Titolo: Giustizia Sociale e Progettazione Inclusiva in C#

Cos'è la giustizia sociale in ambito digitale e informatico?

- Equità, diritti, pari opportunità, inclusione.
- Responsabilità del programmatore come cittadino digitale.
- Scelte di progettazione che possono includere o escludere utenti.
- Concetti di **accessibilità** e **inclusività** nel software.

Giustizia sociale nel contesto digitale

Anche in programmazione è possibile "commettere ingiustizie".
Per evitarlo è necessario:

- **Garantire l'accessibilità**: tutti devono avere un accesso equo a tecnologie, dati e opportunità digitali.
- **Contrastare il digital divide**: alcune persone rischiano di essere escluse per motivi economici, geografici o culturali.
- **Progettare sistemi non discriminatori**, che tengano conto anche dei gruppi socialmente meno rappresentati (etica degli algoritmi).
    - Software Open Source e Software libero: strumenti che favoriscono l'uguaglianza nell'accesso alla conoscenza.
    - Cyberbullismo e hate speech: la giustizia sociale include il rispetto online.
- **Evitare bias negli algoritmi**, cioè comportamenti discriminatori dovuti a presupposti sbagliati o dati distorti.

Esempi concreti

- Una piattaforma di e-learning che funziona anche offline, per chi non ha una connessione stabile.
- Un'app pubblica con interfaccia accessibile a persone con disabilità visive.
- Sistemi scolastici che offrono opzioni multilingua per studenti stranieri.

Domande importanti da porsi

- Quali gruppi sociali potrebbero essere esclusi da un software progettato senza attenzione?
- Cosa può fare un programmatore per garantire inclusione nei propri progetti?
- Quando un software diventa discriminatorio?
- Conosci app o siti che non sono inclusivi?

Il ruolo del programmatore

- Il programmatore non crea solo codice: costruisce una parte della società digitale.
- Le sue scelte influenzano chi ha accesso a servizi, informazioni e diritti.

- Essere un cittadino digitale significa assumersi responsabilità etiche e sociali.
- Bisogna considerare l'impatto dei dati e degli algoritmi sulle persone ed evitare pregiudizi.

Esempio:
un'app che suggerisce solo professioni "maschili" a tutti gli utenti rinforza stereotipi e ingiustizie.

Progettare in modo etico significa chiedersi: Chi includo? Chi lascio fuori? Chi penalizzo senza volerlo?

Progettazione inclusiva in C#

Le classi sono modelli della realtà.
Se un modello è distorto o incompleto, anche il software può diventare discriminatorio.

Esempio:
un sistema che gestisce utenti non dovrebbe presupporre categorie rigide (solo due generi, una sola lingua, ecc.).

Quando progettiamo classi e sistemi, compiamo scelte che hanno conseguenze anche etiche.

È importante includere attributi e comportamenti che tengano conto di:

- accessibilità
- equità
- diversità degli utenti

3.1 Incapsulamento e privacy

Un principio etico fondamentale è proteggere i dati degli utenti.
Non tutte le informazioni devono essere accessibili o pubbliche.

```
class Utente
{
    // Attributo privato → protegge la privacy
    private string password;
    public string Nome { get; private set; }
    public int Età { get; set; }
    public bool SetPassword(string pwd)
    {
        if (pwd.Length >= 8)
        {
            password = pwd;
            return true;
        }
        else
            return false;  //Password troppo debole!
    }
    public bool CheckPassword(string input)
    {
        return password == input;
    }
}
```

Proteggere i dati è una SCELTA ETICA.

3.2 Inclusione e accessibilità

Molto spesso le classi vengono progettate in modo semplice, ma anche limitante. Consideriamo questa classe:

```
class Persona
{
    public string Nome { get; private set; }
    private string genere;
    public string Genere
    {
        get { return genere; }
        private set
        {
            if (value.ToUpper() == "M")
                genere = "M";
            else
                genere = "F";
        }
    }
    public string LinguaPreferita { get; set; }
}
```

Questa classe è inclusiva? No. Ci sono pregiudizi? Sì.

- Il campo *genere* permette solo due scelte, escludendo persone che non si riconoscono nelle categorie previste.
- *LinguaPreferita* accetta una sola lingua, ignorando le persone multilingue.
- Non prevede attributi che potrebbero migliorare l'accessibilità.

Versione più inclusiva della classe

```
class Persona
{
    public string Nome { get; private set; }
    private string genere;
    public string Genere
    {
        get { return genere; }
        private set
        {
            if (value.ToUpper() == "M")
                genere = "M";
            else
                if (value.ToUpper() == "F")
                genere = "F";
            else
                genere = "ALTRO";
        }
    }
    private List<string> linguaPreferita = new List<string>();
    public string LinguaPreferita
    {
        get
        {
            string s = string.Empty;
            linguaPreferita.ForEach(x => s += x + "\t");
            return s;
        }
        set { linguaPreferita.Add(value); }
    }
    private bool HasDisability { get; set; }

    public string Accessibilità()
    {
        if (HasDisability)
            return "Modalità accessibile attivata!";
        else
            return "";
    }
}
```

```
public string? Genere{ get; set; }
```