1. Di regola un metodo restituisce un solo valore di ritorno il cui tipo è definito nella dichiarazione del metodo stesso. Tuttavia, ci sono vie alternative attraverso le quali è possibile far restituire a un metodo più di un valore. La prima modalità consiste nell'uso della parola chiave *out* che identifica il parametro che verrà restituito in aggiunta all'usuale parametro di ritorno del metodo. La seconda modalità consiste, invece, nel far ritornare al metodo una tupla di valori. La terza e ultima modalità consiste, infine, nella definizione di una classe (o una struttura) che contiene al suo interno tutte le informazioni che si vuole che il metodo ritorni.
2. Con il termine *cast* si sta ad indicare un'operazione di conversione tra tipi. Quando avviene una simile conversione si possono verificare due casi. Un caso (la cosiddetta *conversione implicita*) in cui non c'è perdita di informazione (in quanto, ad esempio, la conversione avviene da un tipo "più piccolo" a un tipo "più grande" – si pensi alla conversione da un tipo derivato a un tipo base). Il secondo caso, invece, prevede una perdita di informazione: in tal caso, sarà necessario fare un cast esplicito (pertanto si parla di *conversione esplicita*). Esempio di casting implicito:
```
int a = 10;
double b = a;
```
. Esempio di casting esplicito:
```
double a = 1.0;
int b = (int)a;
```
(la perdita di informazione qui è data dal fatto che 1.0 sarà rappresentato come 1 nella variabile intera b).
3. La parola chiave *static* può identificare classi o membri di una classe. Una classe statica è una classe che non può essere istanziata (è come se la classe esistesse in quanto unità inscindibile). Un esempio di classe statica è la classe *Math* che fornisce un insieme di funzioni utili da usare in operazioni matematiche. Variabili e metodi statici, invece, sono membri che appartengono, come si può intuire dalla definizione di "classe statica", alla classe e non ai singoli oggetti. Pertanto, per recuperare le variabili o invocare dei metodi, essi devono essere richiamati sulla classe.
4. L'evento è un particolare membro di una classe identificabile (e definibile) per mezzo della parola chiave *event*. L'evento, inoltre, altro non è che il delegato di un tipo di delegate. Un tipo predefinito di delegate (che è spesso utilizzato) è il tipo EventHandler. Di solito, infatti, quando si deve implementare un evento, lo si dichiara come membro di una classe e di tipo EventHandler, come qui di seguito: `public event EventHandler ProcessoIniziato;` . A questo punto, bisognerà lanciare l'evento, ad esempio all'interno di un metodo di istanza della classe: se così è, l'istanza della classe sarà il cosiddetto *publisher* dell'evento. Es. di lancio di un evento `ProcessoIniziato(this, EventArgs.Empty);` . Dopodiché si dovrà andare ad aggiornare la lista dei *subscriber* all'evento (ovvero quegli oggetti che ricevono la notifica dell'avvenuto evento e che, eventualmente, lo gestiscono). Per aggiornare la lista dei subscriber si useranno gli operatori += e -= (rispettivamente per aggiungere o rimuovere un subscriber). Infine, si dovranno implementare i metodi che, ricevendo la notifica, faranno qualcosa in risposta ad essa. Tali metodi, dovranno, ovviamente, rispettare la firma del delegato (dell'evento) a cui vengono associati.
5. I Generic sono classi, metodi, interfacce, collezioni che usano un tipo generico T. L'uso di un Generic offre numerosi vantaggi al programmatore, tra cui la possibilità di riusare il codice e una maggiore controllo sui tipi. Difatti, nel momento in cui un Generic viene usato, un tipo viene definito e il compilatore effettuerà tutta una serie di controlli sul tipo segnalando al programmatore eventuali situazioni di errore.