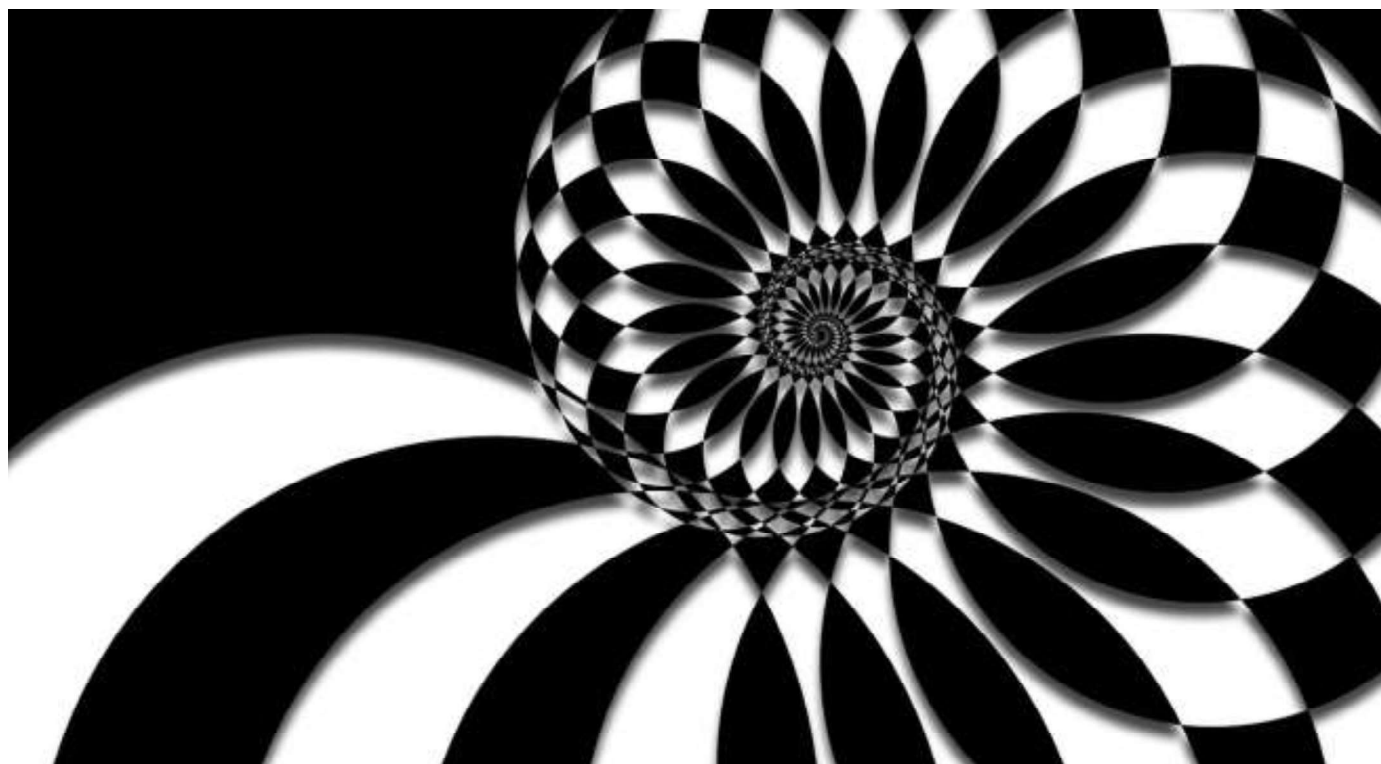
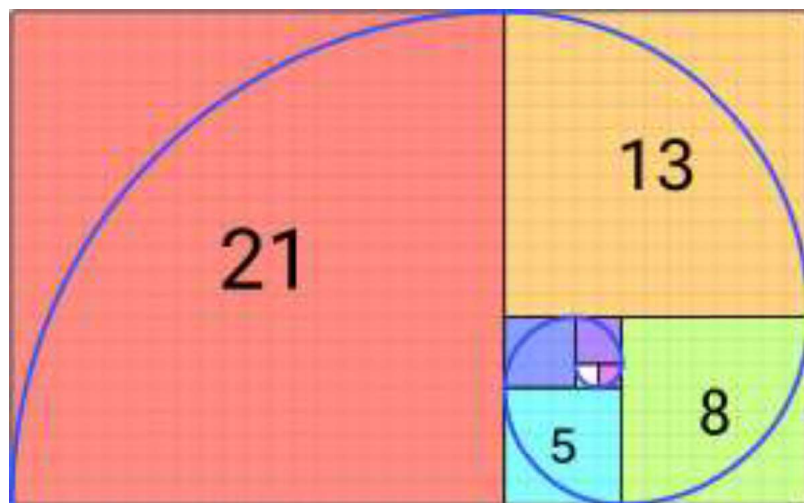
< Back Next > **Finish** Cancel Help



0 1 1 2 3 5 8 13 21



Serie de Fibonacci



Serie de Fibonacci

```
1 package fibonacci;
2 import java.util.Scanner;
3
4 // Comentario aquí
5 public class Fibonacci {
6
7     // Comentario aquí
8     public static void main(String[] args) {
9         Scanner lectura = new Scanner (System.in);
10        System.out.println("Ingrese cantidad de dígitos: ");
11        int num = Integer.parseInt(lectura.next());
12
13        if (num <= 2) {
14            System.out.print("Número no válido");
15            return;
16        }
17
18        int n1 = 0;
19        int n2 = 1;
20        int aux;
21        System.out.print(n1 + " " + n2 + " ");
22
23        for (int i=0; i<num-2; i++) {
24            aux = n1;
25            n1 = n2;
26            n2 = generarNumero(aux, n2);
27            System.out.print(n2 + " ");
28        }
29
30        System.out.println();
31    }
```

```
// Comentario aquí
public static int generarNumero(int n1, int n2) {
    return n1+n2;
}
```

Ahora implementa la serie usando un método sin retorno!



1.1 Recursividad

- Estrategia a través de la cual una función se invoca a sí misma.
- Concepto teórico de fácil comprensión pero comúnmente de difícil implementación práctica.
- Por lo general, todo problema iterativo puede ser implementado a la vez a través de recursividad.





```
1 package fibonacci;  
2 import java.util.Scanner;  
3  
4 // Comentario aquí  
5 public class Fibonacci {  
6  
7     // Comentario aquí  
8     public static void main(String[] args) {  
9         Scanner lectura = new Scanner (System.in);  
10        System.out.println("Ingrese cantidad de dígitos: ");  
11        int num = Integer.parseInt(lectura.next());  
12  
13        if (num <= 2) {  
14            System.out.println("Número no válido");  
15            return;  
16        }  
17  
18        for (int i=0; i<num; i++) {  
19            System.out.print(generarNumero(i) + " ");  
20        }  
21  
22        System.out.println();  
23    }  
24
```





```
25 // Comentario aquí
26 public static int generarNumero(int numero) {
27     if (numero == 0)
28         return 0;
29     else if (numero ==1)
30         return 1;
31     else
32         return generarNumero(numero-1)+generarNumero(numero-2);
33 }
34 }
```



Tarea 2: Triángulo de Pascal con Funciones

- Implementar una solución iterativa y recursiva del triángulo de Pascal.
- Puedes usar ChatGPT, pero por favor procura entender el código fuente y comentarlo adecuadamente...
- El parámetro de entrada será el número de filas del triángulo.
- Crea tantas funciones como consideres necesario.

