[illegible]

Le Classi In Java
Una classe in Java si definisce tramite la parola chiave class.

```
public class MiaClasse {
//variabili
// metodi
}
```

All’interno del corpo della classe, ossia tra le parentesi graffe, si definiscono i vari metodi e variabili.
La regola fondamentale è che in ogni file .java deve avere al massimo una classe con la parola chiave public.
Per quanto riguarda le regole di naming (ossia come scegliere i nomi) delle variabili e metodi, tutti nomi di classi devono iniziare per lettera maiuscola (così come le lettere iniziali delle parole composte), mentre per le variabili e i metodi devono iniziare con la prima lettera minuscola e con le parole composte con le iniziali maiuscole.

4.3 I Metodi
In tutti i linguaggi di programmazione si può definire un insieme di istruzioni, identificate con un nome, in modo da poterle richiamare, tramite esso, in qualsiasi punto del programma. In questo modo otteniamo un codice riutilizzabile, senza doverlo riscrivere quelle istruzioni ogni volta. Questo insieme di istruzioni viene chiamato metodo.
La sintassi per definire un metodo è:

```
[ModificatoreDiVisibilità] tipoDiRitornoDelMetodo nomeDelMetodo([parametri])
{
corpoDelMetodo
}
```

- ModificatoriDiVisibilità: verranno discussi in maniera approfondita più avanti, per ora usiamo sempre la keyword public.
- tipoDiRitornoDelMetodo: indica il tipo di oggetto (un numero, una classe, etc) che il metodo restituisce, dopo aver eseguito le istruzioni del corpoDelMetodo. Nel caso in cui il metodo non ritorni nessun valore, si usa la parola chiave void, come tipo di ritorno.
- nomeDelMetodo: nome con cui verrà identificato il metodo.
- parametri: indica i dati che il metodo prenderà in ingresso. La sintassi è tipoVariabile nomeVariabile. I vari dati sono separati tra di loro tramite la virgola
- corpoDelMetodo: insieme di istruzioni che vengono eseguiti quando il metodo viene richiamato.

Esempio

```
public class Aritmetica {
public void hello()
{
System.out.println("ciaooo");
}


}
```

Esempio: metodo con un parametro in ingresso

```
public class Aritmetica {
public void hello(int s)
```

```
{
System.out.println("ciaooo");
}


}
```

Esempio: metodo con più parametri in ingresso

```
public class Aritmetica {
public void hello(int s, String g)
{
System.out.println("ciaooo");
}


}
```

Una classe può avere più metodi. Per poterli richiamare, abbiamo bisogno di una istanza della classe (chiamiamola x) e su di essa, tramite l'uso del punto, selezionare il metodo che ci interessa (esempio x.pippo() richiamerà il metodo pippo sull'istanza x).
Nel caso in cui un metodo richieda dei parametri di input, è necessario passare dei valori nello stesso numero, e nello stesso tipo, di quanti sono i parametri dichiarati nel metodo.

Esempio:

```
public class Aritmetica
{

public void hello()
{
System.out.println("ciaooo");
}
public int somma(int a, int b)
{
int r = a + b;
return r;
}

public static void main(String args[])
{
Aritmetica oggetto1 = new Aritmetica();
oggetto1.hello();
int risultato = oggetto1.somma(5, 6);
System.out.println("risultato:"+risultato);
}
}
```

La classe Aritmetica ha un metodo chiamato hello e un altro somma.
Hello stampa semplicemente "ciaoo" sulla console, ed essendo void non ritorna niente.
Somma prende in input due variabili di tipo int, di nome "a" e "b", li somma e ritorna il risultato.

Notare due aspetti importanti:

- la parola chiave return per far ritornare il risultato di un metodo.

- Il valore restituito da un metodo è memorizzato nella variabile intera risultato. Ciò opzionale, nel senso che si può richiamare il metodo senza “raccogliere” il valore che ritorna in una variabile. Es. oggetto1.somma(5, 6);
- Nel main, viene creata una istanza della classe e, su di essa, richiamiamo il metodo, passando in input i valori 5 e 6. Il risultato viene stampato sulla console tramite, sempre tramite l'istruzione System.out.println.

Esempio

```
public class Persona {

public String nome;
public String cognome;
public int eta;

public boolean verificaMaggiorenne(int limite) {
return eta >= limite;
}

public boolean verificaMaggiorenne() {
return eta >= 18;
}

public String calcolaCodiceFiscale() {
return nome + cognome + eta;
}

public static void main(String[] args) {
// primo oggetto
Persona p1 = new Persona();
p1.nome = "Pippo";
p1.cognome = "Rossi";
p1.eta = 14;

// secondo oggetto
Persona p2 = new Persona();
p2.nome = "Vasco";
p2.cognome = "Rossi";
p2.eta = 65;

if (p1.verificaMaggiorenne())
System.out.println(p1.nome + " " + p1.cognome + " è maggiorenne");
else
System.out.println(p1.nome + " " + p1.cognome + " è minorenne");

if (p2.verificaMaggiorenne())
System.out.println(p2.nome + " " + p2.cognome + " è maggiorenne");
else
System.out.println(p2.nome + " " + p2.cognome + " è minorenne");

if (p1.verificaMaggiorenne(30))
System.out.println(p1.nome + " " + p1.cognome + " ha più di 30 anni");
else
System.out.println(p1.nome + " " + p1.cognome + " ha meno di 30 anni");

String cf1 = p1.calcolaCodiceFiscale();
System.out.println("il codice fiscale di p1 è:" + cf1);

String cf2 = p2.calcolaCodiceFiscale();
System.out.println("il codice fiscale di p2 è:" + cf2);
```

```
}

}
```

4.5 I Costruttori

Il costruttore di una classe è un metodo speciale utilizzato per creare una istanza della classe (ripetiamo, istanza di una classe P = una variabile del tipo classe P).

Per definirlo, è sufficiente creare un metodo con lo stesso nome della classe.

Ipotizzando di avere una classe chiamata MyClass, creiamo un costruttore che prende in input un intero:

Esempio

```
public class Prova {

Prova(int l){

}


}
```

Il costruttore viene invocato, in automatico, durante la creazione di una istanza, tramite la keyword new:

```
public static void main(String[] args) {

Prova p = new Prova(2);

}
```

Una classe può avere più costruttori, che si differenziano tra loro per il numero, e il tipo, di parametri in ingresso. Durante la creazione di una istanza, in base ai parametri passati a new, verrà selezionato il costruttore corrispondente.

Esempio:

```
public class Prova {

Prova(int l){

}

Prova(int h, String g){

}

public static void main(String[] args) {

Prova p = new Prova(2);//invocato costruttore con un parametro

Prova p1 = new Prova(23, "ciao");//invocato costruttore con due parametri

}

}
```

I costruttori sono utilizzati per inizializzare i campi della classe.

Esempio

```
public class Prova {
int count;
String nome;

Prova(int l){
count = l;
}

Prova(int h, String g){
nome = g;
count = h;
}

public static void main(String[] args) {

Prova p = new Prova(2);

Prova p1 = new Prova(23, "ciao");

}

}
```

Un costruttore che non prende nessun parametro in input è detto costruttore di default.

Ipotizziamo di avere una classe in cui non è definito nessun costruttore

Esempio

```
public class Prova {


public static void main(String[] args) {

Prova p = new Prova();

}
```

```
}
```

Il programma compila senza nessun problema.
Definiamo ora un costruttore che prenda un intero in ingresso

Esempio

```
public class Prova {

Prova(int h){

}

public static void main(String[] args) {

Prova p = new Prova();//errore di compilazione

}

}
```

Appare, ora, un errore di compilazione alla riga in cui si crea una istanza.

Modifichiamo il codice, aggiungendo un costruttore con nessun parametro in input

Esempio

```
public class Prova {

Prova(int h){

}

Prova(){

}
public static void main(String[] args) {

Prova p = new Prova();

}

}
```

Ora, l'errore di compilazione è scomparso.
Questo perché, quando una classe non ha definito nessun costruttore, java, dietro le quinte, ne crea uno in automatico. Questo costruttore nascosto non ha nessun parametro in ingresso.

Nel momento in cui noi definiamo un nostro costruttore, questa creazione di un costruttore nascosto non avviene più.
Per questo motivo il costruttore senza argomenti è detto di default, perché è il costruttore che una classe ha di default.

Esempio

```
public class Persona {

public String nome;
public String cognome;
```

```
public int eta;

public Persona(String nome, String cognome, int eta) {
this.nome = nome;
this.cognome = cognome;
this.eta = eta;
}

public Persona(String nome, String cognome) {
this.nome = nome;
this.cognome = cognome;
}



public static void main(String[] args) {
// primo oggetto
Persona p1 = new Persona("Pippo", "Rossi", 14);

// secondo oggetto
Persona p2 = new Persona("Vasco", "Rossi");

System.out.println(p1.nome + " " + p1.cognome + " " + p1.eta);

System.out.println(p2.nome + " " + p2.cognome + " " + p2.eta);




}

}
```