### Aufgabe 1: Eigener Name ausgeben

**Ziel:** Eine einfache `void`-Methode ohne Parameter definieren und aufrufen.

- Schreibt eine Methode namens `PrintMyName`.
- Die Methode soll keinen Rückgabewert haben (`void`) und keine Parameter annehmen.
- Innerhalb der Methode, gebt euren vollen Namen auf der Konsole aus.
- Ruft die Methode `PrintMyName` in eurem Hauptprogramm genau einmal auf.
- Der name selbst ist im Code statisch definiert

---

### Aufgabe 2: Persönliche Begrüßung erstellen

**Ziel:** Eine Methode mit einem `string`-Parameter und einem `string`-Rückgabewert erstellen.

- Schreibt eine Methode `GetGreeting`, die einen `string`-Parameter namens `name` entgegennimmt.
- Die Methode soll einen Begrüßungstext nach dem Muster `"Hallo, [name]!"` zusammenbauen und als `string` zurückgeben.
- Ruft die Methode mit eurem Namen auf.
- Speichert den zurückgegebenen `string` in einer Variable und gebt diese Variable auf der Konsole aus.

---

### Aufgabe 3: Gerade oder Ungerade?

**Ziel:** Eine Methode mit einem `int`-Parameter und einem `bool`-Rückgabewert implementieren.

- Schreibt eine Methode `IsEven`, die einen `int`-Parameter `number` entgegennimmt.
- Die Methode soll überprüfen, ob die übergebene Zahl gerade ist.
- Wenn die Zahl gerade ist, soll `true` zurückgegeben werden, andernfalls `false`.
- **Tipp:** Verwendet den Modulo-Operator (`%`). Eine Zahl ist gerade, wenn `number % 2` das Ergebnis `0` hat.
- Testet die Methode, indem ihr sie mit einer geraden und einer ungeraden Zahl aufruft und die Ergebnisse ausgebt.

---

### Aufgabe 4: Werte tauschen mit `ref`

**Ziel:** Die Funktionsweise von `ref` verstehen, um Variablen außerhalb einer Methode zu verändern.

- Schreibt eine Methode `Swap`, die zwei `string`-Parameter `a` und `b` als Referenz (`ref`) entgegennimmt.
- Innerhalb der Methode sollen die Werte der beiden Variablen vertauscht werden.
- **Tipp:** Ihr benötigt eine dritte, temporäre Variable, um einen der Werte zwischenzuspeichern.
- Testet die Methode, indem ihr zwei `string`-Variablen deklariert, ihre Werte vor dem Aufruf ausgebt, die `Swap`-Methode aufruft und die Werte danach erneut ausgibt.

---

### Aufgabe 5: Mehrere Werte zurückgeben mit `out`

**Ziel:** `out`-Parameter verwenden, um mehrere Berechnungsergebnisse aus einer einzigen Methode zu erhalten.

- Schreibt eine Methode `GetCircleDetails`, die einen `double`-Parameter `radius` entgegennimmt.
- Die Methode soll zusätzlich zwei `out double`-Parameter haben: `circumference` und `area`.
- Innerhalb der Methode berechnet ihr den Umfang und die Fläche des Kreises und weist die Ergebnisse den `out`-Parametern zu.
  - Formel Umfang: `2 * Math.PI * radius`
  - Formel Fläche: `Math.PI * radius * radius`
- Ruft die Methode auf. Deklariert die Variablen für die `out`-Parameter direkt beim Aufruf (z.B. `out double c`).
- Gebt die beiden zurückgegebenen Werte (Umfang und Fläche) auf der Konsole aus.

Note:

- Aufgabe A: `ref` praktisch anwenden und verinnerlichen.
- Aufgabe B: Typischer Anwendungsfall für `out`-Parameter (mehrere Ergebnisse).

---

### Aufgabe 6: Werte verschiedener Typen ausgeben

**Ziel:** Das Konzept der Methodenüberladung durch die Implementierung von Methoden mit gleichem Namen, aber unterschiedlichen Parametern, verstehen.

- Schreibt eine überladene Methode namens `DisplayValue`.
- **Version 1:** Erstellt eine Methode, die einen `int` entgegennimmt und ihn formatiert ausgibt, z.B. `"Ganze Zahl: [wert]"`.
- **Version 2:** Erstellt eine Methode, die einen `string` entgegennimmt und ihn als `"Text: [wert]"` ausgibt.
- **Version 3:** Erstellt eine Methode, die einen `bool` entgegennimmt und ihn als `"Wahrheitswert: [wert]"` ausgibt.
- Ruft jede dieser drei Methoden einmal auf, um zu zeigen, dass je nach übergebenem Datentyp die korrekte Version ausgeführt wird.

---

### Aufgabe 7: Summe mit Rekursion berechnen

**Ziel:** Eine rekursive Methode schreiben, die einen Basisfall und einen rekursiven Schritt korrekt implementiert.

- Schreibt eine rekursive Methode `SumUpTo(int n)`.
- Die Methode soll die Summe aller ganzen Zahlen von 1 bis `n` berechnen (z.B. `SumUpTo(3)` ist `3 + 2 + 1 = 6`).
- **Basisfall:** Legt fest, was passieren soll, wenn `n` den Wert 1 erreicht. In diesem Fall ist das Ergebnis einfach 1 und die Rekursion endet.
- **Rekursiver Schritt:** Für alle `n > 1` soll die Methode `n` zum Ergebnis von `SumUpTo(n - 1)` addieren.
- Testet die Methode mit einer kleinen Zahl wie 4 und gebt das Ergebnis aus.