OOP

Ereditarietà

LE TRE CARATTERISTICHE FONDAMENTALI DELLA OOP

Abbiamo già detto che **l'incapsulamento** è la prima.

Le altre due sono **l'ereditarietà** e il **polimorfismo**.

Qui ci occuperemo dell'**ereditarietà**.

EREDITARIETÀ

Punto		
-	int	x
-	int	y
+		Punto()
+		Punto(int x, int y)
+	int	getAscissa()
+	int	getOrdinata()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	String	toString()

Punto3d		
-	int	x
-	int	y
-	int	z
+		Punto3d()
+		Punto3d(int x, int y, int z)
+	int	getAscissa()
+	int	getOrdinata()
+	int	getQuota()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	void	setQuota(int z)
+	String	toString()

Supponiamo di avere la classe Punto..

..e di voler realizzare la classe Punto3d

In casi come questo possiamo usare il meccanismo dell'**ereditarietà** che ci permette di specificare solo le differenze rispetto a una classe già esistente.

La classe già esistente si chiama **superclasse** (o classe **padre**)

La nuova classe si chiama **sottoclasse** (o classe **figlio**)

QUALCOSA DA EREDITARE

Punto		
-	int	x
-	int	y
+		Punto()
+		Punto(int x, int y)
+	int	getAscissa()
+	int	getOrdinata()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	String	toString()

Punto3d			
-	int	x	←
-	int	y	←
-	int	z	
+		Punto3d()	
+		Punto3d(int x, int y, int z)	
+	int	getAscissa()	←
+	int	getOrdinata()	←
+	int	getQuota()	
+	void	setAscissa(int x)	←
+	void	setOrdinata(int y)	←
+	void	setQuota(int z)	
+	String	toString()	

La sottoclasse **eredita** tutti gli attributi e tutti i metodi (non privati) della superclasse.

QUALCOSA DA AGGIUNGERE

Punto		
-	int	x
-	int	y
+		Punto()
+		Punto(int x, int y)
+	int	getAscissa()
+	int	getOrdinata()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	String	toString()

Punto3d		
-	int	x
-	int	y
-	int	z
+		Punto3d()
+		Punto3d(int x, int y, int z)
+	int	getAscissa()
+	int	getOrdinata()
+	int	getQuota()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	void	setQuota(int z)
+	String	toString()

La sottoclasse **eredita**
tutti gli attributi e tutti i metodi
(non privati) della superclasse.

La sottoclasse **estende**
(se necessario) la superclasse con
nuovi attributi e/o metodi

QUALCOSA DA RIDEFINIRE

Punto		
-	int	x
-	int	y
+		Punto()
+		Punto(int x, int y)
+	int	getAscissa()
+	int	getOrdinata()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	String	toString()

Punto3d		
-	int	x
-	int	y
-	int	z
+		Punto3d()
+		Punto3d(int x, int y, int z)
+	int	getAscissa()
+	int	getOrdinata()
+	int	getQuota()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	void	setQuota(int z)
+	String	toString()

La sottoclasse **eredita**
tutti gli attributi e tutti i metodi
(non privati) della superclasse.

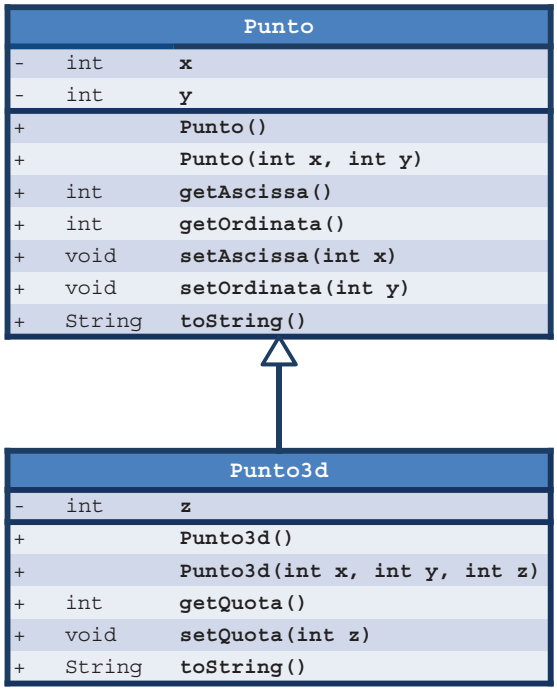
La sottoclasse **estende**
(se necessario) la superclasse con
nuovi attributi e/o metodi

La sottoclasse **ridefinisce**
(se necessario) gli attributi e/o i metodi
della superclasse che non sono più
«attendibili»

RAPPRESENTAZIONE UML

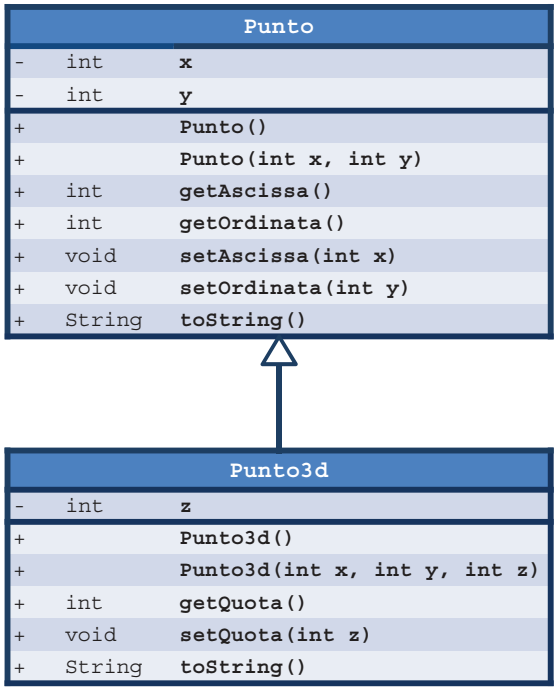
Punto		
-	int	x
-	int	y
+		Punto()
+		Punto(int x, int y)
+	int	getAscissa()
+	int	getOrdinata()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	String	toString()

Punto3d		
-	int	x
-	int	y
-	int	z
+		Punto3d()
+		Punto3d(int x, int y, int z)
+	int	getAscissa()
+	int	getOrdinata()
+	int	getQuota()
+	void	setAscissa(int x)
+	void	setOrdinata(int y)
+	void	setQuota(int z)
+	String	toString()



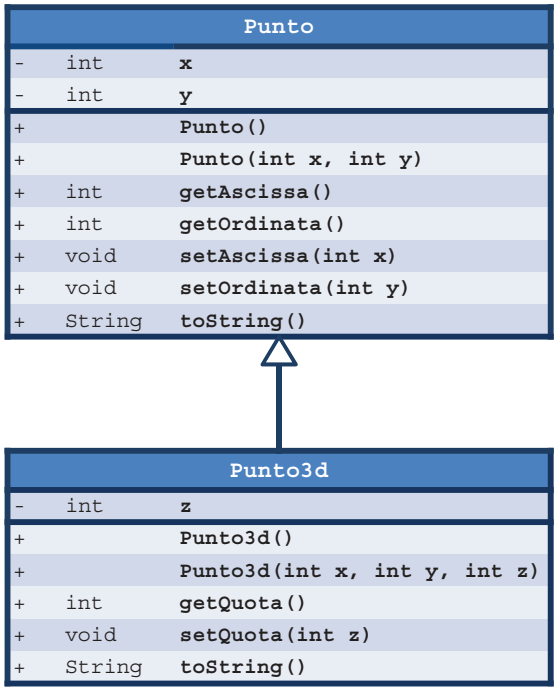
CODIFICA JAVA

```
public class Punto {



}
```



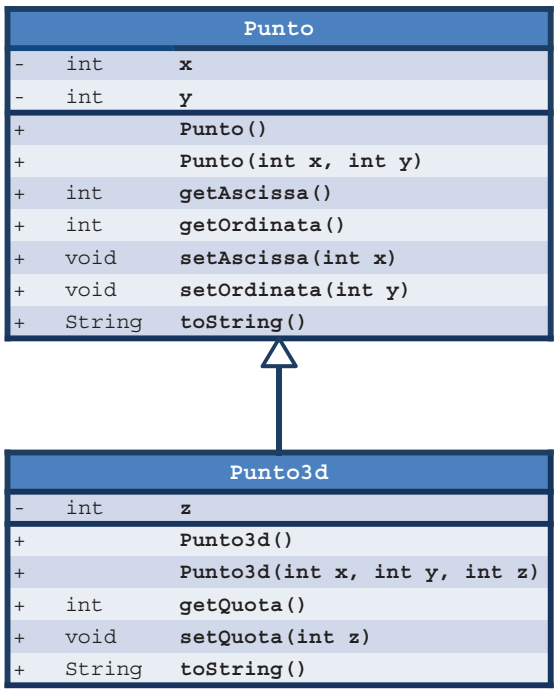
CODIFICA JAVA

```
public class Punto {
    private int x, y;









}
```



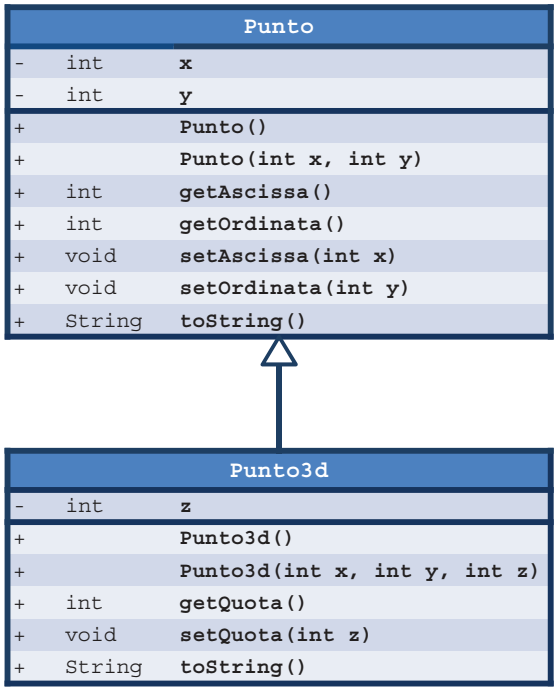
CODIFICA JAVA

```
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }









}
```



CODIFICA JAVA

```
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()         { return x;    }
    public int getOrdinata()        { return y;    }
    public void setAscissa(int x)   { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }


}
```



CODIFICA JAVA

```
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()         { return x;    }
    public int getOrdinata()        { return y;    }
    public void setAscissa(int x)   { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

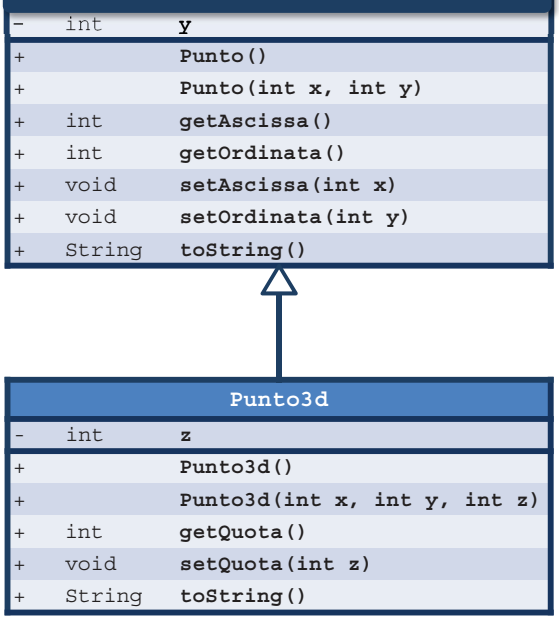
Qui di solito usiamo i metodi **get** per accedere agli attributi: in questo caso non lo abbiamo fatto per questioni di spazio sulla slide e soprattutto perché tutto sommato ci è consentito dallo specificatore **private**

CODIFICA JAVA

```java
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```java
public class Punto3d extends Punto {



}
```

Con **extends** indichiamo la superclasse dalla quale ereditiamo tutto ciò che non è privato.



CODIFICA JAVA

```java
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```java
public class Punto3d extends Punto {
    private int z;



}
```

Con **extends** indichiamo la superclasse dalla quale ereditiamo tutto ciò che non è privato.

CODIFICA JAVA

```java
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```java
public class Punto3d extends Punto {
    private int z;
    public Punto3d() {this(0,0,0);}




}
```

Con **extends** indichiamo la superclasse dalla quale ereditiamo tutto ciò che non è privato.

CODIFICA JAVA

```java
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```java
public class Punto3d extends Punto {
    private int z;
    public Punto3d() {this(0,0,0);}
    public Punto3d(int x, int y, int z) {
        super(x,y);
        setQuota(z);
    }




}
```

Con **extends** indichiamo la superclasse dalla quale ereditiamo tutto ciò che non è privato.

Per invocare il costruttore della superclasse usiamo il metodo speciale **super()**

CODIFICA JAVA

```java
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```java
public class Punto3d extends Punto {
    private int z;
    public Punto3d() {this(0,0,0);}
    public Punto3d(int x, int y, int z) {
        super(x,y);
        setQuota(z);
    }
    public int getQuota() {return z;}
    public void setQuota(int z) { this.z = z; }


   }
}
```

Con **extends** indichiamo la superclasse dalla quale ereditiamo tutto ciò che non è privato.

Per invocare il costruttore della superclasse usiamo il metodo speciale **super()**

ALESSANDRO URSOMANDO

CODIFICA JAVA

```java
public class Punto {
    private int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```java
public class Punto3d extends Punto {
    private int z;
    public Punto3d() {this(0,0,0);}
    public Punto3d(int x, int y, int z) {
        super(x,y);
        setQuota(z);
    }
    public int getQuota() {return z;}
    public void setQuota(int z) { this.z = z; }
    public String toString() {
        return "(" + getAscissa() + ";" +
               getOrdinata() + ";" + z + ")";
    }
}
```

Con **extends** indichiamo la superclasse dalla quale ereditiamo tutto ciò che non è privato.

Per invocare il costruttore della superclasse usiamo il metodo speciale **super()**

Qui per costruire la stringa da restituire ho bisogno dei valori di tutti gli attributi, ma poiché **x** e **y** sono attributi **privati** della classe Punto io non vi posso accedere: ecco perché uso i metodi **get**.

Per concedere alle nostre sottoclassi l'accesso diretto ai nostri metodi o attributi dobbiamo usare lo specificatore **protected**.

ALESSANDRO URSOMANDO

CODIFICA JAVA

```
public class Punto {
    private protected int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```
public class Punto3d extends Punto {
    private int z;
    public Punto3d() {this(0,0,0);}
    public Punto3d(int x, int y, int z) {
        super(x,y);
        setQuota(z);
    }
    public int getQuota() {return z;}
    public void setQuota(int z) { this.z = z; }
    public String toString() {
        return "(" + getAscissa() x + ";" +
               getOrdinata() y + ";" + z + ")";
    }
}
```

Fermo restando che è (ovviamente) lecito seguire la nostra abitudine di usare i metodi **get** con lo specificatore **protected** rendiamo possibile alle nostre sottoclassi di usare membri ereditati proprio come se fossero propri.

ISTANZIAZIONE DI SOTTOCLASSI

```
public class Punto {
    protected int x, y;
    public Punto() { this(0,0); }
    public Punto(int x, int y) {
        setAscissa(x);
        setOrdinata(y);
    }
    public int getAscissa()        { return x;   }
    public int getOrdinata()       { return y;   }
    public void setAscissa(int x)  { this.x = x; }
    public void setOrdinata(int y) { this.y = y; }
    public String toString() {
        return "(" + x + ";" + y + ")";
    }
}
```

```
public class Punto3d extends Punto {
    private int z;
    public Punto3d() {this(0,0,0);}
    public Punto3d(int x, int y, int z) {
        super(x,y);
        setQuota(z);
    }
    public int getQuota() {return z;}
    public void setQuota(int z) { this.z = z; }
    public String toString() {
        return "(" + x + ";" + y + ";" + z + ")";
    }

}
```

```
Punto3d p = new Punto3d();
p.setAscissa(13);
p.setOrdinata(41);
p.setQuota(9);
System.out.println(p);
```

```
Punto3d p = new Punto3d(5,78,1);
System.out.println(p);
```

PROTECTED IN UML

Punto	
# int	x
# int	y
+	Punto()
+	Punto(int x, int y)
+ int	getAscissa()
+ int	getOrdinata()
+ void	setAscissa(int x)
+ void	setOrdinata(int y)
+ String	toString()

Punto3d	
# int	z
+	Punto3d()
+	Punto3d(int x, int y, int z)
+ int	getQuota()
+ void	setQuota(int z)
+ String	toString()

In UML descriviamo
un attributo o un metodo
come **protected**
mediante il simbolo **#**

EREDITARIETÀ

Riassumendo,
l'ereditarietà è quel principio mediante il quale
posso definire una classe complessa
a partire da una classe più semplice

La classe figlio
eredita dalla classe padre tutto ciò che non è privato
estende la classe padre con quello che non c'era
ridefinisce ciò che nella classe padre è inadatto