**1. Implicit Casting**

**What it is:** Automatic type conversion by the compiler when there's no risk of losing data.

**When to use:** Going from a **smaller to a larger** data type.

**Example:**

int x = 10;

double y = x; // Implicit: int to double

**Pros:** Safe and clean. **Cons:** Only works when the conversion is risk-free.

**2. Explicit Casting**

**What it is:** Manual type conversion using a cast operator ().

**When to use:** Going from a **larger to a smaller** type (which may lose data).

**Example:**

double x = 12.7;

int y = (int)x; // Explicit: double to int

**Pros:** You’re in control. **Cons:** You may lose precision or get runtime errors.

**3. Convert Class**

**What it is:** A method-based way to convert types. Safer than explicit casting in some cases.

**When to use:** When you want to convert **any type** (e.g., string, bool, int) with built-in error handling.

**Example:**

string str = "100";

int x = Convert.ToInt32(str); // String to int

**Pros:** More robust and flexible. **Cons:** Can throw exceptions if data is invalid.

**4. Parse Method**

**What it is:** Used for **strings only**, to convert text to a value type.

**When to use:** When you’re sure the string is correctly formatted.

**Example:**

csharp

string str = "250";

int x = int.Parse(str); // String to int

**Pros:** Direct and efficient. **Cons:** Throws exceptions if the string is badly formatted.