# Kadir's Code:

Vorteile:

**Benutzerfreundlichkeit**: Es interagiert direkt mit dem Benutzer über die Konsole, was die Bedienung intuitiver macht.

**Einfachheit**: Es verwendet einfache Kontrollstrukturen wie if-else und einfache Berechnungen.

**Flexibilität**: Es ermöglicht die Berechnung verschiedener Formen ohne die Notwendigkeit von Kommandozeilenargumenten.

**Wiederverwendbarkeit**: Die „user\_input“-Methode ermöglicht die Wiederverwendung des Codes für die Eingabe von Messungen.

Nachteile:

**Mangel an Modularität**: Der Code ist eher monolithisch und könnte in kleinere Funktionen aufgeteilt werden, um die Lesbarkeit und Wartbarkeit zu verbessern.

**Fehleranfälligkeit**: Benutzereingaben werden nicht validiert, was zu potenziellen Ausnahmen führen kann.

**Keine Anpassungsfähigkeit**: Es erwartet eine bestimmte Eingabeaufforderung und erlaubt keine flexiblen Eingaben.

**Begrenzte Erweiterbarkeit**: Das Hinzufügen neuer Formen erfordert Änderungen im Code und könnte unübersichtlich werden.

Musterlösung:

Vorteile:

**Modulare Struktur:** Der Code ist in separate Methoden für jede Form aufgeteilt, was die Wartbarkeit und Lesbarkeit verbessert.

**Validierung von Eingaben**: Es überprüft, ob genügend Argumente vorhanden sind, und gibt eine Hilfemeldung aus, wenn nicht.

**Verwendung von Methoden**: Die Verwendung von Methoden erleichtert die Wiederverwendung und die Organisation des Codes.

Nachteile:

**Mangel an Benutzerinteraktion**: Es hat keine direkte Benutzerinteraktion und erfordert die Eingabe von Befehlen über die Kommandozeile.

**Begrenzte Flexibilität**: Es kann schwieriger sein, neue Formen hinzuzufügen oder den Code für andere Zwecke zu verwenden, da er stark auf Kommandozeilenargumente ausgerichtet ist.

# Für Clean-Code-Prinzipien:

**Sprechende Namen**: Beide Codes verwenden sprechende Namen für Variablen, Methoden und Funktionen, die ihre Funktion klar beschreiben.

**Einzelverantwortlichkeit (Single Responsibility Principle):** Beide Codes haben separate Methoden für die Berechnung jeder Form, was ihre Verantwortlichkeiten trennt und die Wartbarkeit verbessert.

**Formatierung**: Beide Codes folgen einer konsistenten Formatierung und Einrückung, was die Lesbarkeit erhöht.

**Magische Zahlen**: Beide Codes enthalten sogenannte "magische Zahlen", d.h. numerische Werte, die direkt im Code verwendet werden, ohne dass ihre Bedeutung klar ist. Diese sollten durch aussagekräftige Konstanten oder Variablen ersetzt werden.

**Benutzerinteraktion und Input-Validierung**: Der Code von Kadir interagiert direkt mit dem Benutzer über die Konsole und verlässt sich darauf, dass die Eingaben korrekt sind, ohne sie zu validieren. Es wäre besser, Benutzereingaben zu überprüfen und Fehler abzufangen, um eine robustere Anwendung zu gewährleisten.

**Kapselung und Abstraktion**: Beide Codes könnten besser von der Benutzeroberfläche und den Eingabeaufforderungen entkoppelt werden, um eine bessere Abstraktion zu erreichen. Dies würde die Wiederverwendbarkeit des Codes erhöhen.

**Testbarkeit**: Beide Codes sind nicht besonders testbar, da sie stark von Benutzereingaben und externen Argumenten abhängen. Eine bessere Modularisierung und Trennung von Eingabe/Ausgabe würde die Testbarkeit verbessern.