Analisi: 08CrownTime_x

Analisi del problema:

Il problema chiede di creare un programma che simuli, tramite i costrutti del multithreading, dei test su delle colture di batteri. Bisogna simulare la crescita della popolazione, inserendo il tempo di raddoppio dei batteri, e il consumo di glucosio, che permette ai batteri di vivere ed effettuare la mitosi. Si dovrà quindi inserire, oltre al tempo di raddoppio, la quantità di glucosio per ogni coltura e la popolazione iniziale delle colture. Infine calcolare e mostrare il tempo di crescita della popolazione fino al termine del glucosio e indicare la coltura con il più alto rapporto tra la popolazione massima raggiunta al termine del glucosio e la quantità iniziale di quest'ultimo.

Classe: Batteri

Attributi della classe:

Attributo	Tipo	Descrizione e utilizzo
doublingTime	Variabile double.	Contiene il tempo, in minuti, necessario alle cellule per effettuare la mitosi.
qGl	Variabile double.	Indica la quantità di glucosio a disposizione per i batteri.
qGlH	Variabile double	Indica la quantità di glucosio consumata dai batteri
pI	Variabile intera.	Indica la popolazione di batteri.
iOCl	Oggetto **InOutCtrl**.	Puntatore dell'oggetto **InOutCtrl** passato al costruttore della classe.
del	Delegate **assist**.	Delegato che punterà al metodo **Display** della classe. Non ritorna nulla e chiede un parametro double nella firma.

Costruttore della classe:

- **Firma:**
  `public` **`Batteri`**`(InOutCtrl` **`iOCl`**`, double` **`doublingTime`**`, int` **`pI`**`, double` **`qGlH`**`)`
- **Descrizione:** il costruttore inizializzerà l'oggetto impostando gli attributi della classe.

Metodi della classe:

Metodo Vivi:

- **Firma**: `public void` **`Vivi`**`()`
- **Descrizione:** Questo metodo simulerà la vita dei batteri aumentando la popolazione e diminuendo il glucosio, contando anche il tempo necessario a terminare il glucosio.

Metodo Display:

- **Firma del metodo:** `private void` **`Diplay`**`(double` **`i`**`)`
- **Descrizione:** questo metodo sarà puntato dal delegato **`del`**. Renderà la label di output visibile nell'oggetto **`iOCl`** e ne imposterà il testo passando i valori della classe.

Classe: MainForm

Attributi della classe:

Nome Attributo	Tipo	Descrizione e utilizzo
threads	Array di una dimensione di **thread**.	Vettore che conterrà 10 istanze della classe **thread**.
test	Array di una dimensione della classe **Batteri**.	Vettore che conterrà 10 istanze della classe **Batteri**.
running	Variabile booleana.	Indicherà se l'applicazione è in esecuzione o in attesa di eventi.
del	Delegate **MyDelegate**.	Delegato che punterà ai metodi **ChBtnStrt** e **ChIOCltxtBxesRdOnly** della classe. Non ritorna nulla e non richiede parametri.
iOCL	Vettore di oggetti **InOutCtrl**.	Vettore che conterrà i 10 **InOutCtrl** che verranno inizializzati nel form dal costruttore della classe.

Metodi della classe:

Metodo Run:

- **Firma**: `private void Run()`
- **Descrizione:** Questo metodo farà partire i thread e aspetterà che terminino la loro esecuzione per eseguire l'operazione di controllo dei rapporti. Questo metodo verrà eseguito da un thread.

Metodo SetCtrl:

- **Firma**: `private void SetCtrl()`
- **Descrizione:** Questo metodo verrà eseguito dal costruttore ed inizializzerà i controlli **InOutCtrl** nel form.

Metodo ChUse:

- **Firma**: `private void ChUse(int cOut)`
- **Descrizione:** Questo metodo cambierà lo stato dei controlli rendendoli inutilizzabili se attivi e viceversa. Permette anche la visualizzazione del test con rapporto migliore

Metodo ControlTxtBxs:

- **Firma**: `private void ControlTxtBxs()`
- **Descrizione:** Questo metodo cambierà lo stato delle textBox rendendole inutilizzabili se non verranno prima immessi i valori del tempo di raddoppio e della popolazione iniziale.

Eventi della classe:

Evento click bottone start:

- **Firma del metodo:** `private void btnStrt_Click(object sender, EventArgs e)`
- **Descrizione:** Questo evento istanzia gli oggetti della classe **Batteri** e i thread passando al costruttore i valori inseriti nei controlli del form. Infine verrà inizializzato il thread che eseguirà il metodo **Run.**

Evento click bottone Auto:

- **Firma del metodo:** `private void btnAut_Click(object sender, EventArgs e)`
- **Descrizione:** Questo evento imposterà automaticamente dei valori nelle text box dei controlli **InOutCtrl**.

Evento chiusura form:

- **Firma del metodo:** `private void Form1_FormClosing(object sender, EventArgs e)`
- **Descrizione:** Questo evento terminerà i thread in esecuzione, per chiudere il form in sicurezza.

Classe: InOutCtrl

Attributi della classe:

Nome Attributo	Tipo	Descrizione e utilizzo
running	booleano.	Indicherà se l'applicazione è in esecuzione o in attesa di eventi.
startWidth	Variabile statica intera.	Variabile contenente la larghezza del controllo utente.

Metodi della classe:

Metodo Ch_Label_Visible:

- **Firma**: `public void ChLblOutVisible()`
- **Descrizione:** Questo metodo cambia l'attributo della visibilità della label di output.

Metodi Get e Set:

- **Firme**:
  - `public void SetLblOut(string txt)`
  - `public void SetLblGl(string txt)`
  - `public bool GetLblOutVisible()`
  - `public void SetTxtBxGl(string txt)`
  - `public string GetTxtBxGl()`
  - `public void SetRunning(bool running)`
  - `public bool GetRunning()`
  - `public void SetTxtBxesReadOnly(bool tOf)`
- **Descrizione:** i metodi Get e Set permettono di modificare(Set) e di recuperare(Get) i dati contenuti negli attributi dell'oggetto.