1.

public class Circulo {

private double radio;

private String color;

public Circulo(){

radio = 5.0;

color = "azul";

}

public Circulo(double radio){

this.radio = radio;

color = "azul";

}

public double getRadio() {

return radio;

}

public double getArea(){

return radio \* radio \* Math.PI;

}

}

2.Para ejecutar los ejercicios que mencionamos anteriormente en Android Studio, se siguen los siguientes pasos comunes:

Crear un Proyecto o Módulo:

Abre Android Studio.

Crea un nuevo proyecto o utiliza un proyecto existente (dependiendo de si estás trabajando en un nuevo proyecto o en uno existente).

Crear una Clase:

En el directorio adecuado del proyecto, crea un nuevo archivo Java para la clase que deseas crear. En este caso, hemos creado las clases Circulo y CirculoTest.

Definir la Clase:

Define la estructura de la clase, incluyendo las variables de instancia, constructores y métodos necesarios.

Escribir el Código:

Escribe el código dentro de la clase para definir las variables de instancia, los constructores y los métodos, según sea necesario.

Crear un Programa Principal:

Si estás creando una clase de prueba como CirculoTest, agrega un método main en esta clase. Este método será el punto de entrada para tu programa y te permitirá ejecutar y probar la funcionalidad de la clase Circulo.

Crear Objetos y Realizar Acciones:

En el método main, crea objetos de la clase que estás probando y realiza las acciones que desees probar. En el caso de CirculoTest, creamos objetos Circulo, establecemos valores y realizamos acciones como obtener el radio y el área.

Ejecutar el Programa:

Ejecuta el programa haciendo clic en el botón "Run" (Ejecutar) en Android Studio. Esto compilará y ejecutará el código, mostrando los resultados en la consola.

Observar Resultados:

Observa los resultados en la consola de Android Studio para verificar que el programa está funcionando como se esperaba.

Estos son los pasos comunes para ejecutar programas en Android Studio. Los detalles específicos pueden variar según el tipo de proyecto y el código que estés escribiendo, pero estos pasos generales te ayudarán a ejecutar tus programas y probar su funcionalidad.

3.

public class CirculoTest {

public static void main(String[] args){

Circulo circulo1 = new Circulo();

System.out.println("Radio del cìrculo (Constructor por defecto):" + circulo1.getRadio());

System.out.println("Àrea del cìrculo (Constructor por defecto):" + circulo1.getArea());

Circulo circulo2 = new Circulo(6);

System.out.println("Radio del cìrculo (Constructor sobrecargado):" + circulo2.getRadio());

System.out.println("Àrea del cìrculo (Constructor sobrecargado):" + circulo2.getArea());

}

}

4.

a. ¿Qué es el constructor y para qué sirve?

El constructor es un método especial en una clase que se llama automáticamente cuando se crea un objeto de esa clase. Sirve para inicializar los atributos de la clase y realizar cualquier otra configuración necesaria para que el objeto esté listo para su uso.

b. ¿Con qué método recupero el radio del círculo?

Puedes recuperar el radio del círculo utilizando el método público getRadio() definido en la clase Circulo.

c. ¿Por qué los atributos son privados y los métodos son públicos?

Los atributos son privados para encapsular los datos y evitar que se acceda directamente a ellos desde fuera de la clase. Los métodos son públicos para proporcionar una interfaz controlada para acceder y modificar estos atributos. Esto sigue el principio de encapsulación en la programación orientada a objetos.

d. ¿Qué pasa si accedes directamente a los atributos del círculo?

Si intentas acceder directamente a los atributos privados del círculo desde fuera de la clase, obtendrás un error de compilación, ya que los atributos son inaccesibles desde fuera de la clase.

e. ¿Qué pasa si modificas directamente el atributo area del círculo?

En la clase Circulo, no existe un atributo llamado area. El área se calcula en función del radio y es un atributo calculado a través del método getArea(). Si intentas modificarlo directamente, no tendría sentido y no estarías siguiendo las buenas prácticas de diseño de clases.

f. En la clase "CirculoTest.java", ejecuta la siguiente instrucción: System.out.println(c1.radius), donde c1 es una instancia del objeto Circulo. ¿Qué ocurrió? Explique el mensaje de error.

Obtendrás un error de compilación porque radius es un atributo privado de la clase Circulo, por lo que no puedes acceder directamente a él desde fuera de la clase.

g. En la clase "CirculoTest.java", ejecuta la siguiente instrucción: c1.radius=12.0, donde c1 es una instancia del objeto Circulo. ¿Qué ocurrió? Explique el mensaje de error.

Obtendrás un error de compilación por la misma razón que en la pregunta anterior. radius es un atributo privado y no se puede acceder ni modificar directamente desde fuera de la clase.

h. ¿Con qué método modificas el valor de un atributo de la clase "Circulo.java"?

Para modificar el valor de un atributo de la clase Circulo, deberías utilizar un método setter. Por ejemplo, podrías tener un método setRadio(double nuevoRadio) para modificar el valor del radio.

i. Crear un objeto de tipo Circulo y modifica los valores por defecto. Utiliza el constructor por defecto.

Circulo miCirculo = new Circulo(); // Crea un círculo con constructor por defecto

miCirculo.setRadio(8.0); // Modifica el radio

j. ¿Para qué sirve la palabra reservada “this”? Modifique todos los constructores y métodos setters en la clase “Circulo.java”.

La palabra reservada this se utiliza en Java para hacer referencia a la instancia actual de la clase. Puede ser útil para evitar la ambigüedad cuando hay una variable local con el mismo nombre que un atributo de la clase. Aquí hay un ejemplo de cómo se podría usar en la clase Circulo:

public class Circulo {

private double radio;

private String color;

public Circulo() {

this.radio = 5.0; // Utilizando "this" para referirse al atributo de la clase

this.color = "azul";

}

public void setRadio(double radio) {

this.radio = radio; // Utilizando "this" para referirse al atributo de la clase

}

}

k. ¿Para qué sirve el método toString()?

El método toString() se utiliza para devolver una representación en forma de cadena (String) de un objeto. Es comúnmente utilizado para proporcionar una representación legible de un objeto que puede ser útil para depuración y para imprimir información sobre el objeto.

l. Incluir el método toString() en la clase "Circulo.java".

@Override

public String toString() {

return "Círculo [Radio=" + radio + ", Color=" + color + "]";

}

Este método toString() devuelve una cadena que representa las propiedades del círculo (radio y color) de una manera legible. Puedes llamar a este método en una instancia de la clase para obtener una representación de cadena del objeto.