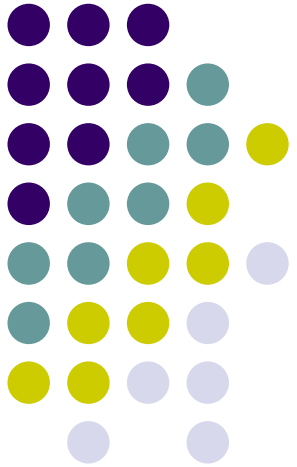
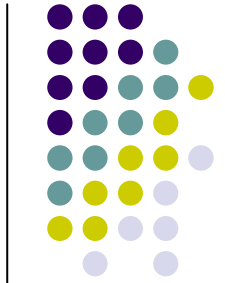
Deque

Corso: Strutture Dati

Docente: Annalisa De Bonis



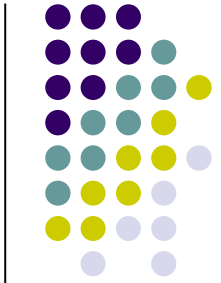
Il TDA deque



- Il TDA deque e` un TDA simile alla coda che però supporta cancellazioni e inserimenti da entrambe le estremita`

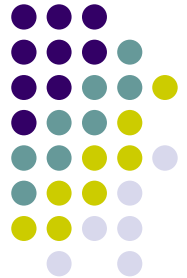
- Si pronuncia deck

Operazioni su deque



- addFirst(e) Inserisce e all'inizio del deque
- addLast(e) Inserisce e alla fine del deque
- removeFirst() Restituisce e rimuove il primo elemento del deque
- removeLast() Restituisce e rimuove l'ultimo elemento del deque
- getFirst() Restituisce il primo elemento del deque
- getLast() Restituisce l'ultimo elemento del deque
- size() Restituisce il numero di elementi del deque
- isEmpty() Restituisce true se il deque e` vuoto e false altrimenti

Interfaccia di Deque



- Richiede la definizione di EmptyDequeException

```
public interface Deque<E> {
 public int size();
 public boolean isEmpty();
 public E getFirst() throws EmptyDequeException;
public E getLast() throws EmptyDequeException;
 public void addFirst (E element);
 public void addLast (E element);
public E removeFirst() throws EmptyDequeException;
public E removeLast() throws EmptyDequeException; }
```

Un'implementazione di Deque basata sulle liste

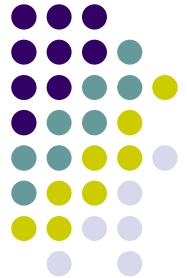
- Si usano liste doppiamente lincate
  - Ciascun nodo contiene un riferimento (next) al nodo successivo e un riferimento (prev) al nodo precedente

- Vantaggio: cancellazione in tempo costante

⇒ removeLast() in tempo costante

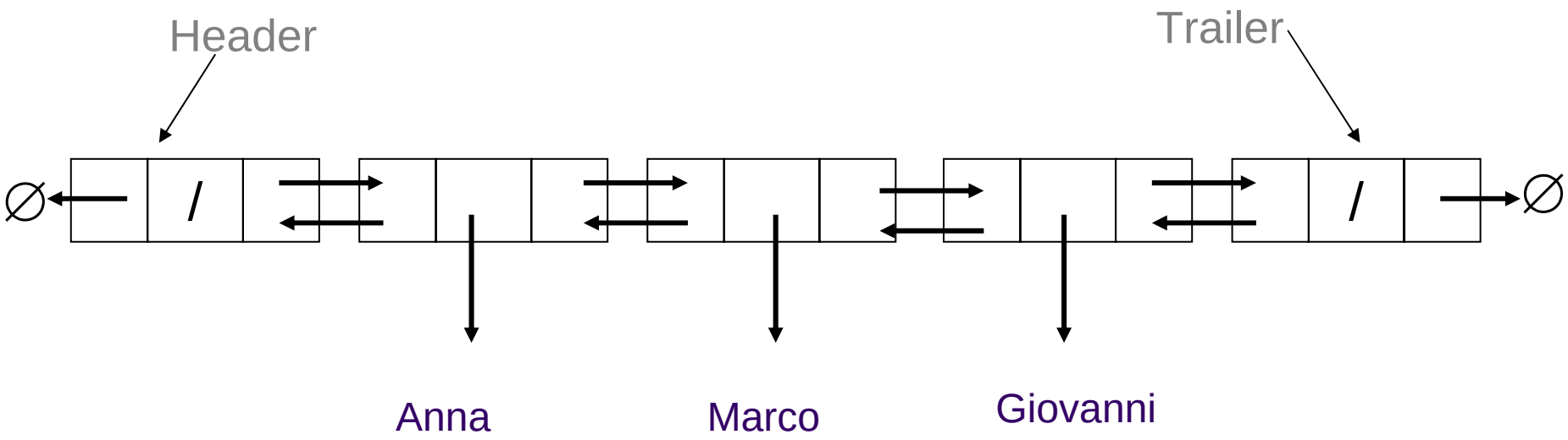
  - Nelle liste a link singoli la cancellazione di un nodo x richiede tempo proporzionale al numero di nodi che precedono x

Un implementazione di Deque basata sulle liste

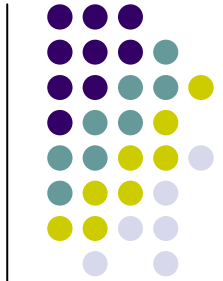


- Per semplificare la scrittura del codice si usano due nodi sentinella
  - header:
    - Il campo next contiene un riferimento al primo nodo della deque
    - Il campo prev e` settato a null
  - trailer
    - Il campo prev contiene un riferimento all'ultimo nodo della deque
    - Il campo next e` settato a null

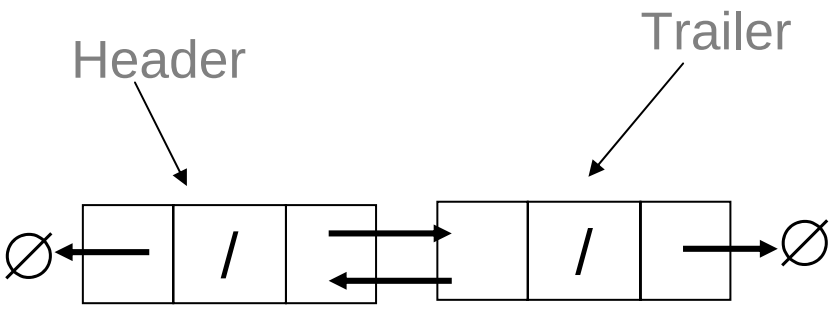
Esempio



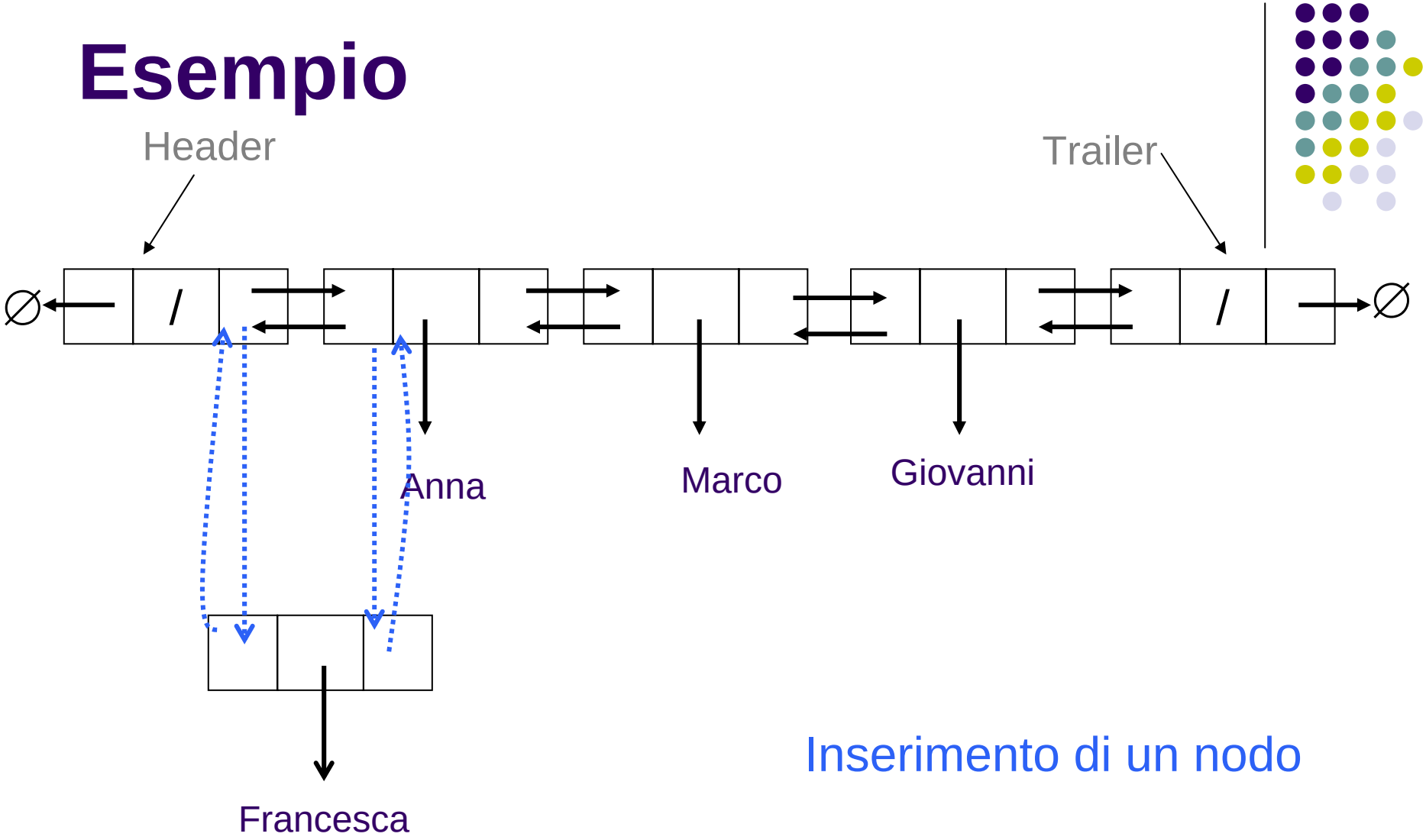
Esempio



Deque vuoto

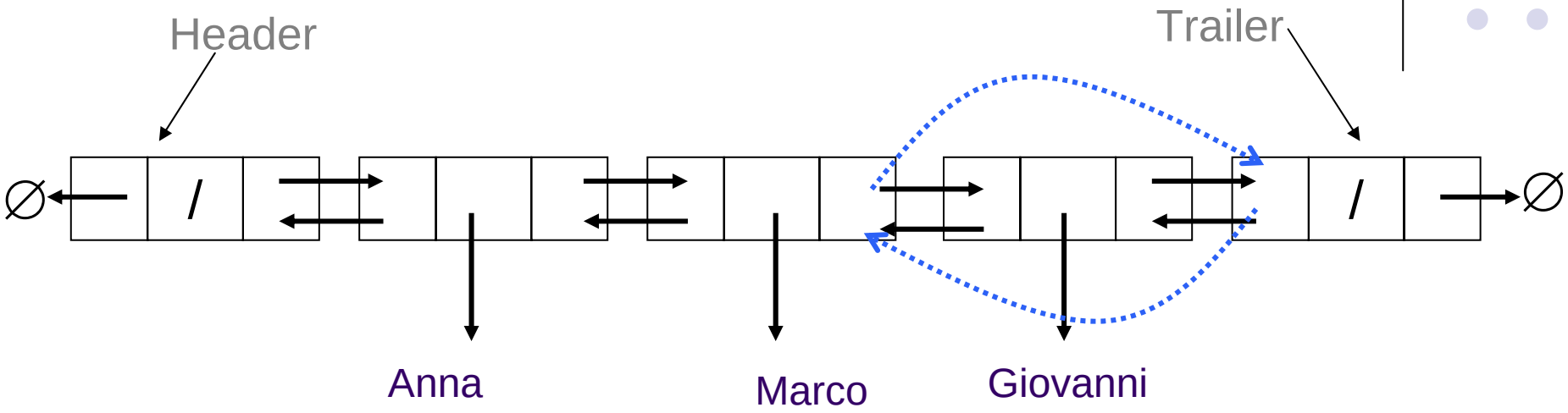


Esempio



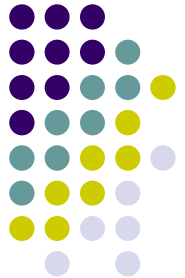
Inserimento di un nodo

Esempio



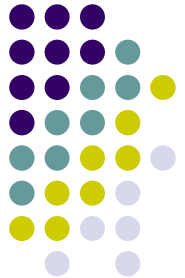
Cancellazione di un nodo

Metodi dei nodi



- Un nodo di una lista doppiamente lincata deve supportare i seguenti metodi
  - setElement(E e)
  - setNext(DLNode<E> newNext)
  - setPrev(DLNode <E>newPrev)
  - getElement()
  - getNext()
  - getPrev()

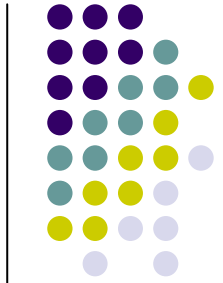
La classe DLNode



```
public class DLNode<E> {
  private E element;
  private DLNode<E> next, prev;
  DLNode() { this(null, null, null); }
  DLNode(E e, DLNode<E> p, DLNode<E> n) {
    element = e;
    next = n;
    prev = p;
  }
```

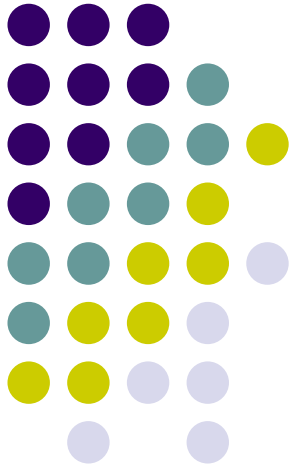
Continua nella prossima slide

La classe DLNode

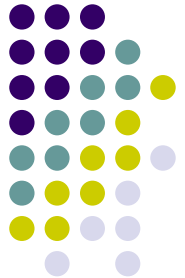


```
//Metodi di modifica
public void setElement(E newElem) {
     element = newElem; }
  public void setNext(DLNode<E> newNext) {
     next = newNext; }
  public void setPrev(DLNode<E> newPrev) {
     prev = newPrev; }
//Metodi di Accesso
 public E getElement() { return element; }
  public DLNode<E> getNext() { return next; }
  public DLNode<E> getPrev() { return prev; }
}
```

Design pattern: Adapter



Design Pattern



- Un design pattern fornisce un metodo generale per risolvere molte situazioni differenti
- Un design pattern descrive
  - Gli scenari in cui può essere applicato
  - Il modo in cui deve essere applicato
  - Il risultato della sua applicazione
- Design pattern per la progettazione di algoritmi
  - Ricorsione
  - *Divide-and-conquer*
  - Metodo *greedy*
  - Ecc.
- Design pattern per software engineering
  - Position
  - Iterator
  - Adapter
  - Ecc

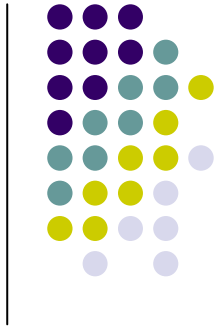
Adapter

- Metodologia di programmazione che consiste nell'adattare le funzionalita` di una classe preesistente a quelle della nuova classe che si intende costruire
  - In genere la nuova classe contiene un'istanza della vecchia classe come variabile di istanza nascosta
  - I metodi della nuova classe sono implementati usando i metodi della vecchia classe invocati per variabile di istanza nascosta

Esempi dell'uso dell'Adapter

- Specializzare una classe per implementare una classe piu` semplice
  - Esempio: Adattiamo la classe che implementa Deque in modo che possa essere utilizzata per implementare Stack
- Specializzare una classe affinche` funzioni solo con un certo tipo di dati
  - Esempio: Adattiamo la classe ArrayStack in modo che lo Stack contenga solo stringhe

Stack e Code implementate mediante Deque



Stack

Top() implementato con getLast()

Push() implementato con addLast()

Pop() implementato con removeLast()

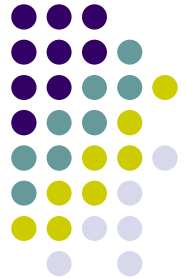
Coda

Front() implementato con getFirst()

Enqueue() implementato con addLast()

Dequeue() implementato con removeFirst()

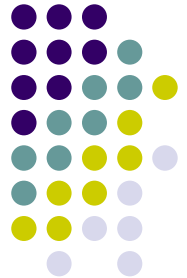
Implementazione di Stack con Deque



```
public class DequeStack <E> implements Stack <E>{
   private Deque <E> D;

   public DequeStack() {
   D = new MyDeque();
  }

  public int size() {
    return D.size();
  }

 public boolean isEmpty() {
    return D.isEmpty();
  }
  public void push(E e) {
    D.addLast(e);
  }
```

Continua nella prossima slide

Implementazione di Stack con Deque



```
public E top() throws EmptyStackException {
    try {
        return D.getLast();
       }
      catch (EmptyDequeException e) {
      throw new EmptyStackException("Stack vuoto!");
    }
  }
...
}
```