La classe Stack e l'interfaccia Queue nel package java.util

Un'implementazione della struttura di dati dinamica di pila è contenuta nel package **java.util** ed è descritta dalla classe **Stack**.

La classe *Stack* è una sottoclasse della classe *Vector* e implementa i seguenti cinque metodi.

Metodo	Descrizione
empty()	Restituisce il valore *true* se la pila è vuota, altrimenti *false*.
peek()	Restituisce l'oggetto in cima alla pila senza rimuoverlo.
pop()	Restituisce l'oggetto in cima alla pila e lo rimuove.
push(oggetto)	Inserisce un oggetto in cima alla pila.
search(oggetto)	Restituisce la posizione dell'oggetto nella pila a partire dalla cima.

Il costruttore della classe *Stack* non richiede parametri. Una generica pila può essere dichiarata e creata nel seguente modo:

```
Stack pila = new Stack();
```

Con le versioni più recenti di Java è possibile specificare il tipo degli oggetti che vengono inseriti nelle strutture dinamiche. Per esempio, una pila composta da oggetti di classe *Integer* può essere dichiarata nel seguente modo:

```
Stack<Integer> pila = new Stack<Integer>();
```

Il nome della classe relativa agli oggetti inseriti nella pila, viene indicata tra i simboli di minore e maggiore, dopo la parola *Stack*. L'indicazione esplicita del tipo permette, in fase di recupero dell'oggetto dalla pila, di non dover eseguire l'operazione di casting.

La soluzione del progetto seguente è stata presentata nel libro di testo con l'uso della classe *Vector*. Adesso viene mostrata la soluzione con la classe predefinita *Stack*.

ESEMPIO

Popolare una pila con dieci numeri casuali e svuotarla mostrando che l'ordine degli elementi viene invertito, dall'ultimo al primo.

Rispetto alla soluzione nel libro, oltre alla dichiarazione tramite la classe *Stack*, cambia il comando con cui un oggetto viene prelevato dalla coda.
La seguente istruzione,

```
numObj = (Integer) pila.pop();
```

viene sostituita con:

```
numObj = pila.pop();
```

Il codice completo del programma è riportato di seguito.

PROGRAMMA JAVA (ProgPila.java)

```
import java.util.*;

class ProgPila
{
   public static void main(String argv[])
   {
      // crea una pila vuota
      Stack<Integer> pila = new Stack<Integer>();
      int num;
      Integer numObj;

      // aggiunge elementi alla pila
      for(int i=0; i<10; i++)
      {
         num = (int) (Math.random()*100);
         numObj = new Integer(num);
         System.out.print(numObj + " ");
         pila.push(numObj);
      }

      System.out.println("\nElementi nella pila: " + pila.size());

      // toglie gli elementi e li visualizza
      while (!pila.empty())
      {
         numObj = pila.pop();
         System.out.print(numObj + " ");
      }

      System.out.println();
   }
}
```

Una descrizione della struttura di dati dinamica di coda è contenuta nel package **java.util** ed è rappresentata dall'interfaccia **Queue**.

L'interfaccia *Queue* non può essere usata per generare degli oggetti, ma impone, alle classi che la utilizzano di implementare i seguenti metodi.

Metodo	Descrizione
element()	Restituisce l'oggetto alla fine della coda senza rimuoverlo.
offer(oggetto)	Inserisce un oggetto all'inizio della coda.
peek()	Restituisce l'oggetto alla fine della coda senza rimuoverlo.
poll()	Restituisce l'oggetto alla fine della coda e lo rimuove.
remove()	Restituisce l'oggetto alla fine della coda e lo rimuove.

Un esempio di implementazione dell'interfaccia *Queue*, in cui sono presenti i precedenti metodi, è la classe **LinkedList**. Questa classe, oltre a fornire i metodi di gestione della coda, implementa i metodi usati dalla struttura di dati di lista concatenata.

La soluzione del progetto seguente è stata presentata nel libro di testo con l'uso della classe *Vector*. Adesso viene mostrata la soluzione con la classe predefinita *LinkedList*.

ESEMPIO

Creare un programma che simula il funzionamento di un bar.

Le ordinazioni vengono organizzate in una coda, dichiarata come un oggetto di classe *LinkedList*, nel seguente modo:

```
LinkedList codaOrdinazioni = new LinkedList();
```

Un ordine, realizzato come istanza della classe *Ordine*, viene aggiunto alla coda delle ordinazioni nel momento in cui viene generato dal cliente. Successivamente i gestori del bar prelevano le ordinazioni dalla coda e le soddisfano.

Per indicare che la coda accetta solo oggetti di classe *Ordine*, nella dichiarazione della coda si può specificare il tipo generico degli oggetti nel seguente modo:

```
LinkedList<Ordine> codaOrdinazioni = new LinkedList<Ordine>();
```

Il seguente programma simula la sequenza di operazioni: ordina, ordina, ordina, soddisfa, ordina, soddisfa, soddisfa, soddisfa.

IMPLEMENTAZIONE DELLA CLASSE (Ordine.java)

```
class Ordine
{
   private String tavolo;
   private String ordine;

   public Ordine(String tav, String ord)
   {
      tavolo = tav;
      ordine = ord;
   }

   public void stampa()
   {
      System.out.println("Tavolo: " + tavolo);
      System.out.println("Ordine: " + ordine);
   }
}
```

PROGRAMMA JAVA (ProgBar.java)

```
import java.util.*;

class ProgBar
{
   public static void main(String argv[])
   {
      LinkedList<Ordine> codaOrdinazioni = new LinkedList<Ordine>();
      Ordine ord;

      ord = new Ordine("05", "4 caffe'");
      codaOrdinazioni.offer(ord);
      System.out.println("--Aggiunto ordine");

      ord = new Ordine("03", "2 bibite");
      codaOrdinazioni.offer(ord);
      System.out.println("--Aggiunto ordine");

      ord = new Ordine("11", "1 caffe' e 2 bibite");
      codaOrdinazioni.offer(ord);
      System.out.println("--Aggiunto ordine");

      ord = codaOrdinazioni.poll();
      System.out.println("Soddisfatto ordine:");
      ord.stampa();

      ord = new Ordine("15", "1 cappuccino");
      codaOrdinazioni.offer(ord);
      System.out.println("--Aggiunto ordine");

      ord = codaOrdinazioni.poll();
      System.out.println("Soddisfatto ordine:");
      ord.stampa();

      ord = (Ordine) codaOrdinazioni.poll();
      System.out.println("Soddisfatto ordine:");
      ord.stampa();

      ord = (Ordine) codaOrdinazioni.poll();
      System.out.println("Soddisfatto ordine:");
      ord.stampa();
   }
}
```