





Gerarchie di Tipi

Metodologie di progetto

- **Estensione di classi**
- **Implementazione di interfacce**
- **Composizione**

Notazione UML

<i>Relazione</i>	<i>Simbolo</i>	<i>Significato</i>
Ereditarietà		is-a
Implementazione		is-a
Aggregazione		has a
Dipendenza		usa

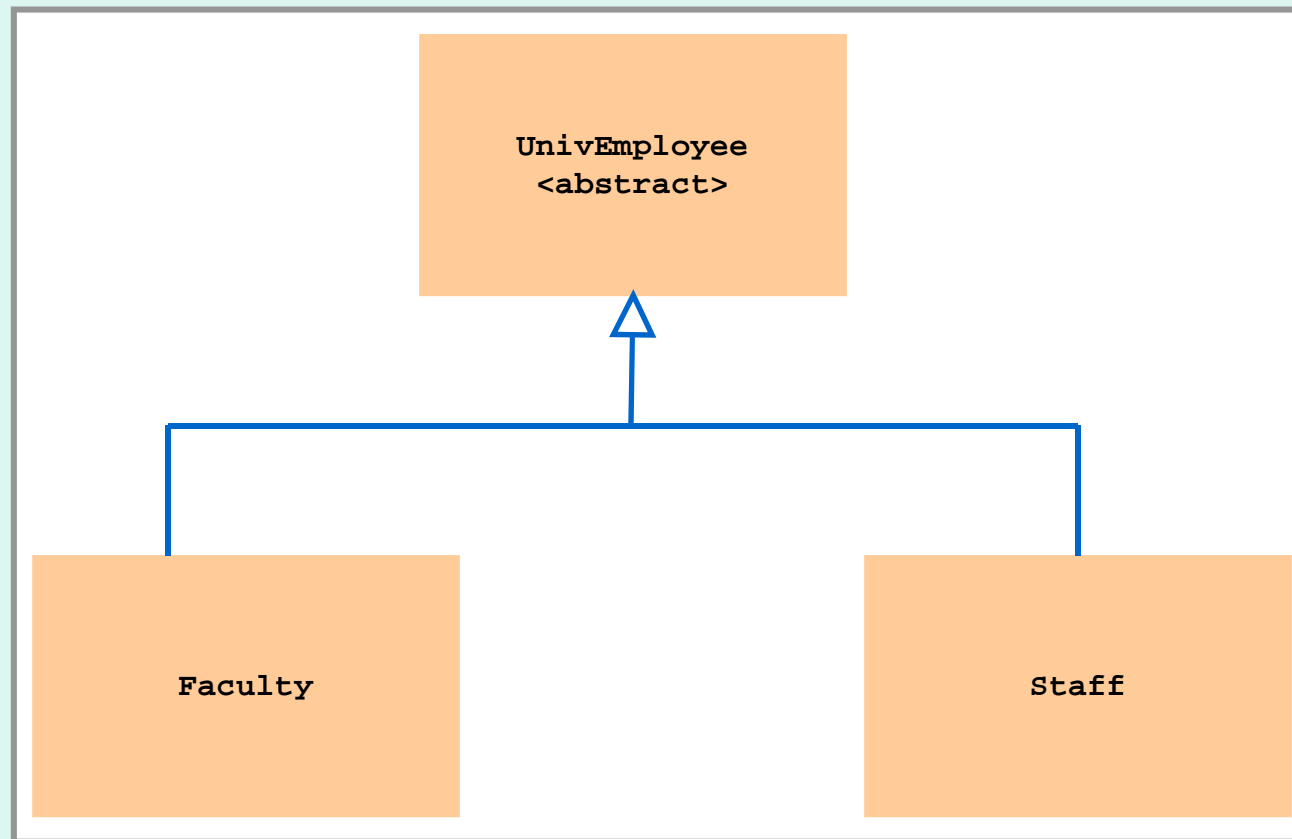
Una struttura universitaria

- Il personale è strutturato in
 - Personale Docente: **Faculty**
 - Personale Tecnico Amministrativo: **Staff**

Una struttura universitaria

- Introduciamo una terza categoria che descrive le due tipologie di personale
- **UnivEmployee**
 - Definisce le caratteristiche di struttura e comportamento comuni a tutto il personale
 - Intesa non per creare istanze proprie
 - definiamo **abstract** per realizzare queste scelte di progetto

Una struttura universitaria



University Employees

- A tutto il personale viene garantito l'uso di posta elettronica
- Rappresentiamo mediante un metodo

```
/**  
 * manda un messaggio all'oggetto su cui è invocato  
 */  
public void mailTo(String msg);
```

- **Includiamo il metodo in UnivEmployee**
 - così che venga ereditato dalle due classi derivate

University Employees

```
public abstract class UnivEmployee
{
    private List<String> mailbox = new ArrayList<String>();
    public void mailTo(String msg)
    { mailbox.add(msg); }
    . . .
}

public class Faculty extends UnivEmployee
{
    . . .
}

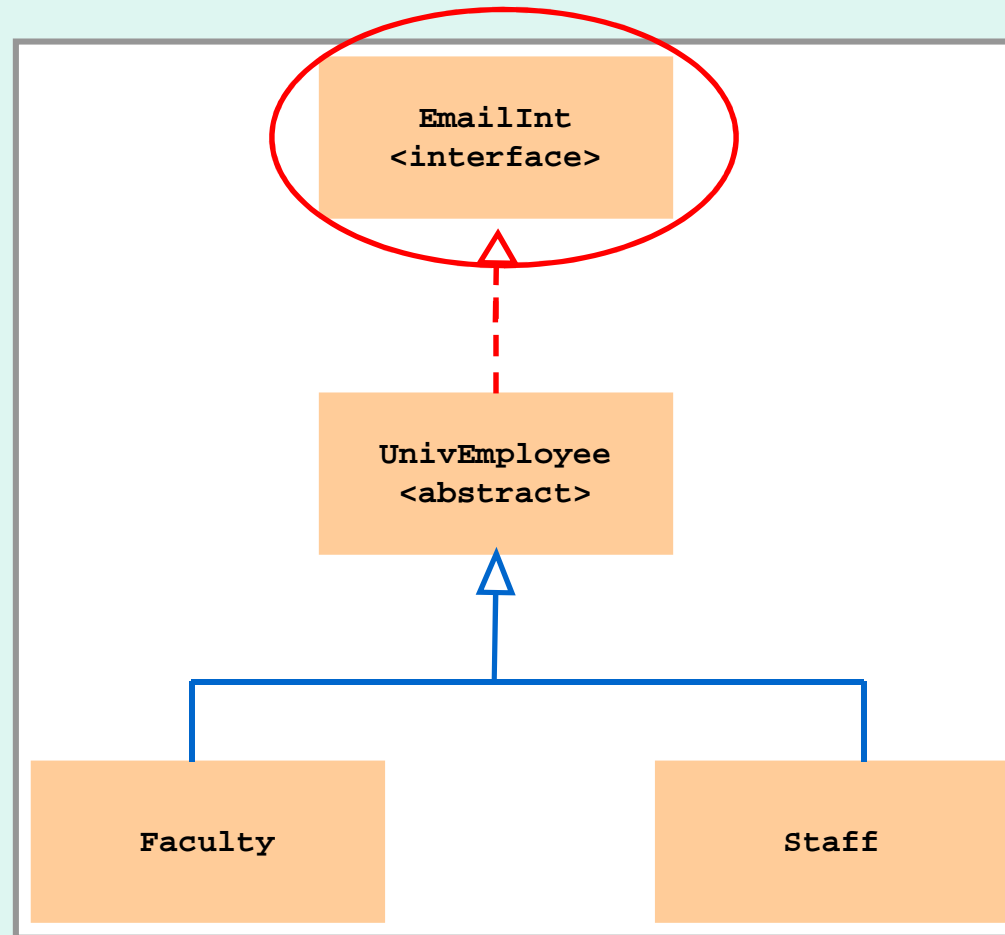
public class Staff extends UnivEmployee
{
    . . .
}
```


University Employees

- L'accesso alla posta elettronica è una proprietà generale
- Possiamo concettualizzare questa proprietà
 - una interfaccia

```
interface EmailInt
{
    public void mailTo(String msg);
}
```

University Employees con email



University Employees con email

```
abstract class UnivEmployee implements EmailInt
{
    . . .
    public void mailTo(String msg) { . . . }
}

class Faculty extends UnivEmployee
{
    . . .
}

class Staff extends UnivEmployee
{
    . . .
}
```

University Employees

- Al personale docente viene inoltre garantita la possibilità di gestire una propria pagina web

```
/**  
 * restituisce la URL associata all'oggetto  
 */  
public Url browse();
```

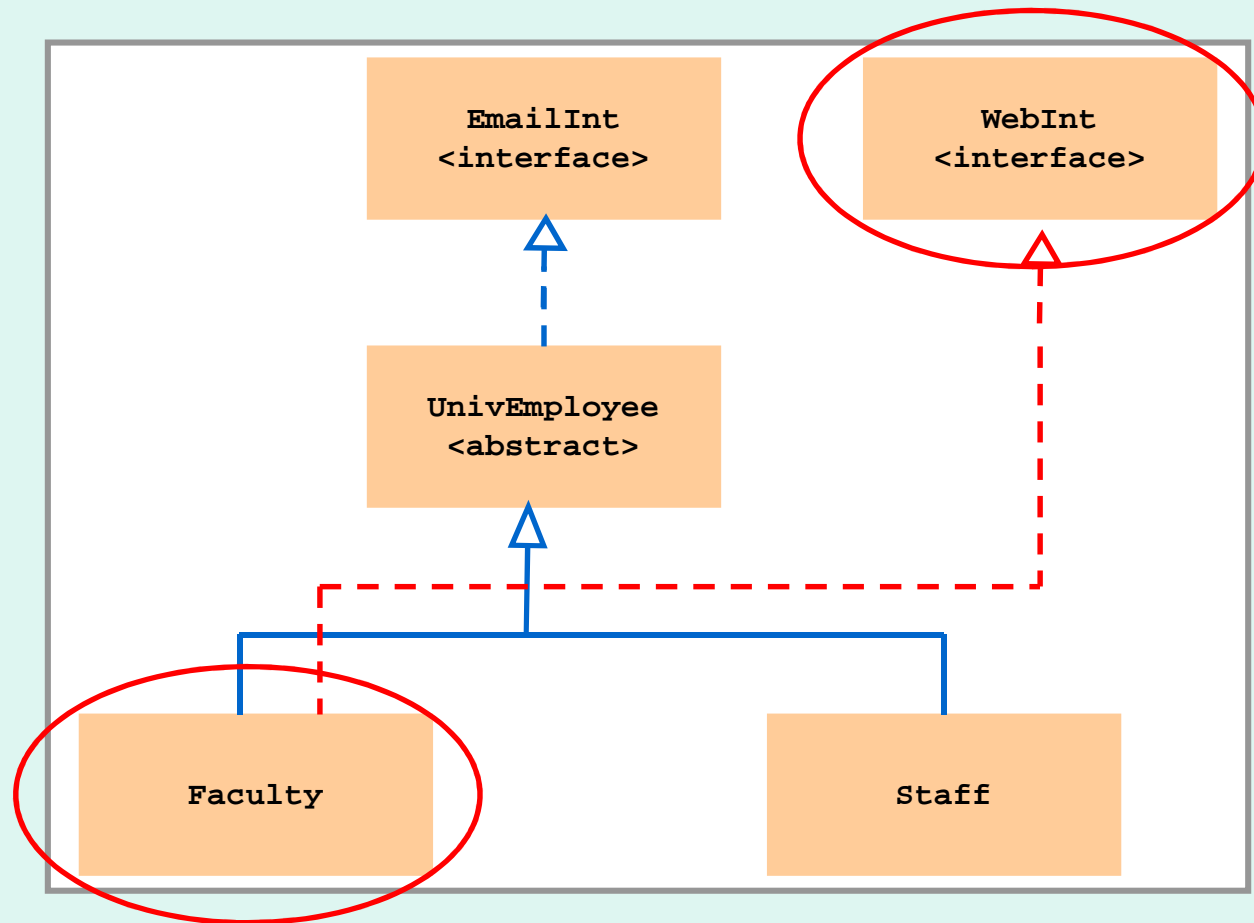
- Definito solo per la classe `Faculty`

University Employees ... on the web

- Anche qui, possiamo generalizzare introducendo una nuova interfaccia

```
interface WebInt
{
    public Url browse();
}
```

University Employees ... on the web



University Employees ... on the web

```
public abstract class UnivEmployee implements EmailInt
{
    . . .
    public void mailTo(String msg) { . . . }
}

public class Faculty extends UnivEmployee implements WebInt
{
    public Url browse() { . . . }
    . . .
}

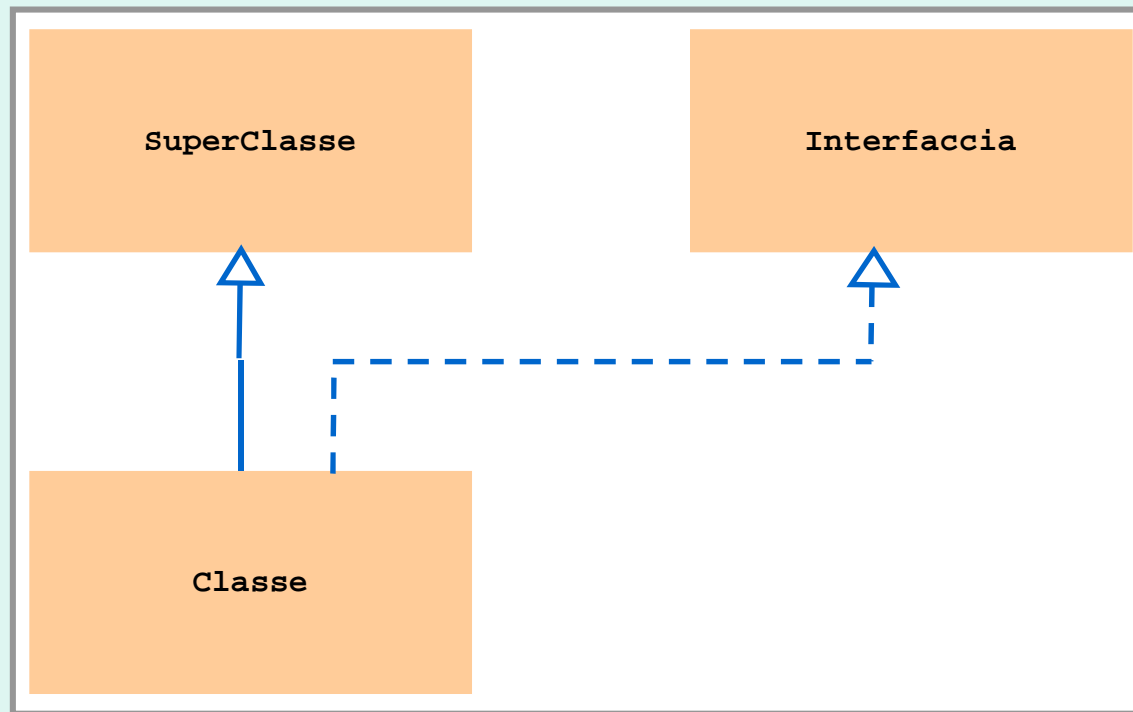
public class Staff extends UnivEmployee
{
    . . .
}
```

Gerarchie di tipi

- **Ogni classe può**
 - implementare più interfacce
 - estendere al più una classe
- **I due meccanismi si possono combinare**

Gerarchie di tipi

- Uno schema ricorrente



- Permette di utilizzare Classe come SuperClasse e Interfaccia

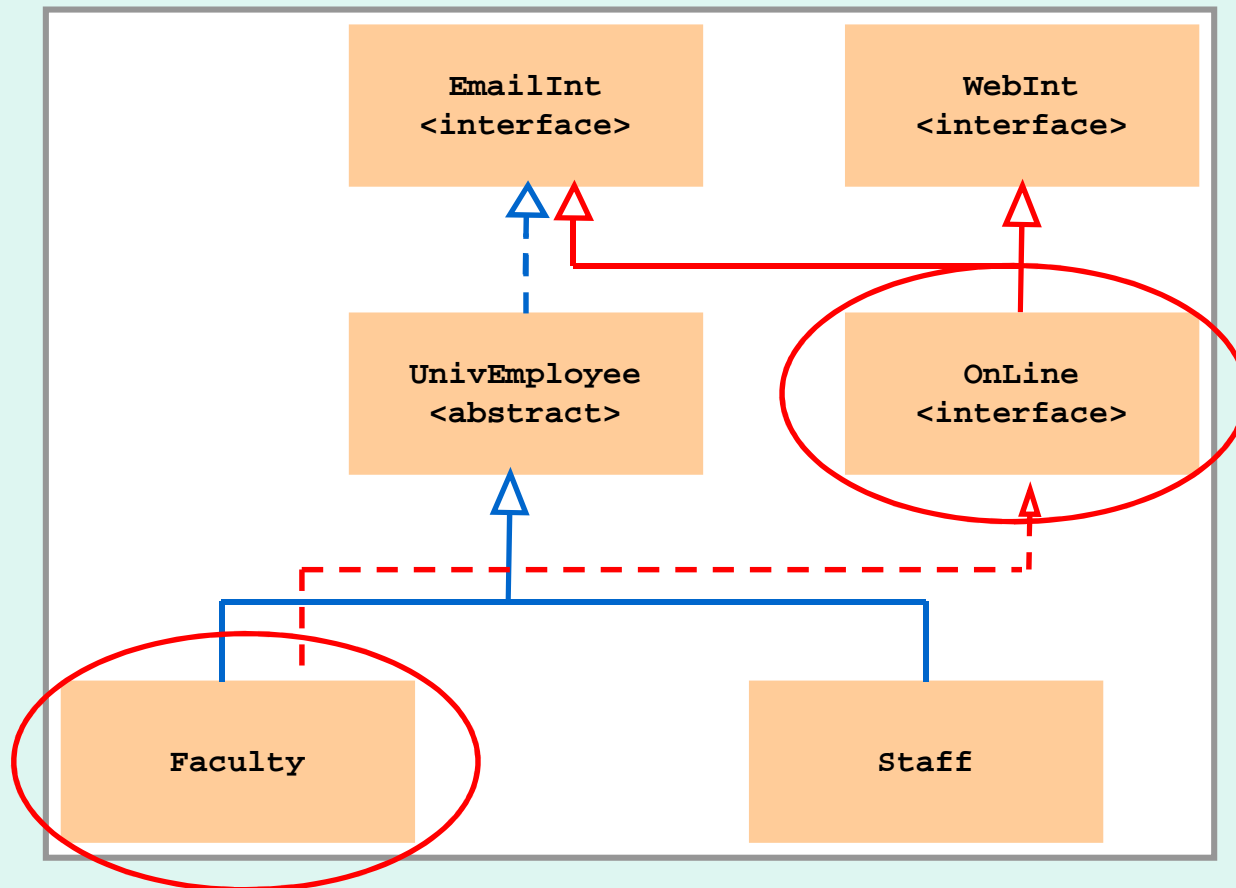
University Employees ... on the web

- **Possiamo dare struttura alla composizione di interfacce**

```
interface Online extends EmailInt,WebInt { }
```

- **Estensione di interfacce**
 - no vincoli: possiamo estendere più di una interfaccia
 - interfaccia che estende è sottotipo di tutte le interfacce che vengono estese

University Employees ... on-line



University Employees ... on line

```
public abstract class UnivEmployee implements EmailInt
{
    . . .
    public void mailTo(String msg) { . . . }
}

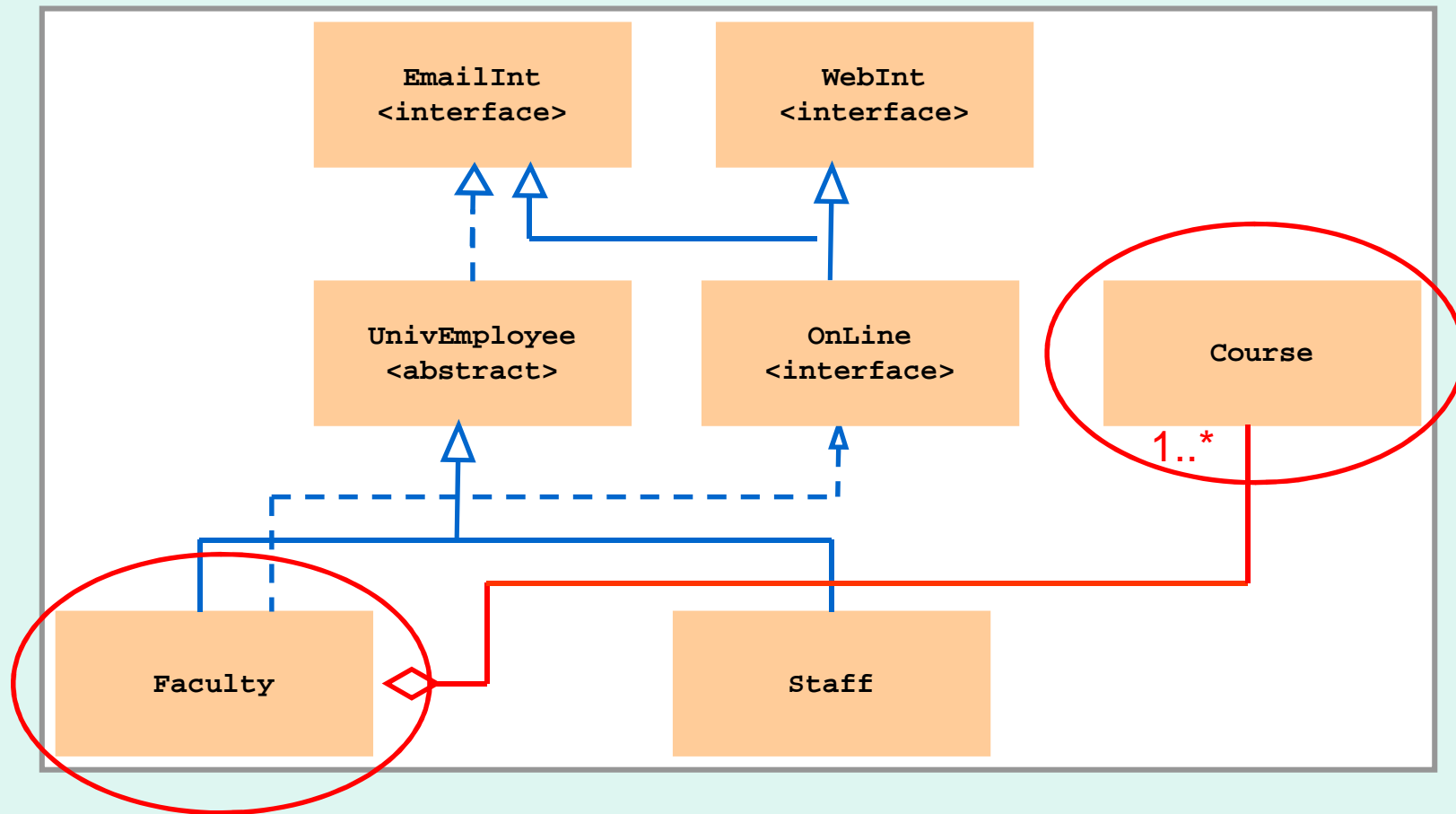
public class Faculty extends UnivEmployee implements OnLine
{
    public Url browse() { . . . }
    . . .
}

public class Staff extends UnivEmployee
{
    . . .
}
```

Corsi

- **Il personale docente insegna uno o più corsi all'interno dei corsi di laurea**
- **Rappresentiamo nei diagrammi con una nuova relazione tra classi**
 - associazione

University Employees



University Employees

```
public abstract class UnivEmployee implements EmailInt
{
    . . .
    public void mailTo(String msg) { . . . }
}

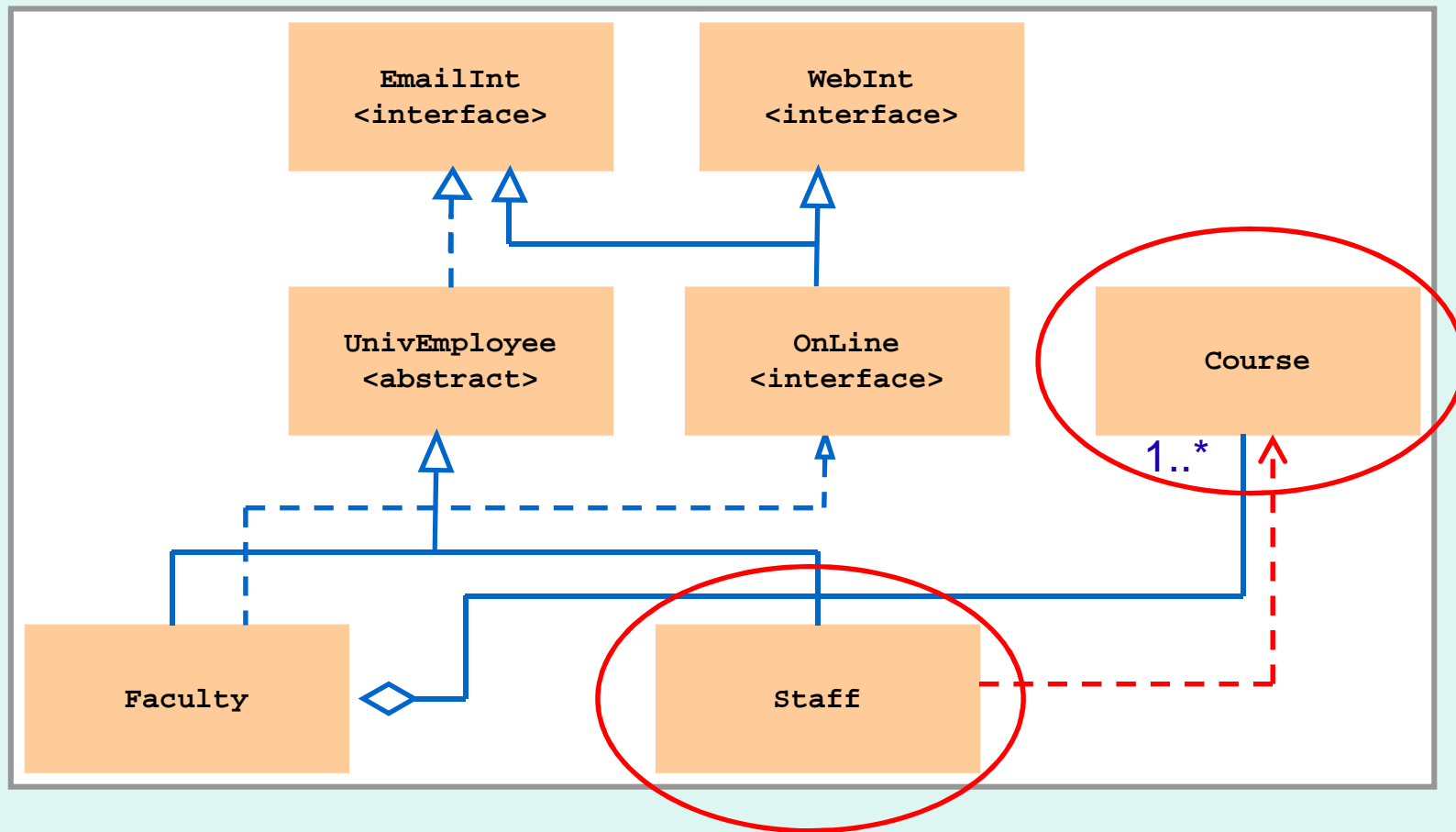
public class Faculty extends UnivEmployee implements OnLine
{
    public Url browse() { . . . }
    . . .
    private ArrayList<Course>  courses;
    . . .
}

public class Staff extends UnivEmployee
{
    . . .
}
```

University Employees

- I corsi vengono gestiti dal personale tecnico amministrativo per gli aspetti relativi a iscrizione, liste di esami ...
- La classe `Staff` dipende quindi dalla classe `Course`

University Employees



University Employees

```
public abstract class UnivEmployee implements EmailInt
{
    . . .
    public void mailTo(String msg) { . . . }
}

public class Faculty extends UnivEmployee implements Online
{
    public Url browse() { . . . }
    . . .
    private ArrayList<Course>  courses;
    . . .
}

public class Staff extends UnivEmployee
{
    public void admin(Course c) { . . . }
    . . .
}
```

Corsi ... on-line

- **Possiamo sfruttare la struttura in più che abbiamo ottenuto dall'introduzione delle interfacce per rappresentare ulteriori funzionalità**
 - Ad esempio, i corsi possono essere dotati di un sito web e di un indirizzo di posta elettronica

Corsi ... on-line

