1.- Complete el código de la clase Stack en el paquete practica3, de manera que implemente una pila de String:

```
public class Stack {
    private java.util.ArrayList items;
    public Stack() { . . }
    public void push(Object item) { . . }
    public Object pop() { . . }
    public boolean isEmpty() { . . }

}
```

a) Implemente un método main() para probar la pila. Agregue Strings a la pila y recórrala para imprimir sus valores. ¿Cuántas veces puede recorrerla?
Si usamos el pop, una sola porque vamos borrando los elementos

b) Agregue una clase anidada llamada StackIterator que provea un objeto de tipo Iterator para recorrer la pila.
c) Agregue en la clase Stack un método para que retorne una instancia de StackIterator. ¿Cuántas veces puede recorrer la pila ahora?
Todas las que quiera

d) ¿Es posible crear objetos StackIterator desde una clase diferente a la clase Stack con el operador new?, ¿cómo lo hace?
Ni. con la clase anidada privada no, pero si la hago pública puedo hacerlo con un stack.new StackIterator();

e) ¿Cómo haría para evitar crear instancias de una clase anidada desde una clase que no sea la que la definió?
haciéndola privada je
2.- Analice el código que figura debajo.

```
class InnerStatic {
    static double PI = 3.1416;
    static class Circulo {
        static double radio = 2;
        static double getArea() {
            double a= PI* Math.pow(radio,2);
            System.out.println("El area es: "+a);
            return a;
        }
        static double getLongitudCircunsferencia(){
            double l= 2*PI*radio;
            System.out.println("La longitud es: "+l);
            return l;
        }
    }
    . . .
}
```

a) Modifique el código de la clase interna estática para que el valor inicial del radio sea ingresado por el usuario en el momento de la ejecución.
b) Defina una clase llamada InnerTest en el paquete practica3 con un método main() que imprime en la pantalla el área y la longitud de la circunferencia. Ejecútela varias veces ingresando distintos radios.
c) Remplazar PI* Math.pow(radio,2) por PI* pow(radio,2), siendo pow() el método de la clase java.lang.Math.

4.- Indicar si son verdaderas o falsas las siguientes afirmaciones sobre las clases anónimas y en cada caso justifique su respuesta:
❏ Se pueden instanciar más allá del punto en donde fueron declaradas.
falso. Como no tienen nombre, no se pueden usar fuera de donde están declaradas
❏ Unos de los usos más comunes de este tipo de clases es la creación de objetos función y procesos on the fly.
verdadero. sirve porque las usas una sola vez y fue. cuando sólo se necesita una instancia puntual, sin tener que crear un archivo de clase separado.
❏ Se puede utilizar el instanceof siempre y cuando la interfaz de la que deriva la clase anónima sea de tipo marker.
instanceof funciona para cualquier tipo de referencia conocido en tiempo de compilación (interfaz o clase) sin importar si la interfaz es de tipo marker (como Serializable o Cloneable) o no. El hecho de que la clase concreta sea anónima no afecta a la verificación de instanceof.
❏ No se puede implementar múltiples interfaces o extender clases e implementar interfaces al mismo tiempo.
verdadero
Si haces new SuperClase(){…} no puedes además poner implements X, Y.

Si haces new Interface(){…} puedes implementar solo esa interfaz (no varias a la vez en la misma anónima).

5.- Modifique el código de la clase Stack, para que ahora la clase anidada StackIterator, se convierta en una clase anónima.
a) ¿En que situación es conveniente definir a una clase cómo anónima?
Cuando la voy a usar una sola vez y no necesito más instancias.
b) Si tendría que inicializar valores de la clase anónima (cuando se crea una instancia de la misma), ¿cómo lo haría?
la seteas de una cuando declaras las variables o le pones un bloque de inicialización

6.- Defina una clase llamada Estudiante que contenga las siguientes variables de instancia: apellido, nombre, edad, legajo y materiasAprobadas. Se necesita poder ordenar un arreglo con estos objetos por los siguientes criterios:
● Por cantidad de materias aprobadas en forma ascendente.
● Por edad en forma descendente.
● Por legajo en forma ascendente.
● Por nombre y apellido en forma descendente.
Implemente un método main() que imprima los resultado de las distintas ordenaciones utilizando clases anónimas y el método Arrays.sort().
7. Uso de Módulos

Retomando la clase Logger que implementaron en el ejercicio 4 de la TP 1, realice ahora un proyecto modular:

● Defina un módulo llamado loggingutils que contenga la clase Logger.

● En cada uno de sus métodos (logInfo(String mensaje), logWarning(String mensaje), logError(String mensaje)), en lugar de imprimir directamente en consola, utilice la clase java.util.logging.Logger para registrar los mensajes.

● ¿Para qué sirve el archivo module-info.java? En el módulo loggingutils, ¿qué declaración debe incluirse en module-info.java para exponer su paquete al resto de los módulos? Para usar java.util.logging.Logger desde loggingutils, ¿es necesario declarar alguna dependencia en module-info.java? ¿Por qué?

En el se declara el nombre del módulo, qué paquetes se exponen a otros módulos y de qué otros módulos depende.

```
module loggingutils {
    exports loggingutils;
}
```

para exportarlo y no es necesario incluir nada porque está en java base que se importa por defecto → si lo tuve que poner xd

● Luego, cree un segundo módulo llamado test que contenga una clase con el método main(). Desde el main(), utilice el Logger del módulo loggingutils para mostrar los tres tipos de mensajes en la salida.