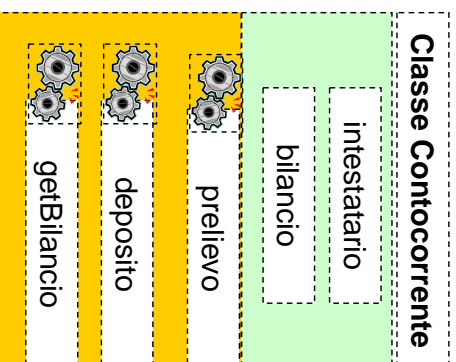
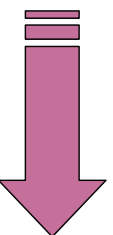
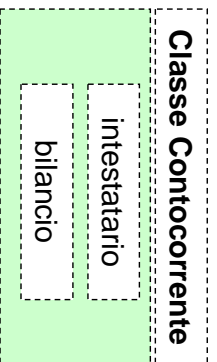


# Il conto corrente: definizione grafica

- Si vuole implementare una classe che simuli un conto corrente
- la classe dovrà contenere il bilancio del conto corrente e il nome dell'intestatario
- Le operazioni possibili dovranno essere quella del prelievo e quella del deposito e una operazione per conoscere il bilancio

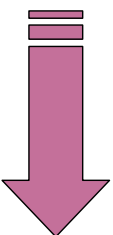
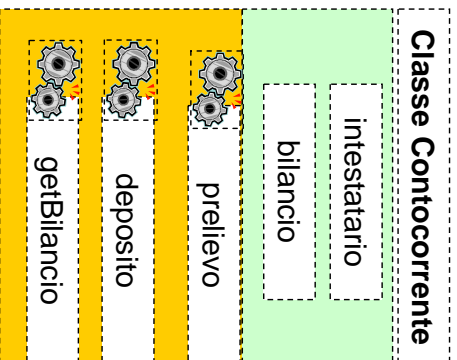


# Il conto corrente: definizione dei dati



```
class Contocorrente {  
    String intestatario;  
    float bilancio;  
}
```

# Il conto corrente: definizione dei metodi



```
class Contocorrente {  
    String intestatario;  
    float bilancio;  
  
    void prelievo(float cifra);  
    void deposito(float cifra);  
    float getBilancio();  
}
```

# Il conto corrente: definizione del costruttore

- Si definisca anche un costruttore che prende in ingresso il nome dell'intestatario

```
class Contocorrente {  
    String intestatario;  
    float bilancio;  
    Contocorrente(String i);  
  
    void prelievo(float cifra);  
    void deposito(float cifra);  
    float getBilancio();  
}
```

# Il conto corrente: implementazione dei metodi

```
class Contocorrente {  
    String intestatario;  
    float bilancio;  
  
    Contocorrente(String i){  
        intestatario=i;  
        bilancio=0;  
    }  
  
    void prelievo(float cifra){  
        bilancio=bilancio-cifra;  
    }  
}
```

```
    void deposito(float cifra){  
        bilancio=bilancio+cifra;  
    }  
  
    float getBilancio(){  
        return bilancio;  
    }  
}
```

amenti di Informatica 2006-07

5

## Il conto corrente: uso

Scrivere il codice per

- creare due conti correnti, uno di Paperone e uno di Paperino
- depositare 100 sul conto di Paperone e 10 su quello di Paperino
- spostare 5 dal conto di Paperino a quello di Paperone
- stampare il bilancio dei due conti correnti

```
Contocorrente a;  
Contocorrente b;
```

```
a = new Contocorrente("Paperone");  
b = new Contocorrente("Paperino");
```

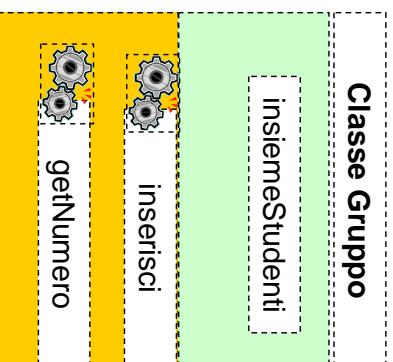
```
a.deposito(100);  
b.deposito(10);
```

```
b.prelievo(5);  
a.deposito(5);
```

```
System.out.println("Bilancio Paperone"  
    +a.getBilancio());  
System.out.println("Bilancio Paperino"  
    +b.getBilancio());
```

# Il gruppo di studenti

- Si vuole implementare una classe che simuli un insieme di studenti
- la classe dovrà contenere un insieme di studenti
- Le operazioni possibili dovranno essere quella dell'inserire uno studente e dell'ottenere il numero di studenti
- Il numero massimo di studenti del gruppo sarà indicato nel costruttore al momento della creazione



Franco Scarseli

Fondamenti di Informatica 2006-07

7

# Il gruppo: definizione di variabili, metodi e costruttore

- Si definisca variabili metodi e costruttori supponendo la seguente definizione di **Studiante**

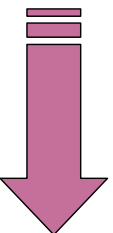
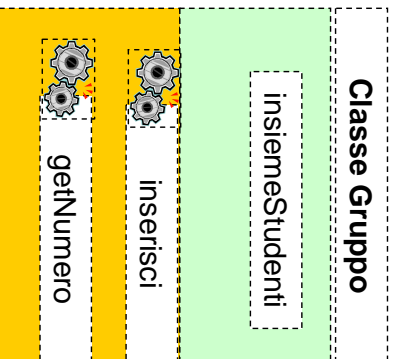
```
class Studente{  
    String nome;  
  
    Studente(String n){  
        nome=n;  
    }  
}
```

Franco Scarseli

Fondamenti di Informatica 2006-07

8

# Il gruppo: definizione di variabili, metodi e costruttore



```
class Gruppo {
    Studente insiemeStudenti[];
    int numeroStudenti;

    Gruppo(int maxStud);
    void inserisci(Studente s);
    int getNumero();
}
```

## Il gruppo: implementazione

```
class Gruppo {
    Studente insiemeStudenti[];
    int numeroStudenti;

    Gruppo(int maxStud){
        insiemeStudenti=
        new Studente[maxStud];
        numeroStudenti=0;
    }
}
```

```
void inserisci(Studente s){
    insiemeStudenti[numeroStudenti]=s;
    numeroStudenti= numeroStudenti+1;
}

int getNumero(){
    return numeroStudenti;
}
}
```



# Il gruppo: uso

Scrivere il codice per

- creare un gruppo con al più 10 studenti
- aggiungere gli studenti Homer, Bart, Lista
- stampare il numero di studenti del gruppo

```
Gruppo a;  
  
a = new Gruppo(a)  
  
a.inserisci(new Studente("Homer"));  
a.inserisci(new Studente("Bart"));  
a.inserisci(new Studente("Lista"));  
  
System.out.println("Numero studenti del gruppo:"  
+a.getNumero());
```