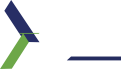
Actividad de Aprendizaje Semana 01



**Indicaciones Generales:**

1. El plagio se sanciona con la suspensión o expulsión del estudiante de la Universidad. Reglamento General de Estudios.
2. Sea cuidadoso con su redacción, la cual formará parte de su calificación (aspectos léxicos, sintácticos, semánticos).
3. Lea bien la pregunta o enunciado antes de proceder a su desarrollo, administre su tiempo eficazmente.

# Actividad 1.

* 1. Crear una clase llamada Círculo la cual contiene:
     1. Dos variables de instancia privada: radio (de tipo double) y color (de tipo String), con valor por defecto de 12.5 y "azul", respectivamente.
     2. Dos constructores sobrecargados (overloaded): un constructor predeterminado o por default sin argumentos y otro constructor que tiene un argumento de input double para el valor del radio.
     3. Dos métodos públicos: getRadio() y getArea(), que devuelven el radio y el área de esta instancia, respectivamente. Para calcular el área de un círculo, deberás seguir la siguiente fórmula: area = radius\*radius\*Math.PI
  2. Compilar y ejecutar la clase "Circulo.java". ¿Puedes ejecutar la clase “Circulo.java”? ¿Por

qué?

* No se puede ejecutar “Circulo.java”, porqué no tenemos un main para poder ejecutar la instancia de la clase que contiene la lógica de los radios del circulo.
  1. Crear una clase CirculoTest, esta clase contendrá el método *main*() para poder visualizar los siguientes datos:

Utilizando un constructor por defecto para imprimir:

* + 1. El radio de un círculo.
    2. El área de un círculo.

Utilizando un constructor sobrecargado, donde radio es igual a 6.55, para imprimir:

1. El radio de un círculo.
2. El área de un círculo.

¿Qué pasos en común debe realizar para ejecutar cada uno de los ejercicios? Describir conceptos del paradigma orientada a objetos.

**Clases y objetos**

En este caso, Circulo es una clase que define propiedades (radio y color) y métodos (getRadio() y getArea()). Entonces cuando escribimos Circulo c1 = new Circulo();, estamos creando un objeto basado en la clase Circulo.

**Encapsulamiento**

Los atributos de la clase (radio y color) haciéndolos private y se proporciona acceso a los datos mediante métodos públicos.

**Sobrecarga y constructores**

Se usa un constructor sobrecargado para permitir la creación de círculos con radios personalizados.

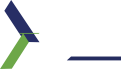
* 1. Ejecutar la clase “CirculoTest.java” y responde las siguientes preguntas:
     1. ¿Qué es el constructor y para qué sirve?
* Un constructor es un método especial de una clase que se ejecuta automáticamente cuando se crea un objeto de esa clase. Su propósito es inicializar los atributos del objeto con valores predeterminados o específicos proporcionados por el usuario.
  + 1. ¿Con qué método recupero el radio del círculo?
* El radio del círculo se obtiene mediante el método getRadio(), que devuelve el valor almacenado en la variable radio
  + 1. ¿Por qué los atributos son privados y los métodos son públicos?
* Los privados son para aplicar el principio de encapsulamiento impidiendo que sean modificados directamente desde fuera de la clase.
* Los públicos para que otros objetos puedan interactuar con los datos, sin modificar directamente las variables internas.
  + 1. ¿Qué pasa si accedes directamente a los atributos del círculo desde el método

*main*()?

* Si intentas acceder directamente a los atributos radio o color desde main(), obtendrás un error de compilación porque son private.
  + 1. ¿Qué pasa si modificas directamente el atributo área del círculo desde el método

*main*()?

* No existe un atributo area en la clase Circulo, solo el método getArea(), que calcula el área cuando se llama, daría error al compilar.



* + 1. En la clase “CirculoTest.java”, ejecuta la siguiente instrucción: System.out.println(c1.radio), donde c1 es una instancia del objeto Círculo. ¿Qué ocurrió? Explique el mensaje de error.
* **java: radio has private access in Circulo**

Indica que radio es private, por lo que no es accesible desde fuera de la clase Circulo.

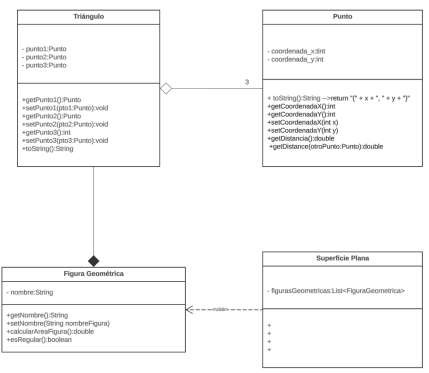
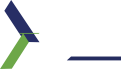
* + 1. En la clase “CirculoTest.java”, ejecuta la siguiente instrucción: c1.radio=42.0, donde c1 es una instancia del objeto Círculo. ¿Qué ocurrió? Explique el mensaje de error.
* En Circulo.java, el atributo radio fue declarado como private, lo que significa que solo puede ser accedido o modificado dentro de la propia clase Circulo.
  + 1. ¿Con qué método modificas el valor de un atributo de la clase “Circulo.java”?
* Utilizando un método setter, por ejemplo un método llamado setRadio.
  + 1. Crear un objeto de tipo Círculo y modifica los valores por defecto.
    2. ¿Para qué sirve la palabra reservada ***“this”***? Modifique el constructor con argumentos (radio y color) y utiliza **“this”** y métodos setters en la clase “Circulo.java”.
* La palabra clave this en Java se usa para referirse a la instancia actual de la clase.
  + 1. ¿Para qué sirve el método toString()?
* El método toString() en Java se usa para representar un objeto como una cadena de texto.
  + 1. Incluir el método toString() en la clase “Circulo.java”.
  1. Trabaja en el diagrama de clase para la clase Circulo utilizando la herramienta online Visio o PlantUML (<https://plantuml.com/es/class-diagram>).

Diagrama

El contenido generado por IA puede ser incorrecto.

# Actividad 2.

1. Se tiene las siguientes clases:
   * Crear la clase Punto.
   * Crear la clase Triángulo.
   * Crear la clase Figura Geométrica. Completar con 2 métodos abstractos: calcular el área y determinar si es una figura geométrica regular o no. Adicionalmente, incorporar métodos de accesos y manipulación.
   * Crear la clase Superficie Plana. Esta clase tendrá un método que devolverá las áreas de cada figura geométrica. Completarlo.
   * Crear una clase genérica para invocar a todas las clases mencionadas. Esta clase genérica debe contener el método main().



1. Para la clase Punto considerar:
2. El método toString()devolverá lo siguiente: "El punto tiene las siguientes coordenadas: " + x + "," + y .
3. Un constructor sin parámetros y otro con parámetros (coordenaX, coordenaY).
4. Incluir todos sus métodos de acceso y manipulación.
5. La clase Punto tiene un método calcular distancia que se sobreescriben, el que no cuenta con parámetros retornará el resultado del otro método con parámetro.
6. Para la clase Triángulo considerar:
7. El método toString() devolverá lo siguiente: "Triángulo: " + getNombre() + " tiene 3 Puntos: "+ getPunto1().toString() + “,” + getPunto2().toString()+ “,” + getPunto3().toString();
8. Un constructor sin parámetros y otro con parámetros pero éste debe invocar al constructor de la superclase.
9. Incluir todos sus métodos de acceso y manipulación.
10. Sobreescribir los métodos hereados de la clase Figura Geométrica.
11. Identificar las clases que heredan, ¿qué atributos y métodos heredan?

* Triangulo hereda de FiguraGeometrica los métodos abstractos y los implementa con su propia lógica.

1. Identificar las clases que conforman una composición.

|  |  |  |
| --- | --- | --- |
| **Clase Contenedora** | **Clase Contenida** | **Tipo de relación** |
| Triangulo | Punto | Composición fuerte |
| SuperficiePlana | FiguraGeometrica | Agregación (composición débil) |

1. ¿Qué es una superclase y una subclase?

* Una superclase (o clase padre) es una clase que proporciona atributos y métodos comunes que pueden ser heredados por otras clases.
* Una subclase (o clase hija) es una clase que hereda atributos y métodos de una superclase y puede agregar nuevos comportamientos o sobreescribir los existentes.

1. ¿Por qué usamos *abstract*? ¿Se puede dejar de heredar un método de una clase abstracta?

* La palabra clave abstract se usa en Java para definir clases y métodos abstractos. Se utiliza cuando queremos que una clase sirva como base para otras clases, pero sin permitir su instanciación directa.

No, una subclase debe implementar todos los métodos abstractos de su superclase, a menos que ella misma también sea abstracta.

1. ¿Qué anotación utilizo para sobreescribir métodos?

* La anotación @Override se usa en Java para indicar que un método en una subclase sobrescribe (reemplaza) un método de la superclase.

1. Los atributos de la clase Figura Geométrica conviértalas en protected. ¿En qué condición convierte a los atributos? ¿Es posible acceder a los atributos protegidos sin utilizar una invocación a super() o sin método get?.

* Sigue siendo encapsulado, pero ahora las subclases pueden acceder a los atributos directamente sin métodos get() o cuando se permite acceso dentro del mismo paquete.
* Es posible permitir el acceso directo en subclases y dentro del mismo paquete, sin necesidad de super() o get().

1. ¿Cómo aplicarías polimorfismo?

* Crear la superclase FiguraGeometrica con un método abstracto
* Crear una subclase Triangulo que sobrescriba calcularArea()
* Aplicar polimorfismo en SuperficiePlana