Corso sul linguaggio Java

Modulo JAVA7

B3–Gestione di una coda

Prerequisiti

- Programmazione base in Java
- Utilizzo di classi e oggetti
- Tecnica di allocazione dinamica
- Concetto di coda e operazioni relative

Introduzione

In questa Unità vediamo come realizzare in pratica applicazioni che implementano una struttura astratta pila (stack), considerando per semplicità la coda formata da numeri interi.

Esaminiamo in particolare i seguenti casi:

- implementazione **sequenziale**
- implementazione **a lista concatenata**
- utilizzo di **Queue**

Implementazione sequenziale

ATTIVITA': considerando le operazioni logiche presentate nello studio teorico, scrivere la UML della classe *queue* implementata con vettore.

La UML della classe è la seguente

queue

-**final int** max; // dimensione
int elemento[]=**new int** [max];
static int in; out // estremità coda

+ queue(); // costruttore
+ **void** insQueue (**int** x); // inserimento
+ **void** delQueue (); // eliminazione
+ **static boolean** IsEmpty (); // vuota?
+ **void** empty (); // svuota
+ **int** next (); // prossima uscita
+ **booolean** IsFull (); // piena?
+ **void** print (**Stringa** str); // stampa la coda

Implementazione sequenziale
- la classe *queue*

```
public class queue                    // implementazione classe queue
{   final int max=3;
    int elemento[]=new int[max];
    static int in, out;
    public queue()  {… }
    public void insQueue (int x)  {…}
    public void delQueue() {... }
    public static boolean IsEmpty()   {….}
    public void empty()  {… }
    public int next()  {… }
    public void print()  {… }
    public boolean IsFull() {… }
    public int lenQueue()  {…}
    public static void main(String args[])  {…        }
} //end class
```


Implementazione sequenziale
- il metodo insQueue()

ATTIVITA': implementare il metodo *insQueue*() della classe *queue*.

```
public void insQueue (int x)
{   if (in<0)
      System.out.println("Coda piena!");
    else
    {
          elemento[in] = x;
          in--;
    }
}
```

Implementazione sequenziale
- il metodo delQueue()

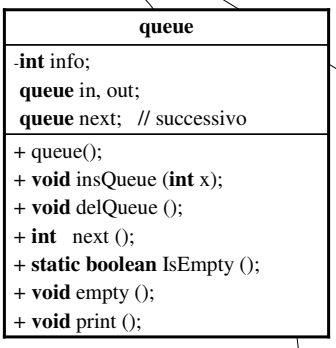
ATTIVITA': implementare il metodo *deQueue*() della classe *queue*.

```
public void delQueue()
{   if (!IsEmpty())
    {     int i;
          for (i = out; i > in+1; i--)
            elemento[i] = elemento[i-1];
          in++;
    }
    else
       System.out.println ("Coda vuota!");
}
```


Implementazione a lista

ATTIVITA': considerando le operazioni logiche presentate nello studio teorico, scrivere la UML della classe *queue* implementata a lista.

La UML della classe è la seguente



Implementazione a lista
- il metodo insQueue()

ATTIVITA': implementare il metodo *insQueue*() della classe *queue*.

```
public void insQueue (int x)
{   queue t =  new queue();
    t.info = x;
    if (IsEmpty())
    {       in = t;
            t.next = null;
            out = t;
    }
    else
    {       out.next = t;
            out = t;
    }
}
```


Implementazione a lista
- il metodo delQueue()

ATTIVITA': implementare il metodo *delQueue*() della classe *queue*.

```
public void delQueue()
{
        in = in.next;
}
```

La classe **Queue** di Java

Java fornisce l'interfaccia **Queue** per rappresentare strutture a coda di diverso tipo, con i seguenti metodi:

METODO	EFFETTO
Queue *s* = **new Queue**();	Costruttore
Object peek()	Restituisce il prossimo da estrarre senza estrarlo
Object remove()	Restituisce il prossimo da estrarre e lo estrae
Boolean add(**Object** *o*)	Inserisce *o* e dà errore se coda piena
int size()	Restituisce il numero di elementi in coda

La classe **Queue** di Java

Un semplice esempio di applicazione della classe **Queue**.

Un particolare tipo di coda, **PriorityQueue**

```
import java.util.*;
public class classeQueue  //utilizza Queue
{    public static void main (String args[])
   {        PriorityQueue q = new PriorityQueue();
            System.out.println("Contiene " + q.size() + " elementi");
            for (int i=1; i<=10; i++)  q.add(i);
            System.out.println("Contiene " + q.size() + " elementi");
            System.out.println("Il prossimo e' " + q.peek());
            q.remove();
            System.out.println("Il prossimo e' " + q.peek());
    }
} //end class
```

Argomenti

- Implementazione sequenziale
  - la classe *queue*
  - il metodo *insQueue()*
  - il metodo *delQueue*()
- Implementazione a lista
  - il metodo *insQueue()*
  - il metodo *delQueue*()
- La classe **Queue** di Java