# Progetto Java

Nome Progetto: Text2Morse

Descrizione:

Programma java di 3 classi + Main.

Classe Morse Entry: Con uno Scanner memorizza il testo scritto

Classe Morse Converter: Prende il testo scritto e lo converte in codice Morse

Classe FileManager: Prende il testo scritto a mano e la sua conversione e li salva in un file TXT.

Tempistica:

UML: 3h

Creazione classe 1: 2h

Creazione classe 2: 2h

Creazione classe 3:2h

ArrayList: 2h

Implementazione Scanner: 2h

Implementazione Scrittura file: 2h

Ultime messe a posto+eccezioni: 2h

JavaDoc: 1h

Totale = 18h~

Lo StringBuilder è una classe di Java (contenuta nel package java.lang) che permette di **costruire e modificare stringhe in modo efficiente**. È particolarmente utile quando devi **concatenare molte stringhe** in un ciclo o fare molte modifiche, perché è **più veloce di usare direttamente le stringhe (String)**.

### Perché usare StringBuilder invece di String?

In Java, gli oggetti String sono **immutabili**, cioè **non possono essere modificati** una volta creati. Ogni volta che "modifichi" una String, in realtà ne stai creando una nuova. Questo può essere molto inefficiente se hai tante concatenazioni (es. in un ciclo).

Lo StringBuilder, invece, è **mutabile**: puoi modificare direttamente il contenuto senza creare ogni volta un nuovo oggetto.

### Esempio semplice:

StringBuilder sb = new StringBuilder();

sb.append("Ciao");

sb.append(" ");

sb.append("mondo!");

System.out.println(sb.toString()); // Output: Ciao mondo!

### Metodi principali:

* append(String s) – aggiunge testo alla fine
* insert(int offset, String s) – inserisce testo a una certa posizione
* delete(int start, int end) – elimina una parte del testo
* reverse() – inverte il contenuto
* toString() – converte lo StringBuilder in una String

### Confronto prestazioni:

// Inefficiente con String

String s = "";

for (int i = 0; i < 1000; i++) {

s += i;

}

// Efficiente con StringBuilder

StringBuilder sb = new StringBuilder();

for (int i = 0; i < 1000; i++) {

sb.append(i);

}

### Quando usare StringBuilder:

* Se stai facendo molte concatenazioni o modifiche su stringhe
* Se vuoi ottimizzare le prestazioni del tuo codice

Se ti serve **thread safety** (cioè sicurezza nei programmi con più thread), puoi usare invece StringBuffer, che è come StringBuilder ma sincronizzato (più lento, ma sicuro in ambienti multi-thread).

Metodi StringBuilder:

trim(): rimuove gli spazi bianchi tra i char in questo caso.

Il programma utilizza la mappatura, per avere a portata tutte le traduzioni del Morse.