# DECIMAL FORMAT

Decimal format is a class in 'java.text' package for formatting decimal numbers.
It allows us to format numbers into human readable string or parse string back into numbers.

- Key Points

Pattern-based Formatting :- Define patterns to specify how numbers should be formatted.

Locale-Sensitive - formats numbers according to locale specific conventions.

Customizable - You can customize the pattern for decimal places, grouping separators, prefixes, suffixes, etc.

- Common Patterns

0 - Digit, shows zero if no digit.

# - Digit, does not show if zero.

. - Decimal separator

, - Grouping separator

% - Multiplying by 100 and shows as percentage

¤ - Currency sign, replaced by currency symbol

E - seprates mantissa and exponent in scientific notation.

- Example Patterns

"###,###.##" - formats numbers with commas as thousands seprators and up to two decimal places.

"000.00" - Always shows atleast one digit before and two digit after decimal point

"#.###E0" - Scientific notation with upto three decimal places

- Example

```
import java.text.DecimalFormat;
public static DecimalFormatExample 1
    public static void main(String[] args) {
        double number = 12345.6789;
        //Default pattern
        DecimalFormat df = new DecimalFormat();
        System.out.println(df.format(number)); //12,345.679
        //Custom pattern
        df.applyPattern("###,###.##");
        System.out.println(df.format(number)); //12,345.68
        //Leading zeros
        df.applyPattern("000000.00");
        System.out.println(df.format(number)); #012345.679
        //Scientific notation
        df.applyPattern("#.###E0");
        System.out.println(df.format(number)); #1.235E4
        //Percentage
        df.applyPattern("#%");
        System.out.println(df.format(0.85)); //85%
        //Currency
        df.applyPattern("¤#,##0.00");
        System.out.println(df.format(number)); //$12,345.68
    }
}
```

- Parsing String

Decimal format can also parse strings back into numbers.

try {

```
Number num = df.parse("12,345.68");
System.out.println(num); // 12345.68
} catch (ParseException e) {
e.printStackTrace();
}
```

- Custom Symbols

You can customize symbols like decimal separators, grouping seprators

```
DecimalFormatSymbols symbols = new DeimalFormatSymbols();
symbols.setDecimalSeparators('.');
symbols.setGroupingSeparators(',');
df.setDecimalFormatSymbols(symbols);
```