INTRODUZIONE

Un algoritmo è la sequenza di passi non ambigui che definiscono il modo in cui viene eseguita un'operazione che deve essere eseguita dell'elaboratore e che deve portare a terminazione, cioè non infinita, come per esempio le ricette di cucina, indicazioni stradali, gli spartiti, ecc. Lo sviluppo di un algoritmo viene fatto inizialmente tramite lo pseudocodice, ossia un linguaggio informale per descrivere le azioni che esso deve compiere. La rappresentazione di un algoritmo è detta programma. Sviluppare un programma, codificarlo in un formato e inserirlo in una macchina è definito programmazione. I programmi e gli algoritmi sono chiamati globalmente software, mentre i dispositivi fisici hardware. Perciò l'intelligenza necessaria per risolvere il problema è codificata nell'algoritmo e soltanto se troviamo un algoritmo che formalizza l'esecuzione di un'operazione possiamo creare una macchina per automatizzarla. Però è stato definito un teorema di incompletezza, cioè sui limiti degli algoritmi, nel 1930, il quale afferma che in qualsiasi teoria matematica che applichi il sistema aritmetico tradizionale ci sono affermazioni di cui non può essere stabilita la verità o falsità attraverso un algoritmo.

Un computer è una macchina automatizzata in grado di eseguire complessi calcoli matematici ed eventualmente altri tipi di elaborazioni dati. In particolare il computer nasce esclusivamente come macchina in grado di eseguire calcoli matematici e, solo a partire dalla seconda metà del XX secolo, evolve in macchina in grado di eseguire le elaborazioni dati più varie. Esso è composto da parti hardware e software. La flessibilità dell'hardware è definita come la possibilità di fare più di una azione, perciò per esempio il lettore CD è meno flessibile di un computer perché può fare solo una azione, cioè leggere appunto un CD, mentre il computer ha le istruzioni per eseguire i vari programmi.

Le parti di un computer sono:

- CPU: controlla l'esecuzione dei programmi ed elabora i dati, quindi esegue le istruzioni di un programma e reperisce i dati dalla memoria o dai dispositivi periferici per memorizzarne i risultati.
- Memoria: si distingue in Primaria (RAM) e Secondaria (HARD DISK: contiene le istruzioni dei programmi e i rispettivi dati).
- Periferiche I/O: interagiscono con l'utente (altoparlanti, schermo, tastiera, ...).
- Networks: per interagire con altri dispositivi nella rete.

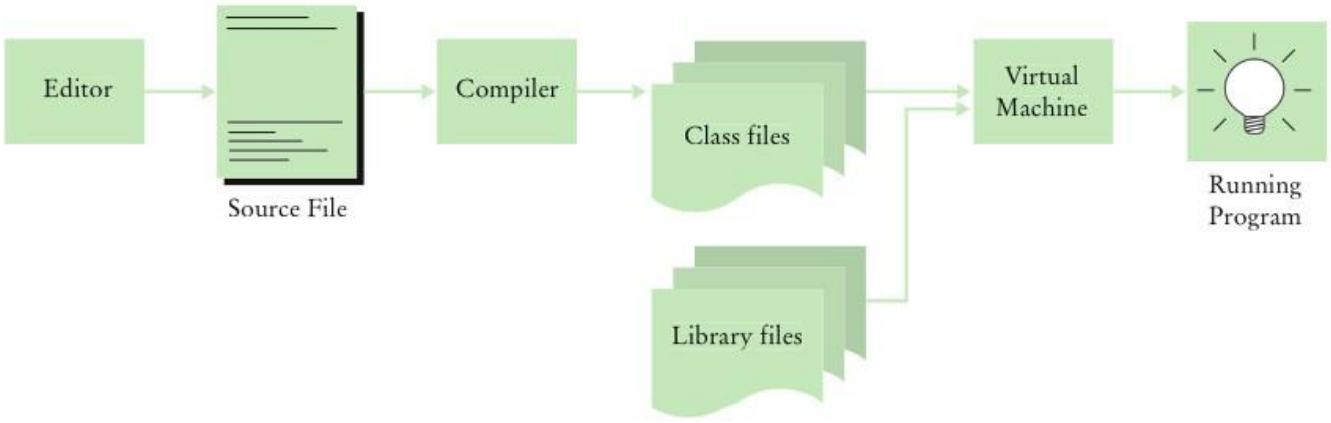
PROGRAMMAZIONE JAVA

Inizialmente fu pensata per programmare gli elettrodomestici, ma dopo aver passato diversi anni per progettare il linguaggio di programmazione non furono riusciti a vendere nessun elettrodomestico. Dopo alcuni anni utilizzarono Java per la programmazione delle pagine web e per gli internet browser, perché il linguaggio è:

- Indipendente dalla piattaforma utilizzata, infatti può essere letto da qualunque computer senza tener conto del sistema operativo;
- Sicuro, perché non compromette l'integrità dei dati del computer;
- Ci sono molte librerie da poter utilizzare, quindi è possibile una vasta opportunità di programmazione;
- Progettato per internet.

L'editor è un programma per creare e modificare un testo, come il codice di un programma Java. La programmazione è case sensitive, quindi lettere maiuscole e minuscole sono cose distinte. Importante è dare un layout di facile lettura nel codice del programma usando l'indentazione delle varie righe. Quindi il compilatore di Java traduce il codice sorgente nelle class files, cioè classi che contengono istruzioni per la macchina virtuale di java. Per la compilazione bisogna aprire il cmd del computer e utilizzare il comando "cd NomeCartella" (in cui è contenuto il file .java) e successivamente utilizzare questi comandi:

- Per compilare: javac NomeFile.java
- Per eseguire: java NomeFile



PROGRAMMA: HELLO PRINTER.JAVA

```
1 public class HelloPrinter
2  {
3     public static void main(String[] args)
4           {
5              // Display a greeting in the console window
6
7              System.out.println("Hello, World!");
8           }
9  }
```

public static void main(String[] args) ➔ Dichiarazione metodo main
System.out.println("Hello, World!"); ➔ Evocazione metodo
Println ➔ è il comando per stampare il messaggio al computer ed “ln” fa andare a capo.
Print ➔ non fa andare a capo
(“Hello World!”) ➔ è una stringa

Le classi sono dei blocchi fondamentali per i programmi Java. Per dichiarare una classe pubblica, cioè che può essere vista da tutti, va fatta in questo modo: public class NomeFile. Non ci possono essere più di una classe pubblica in un programma, in quanto dà il nome al file. Ogni classe contiene le dichiarazioni dei metodi e ogni metodo contiene una sequenza di istruzioni che il programma deve seguire per svolgere i processi. Ogni metodo per funzionare deve essere invocato e devono essere applicate le sue istruzioni per funzionare correttamente. Main è un metodo che deve essere contenuto in ogni programma e la dichiarazione si fa:

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

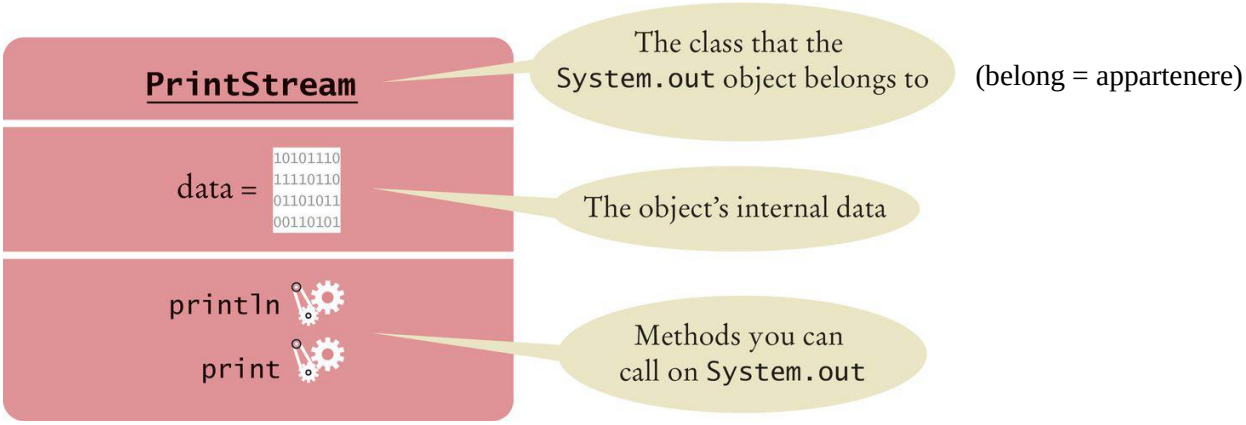
Il corpo del metodo main è composto dagli statements, ossia le istruzioni del programma [System.out.println("Hello, World!");]. Gli enunciati presenti nel metodo main vengono eseguiti uno dopo l’altro in ordine. Dopo l’evocazione del metodo bisogna specificare l’argomento ossia (“Hello World!”) che può essere una stringa se contenuto in “…”, oppure intero (3+7) e quindi l’output stampato a schermo sarà 10. Importante: dopo ogni argomento dei metodi bisogna mettere il ‘;’.

I tipi di errori che si possono verificare sono:

- Sintassi: è la violazione delle regole del linguaggio di programmazione, (esempio: System.ou.println("Hello, World!");

- Logici: causano l'esecuzione di azioni che il programmatore non intendeva fare, quindi il programma non dà il risultato atteso. Un tipo di errore logico è l'Exception causato per esempio da una divisione per 0 (1/0).

In Java la programmazione è ad oggetti, cioè un'entità che possiamo manipolare evocando i suoi metodi, in quanto ogni oggetto ha dei comportamenti certi e quindi possiamo manipolare tale oggetto per avere degli effetti certi. Una classe descrive un insieme di oggetti con comportamenti simili, perciò ogni oggetto appartiene ad una classe e può essere manipolato solo dai metodi della classe a cui appartiene.



(belong = appartenere)

Una variabile è una memoria di un valore che potrà essere utilizzata in un secondo momento. Nella dichiarazione della variabile dobbiamo specificare il nome, il valore iniziale e che tipo di valori essa può contenere. La dichiarazione: int width = 20; dove width è il nome, int è il tipo e 20 è il suo valore iniziale. I tipi di variabile sono: int ➔ interi, double ➔ razionali (numeri con la virgola), string ➔ stringa. Le convenzioni consigliano di dare il nome della variabile con lettera minuscola, le classi con lettera maiuscola, dare commetti per le azioni che vengono compiute dal programma con l'uso delle // o /* all'inizio e */ alla fine.

Per le operazioni di riassegnamento non serve dichiarare il tipo della variabile. Ovviamente può prendere anche questo: width = width + 10; dove prima fa la somma del valore e poi vengono memorizzati nella variabile.

Come avevamo detto, per manipolare un oggetto bisogna usare i metodi della sua classe. Oltre al system.out, che appartiene alla classe PrintStream e può essere modificato con i metodi println e print, esiste lo string, ossia una classe dove l'oggetto può essere una variabile chiamata variabile oggetto, che usa il metodo lenght. Esempio:

String greating = "Hello World!"; ➔ dichiarazione variabile.
int numberOfCharacters = greating.lenght(); ➔ greating= variabile oggetto; length= metodo; ()= argomento.

Un altro esempio dove viene applicato un metodo ad un oggetto ma che esso non viene modificato, ma viene modificata la variabile su cui verrà savato il valore è:

String river="Missisipi";
String bigRiver=river.toUpperCase();➔bigRiver= nuova variabile; river= variabile oggetto; toUpperCase= metodo

La variabile river avrà come valore "Missisipi", mentre bigRiver avrà "MISSISIPI". Atri metodi sono:

System.out.println(river.length()); ➔ restituisce a schermo la lunghezza del valore nella variabile river.

river= river.replace("issip","our"); ➔ il valore nella variabile river viene cambiato in "Missouri".

System.out.println(river.replace("issip","our")); ➔ non viene salvato il valore nella variabile ma il risultato viene stampato a schermo.

In Java c'è anche la possibilità di invocare due metodi contemporaneamente, come da esempio: greeting.replace("World","Dave").length();

Nei programmi per usare un oggetto, bisogna specificarne le proprietà iniziali costruendolo. Utilizzando la classe *Rectangle*, ossia una classe che descrive oggetti rettangolari. Creando un nuovo oggetto quindi non viene creato un rettangolo, ma un oggetto che descrive il rettangolo tramite le coordinate x,y del vertice in alto a sinistra, della sua larghezza (width) e della sua altezza (height). La riga di codice per la creazione è la seguente:

new Rectangle(5,10,20,30)

Per poterlo utilizzare in seguito, va salvato i suoi valori in una variabile attraverso questo codice:

Rectangle box= new Rectangle(5,10,20,30);

[x=5, y=10, width=20, height=30]

Dove Rectangle tipo della variabile; box è il nome della variabile; new operatore che consente la creazione; Rectangle è l'oggetto; quello scritto tra le parentesi è l'argomento e in questo caso il parametro di costruzione.

Classificazione dei metodi:

- Metodi d'accesso: sono dei metodi che accede ad un oggetto e restituisce alcune informazioni ad esso relative senza modificare l'oggetto;
- Metodi modificatori: sono i metodi che hanno lo scopo di modificare i dati interni di un oggetto.

Un metodo che restituisce i valori di un parametro del rettangolo è con l'istruzione: nomeVariabile.getParametro();

esempio: System.out.println("Valore della x " + box.getX());

Un metodo che modifica i valori della x e y e quindi facendo spostare il rettangolo è attraverso l'istruzione: nomeVariabile.translate(NuovoValoreX,NuovoValoreY);

esempio: box.translate(25,40);

API DOCUMENTATION

API= Application Programming Interface. Le classi e i metodi della libreria Java, sono elencati nella documentazione api. I programmatori di applicazioni utilizzano le classi già esistenti per realizzare i programmi, mentre i programmatori che hanno realizzato le classi della libreria sono detti programmatori di sistema. Le classi della libreria sono organizzate in pacchetti, ciascuno dei quali è una raccolta di classi aventi funzioni tra loro correlate. Per esempio la classe *Rectangle* appartiene al pacchetto *java.awt*, che contiene molte classi utili per disegnare finestre e forme grafiche. Per poter utilizzare il pacchetto bisogna importarlo inserendo questa riga di codice: import java.awt.Rectangle; Comunque le classi principali non c'è bisogno che vengano importate in modo esplicito, perché vengono importate automaticamente.

Per testare la corretta realizzazione di uno o più metodi, viene utilizzato un programma collaudo. Le fasi per collaudare un metodo della classe Rectangle sono:

1. Definire una classe collaudo (tester);
2. Definire in essa il metodo main;
3. Costruire uno o più oggetti all'interno del metodo main;
4. Applicare il metodo agli oggetti;
5. Visualizzare i risultati della invocazione dei metodi;
6. Visualizzare i valori previsti da tali invocazioni.

Esempio:

```
import java.awt.Rectangle;
public class MoveTester
{
        public static void main(String[] args)
        {
         Rectangle box= new Rectangle(5,10,20,30);
         System.out.println("Valore X "+ box.getX());
         System.out.println("Valore Y "+ box.getY());
         box.translate(15,25);
         System.out.println("Nuovo valore X "+ box.getX());
         System.out.println("Nuovo valore Y "+ box.getY());
        }
}
```

Una variabile oggetto, cioè una variabile il cui tipo dichiarato sia una classe, non contiene l'oggetto, perché memorizza al proprio interno soltanto la posizione dell'oggetto all'interno della memoria. Questo perché gli oggetti possono occupare anche grandi dimensioni, quindi è più efficiente memorizzare soltanto La posizione dell'oggetto. Perciò l'operatore *new* costruisce un oggetto di tipo *Rectangle* a cui fa riferimento la variabile box, cioè box non contiene l'oggetto ma contiene il riferimento dove l'oggetto viene memorizzato. Da notare che i numeri non sono oggetti, quindi una variabile di tipo intero contiene veramente un numero. In oltre quando si pone due variabile intere una uguale all'altra, i due valori sono indipendenti, mentre se si copia un riferimento ad un oggetto, l'originale e la copia fanno riferimento al medesimo oggetto. Esempio:

```
int numero= 13:
int numero2= numero;
numero2= 12;
```

➔ la variabile numero avrà come valore 13, mentre numero2 avrà come valore 12.

```
Rectangle box= new Rectangle(5,10,20,30);
Rectangle box2= box;
box2.translate(15,25);
```

➔ in questo caso, visto che le due variabili fanno riferimento allo stesso rettangolo, dopo il metodo entrambe si riferiscono al rettangolo spostato.

REALIZZAZIONE DI CLASSI

Per progettare una classe bisogna decidere quale sia la sua interfaccia pubblica, ossia quali siano i metodi che i programmatori potranno utilizzare per manipolare oggetti di quella classe e successivamente realizzare tali metodi. Un primo esempio è la realizzazione di un conta persone. Ogni volta che una persona entra il contatore verrà incrementato di uno, quindi c'è la necessità di usare una variabile che tenga traccia del numero di avanzamenti. Un oggetto memorizza i propri dati all'interno della variabile istanza, cioè una zono di memorizzazione presente in ciascun oggetto della classe. I metodi creati, in questo caso quello che incrementa il valore della variabile, utilizza la variabile istanza appartenente all'oggetto che viene manipolato. La variabile istanza viene dichiarata come una variabile privata, cioè che non vengo visualizzate nella classe pubblica e che quindi la modalità di accesso è privata. Il processo per rendere nascoste le informazioni che stanno dietro al funzionamento delle diverse classi, è detto incapsulamento. È il procedimento per nascondere i dettagli realizzativi rendendo pubblica l'interfaccia. In Java è il costrutto sintattico class che lo realizza, mentre i metodi pubblici di una classe sono l'interfaccia attraverso cui viene manipolata l'implementazione privata.

Per la realizzazione dei metodi di una classe, bisogna distinguere i metodi in metodi modificatori, cioè che modificano i valori di determinate variabili e quindi fanno le istruzioni vere e proprie del programma, e i metodi di accesso, cioè quei metodi che restituiscono un valore che può essere memorizzato in una variabile.

I costruttori vengono utilizzati per inizializzare gli oggetti e devono avere il nome uguale a quello della classe e non definiscono un tipo per il valore restituito (non richiedono il metodo void). A seconda del programma i costruttori richiedono un argomento in entrata.

Nella realizzazione delle classi bisogna per prima cosa definire le variabili istanza, che di solito sono variabili provate, che conterranno i dati contenuti negli oggetti del programma, successivamente creare i costruttori che conterra il comando di assegnazione del valore iniziale delle variabili istanza, per poi definire i vari metodi modificatori o d'accesso.

```
public class Cellulare
{
	private double carica=0;
	private int nchiamate=0;
	private static final double COSTO= 0.20;
	public Cellulare(double unaCarica) {
		carica=unaCarica;		}
	public void ricarica(double unaRicarica){
		carica= carica+ unaRicarica; }
	public void chiamata(double MinutiDurata){
		MinutiDurata= MinutiDurata*COSTO;
		if (carica>MinutiDurata)
		{
			carica= carica-MinutiDurata;
			nchiamate= nchiamate+1;
		}
		else
		{
			System.out.println("Credito Insufficente per effettuare la chiamata");
		}
					}
	public double numero404(){
		return carica;	}
	public int getNumeroChiamate(){
		return nchiamate;	}
	public void azzeraChiamate(){
		nchiamate=0;	}
}
```

TIPI DI DATI FONDAMENTALI

In Java ogni valore è un riferimento ad un oggetto oppure appartiene ad uno degli otto tipi primitivi.

I tipi primitivi

Nome	Tipo di valore	Memoria usata	Range di valori
byte	intero	8 bit = 1 byte	-128 : +127
short		16 bit = 2 byte	-32.768 : +32.767
int		32 bit = 4 byte	-2.147.483.648 : +2.147.483.647
long		64 bit = 8 byte	-9.223.372.036.854.775.808 : +9.223.374.036.854.775.808
float	floating point	32 bit = 4 byte	+/- 3,4028… x 10+38 : +/- 1,4023… x 0-45
double		64 bit = 8 byte	+/- 1,767… x 10+308 : +/- 4,940… x 0-324
char	singolo carattere	16 bit = 2 byte	Tutti i caratteri Unicode
boolean	true o false	1 bit	*true* o *false*

I calcoli che coinvolgono gli interi possono dare vita ad un overflow, cioè quando il risultato di un calcolo non rientra nell'intervallo di variabilità del tipo numerico, perciò per risolvere questo problema (che non viene segnalato del computer) viene utilizzato il tipo long. Per quanto riguarda il tipo double, se un valore non può essere rappresentato in modo esatto, viene arrotondato al valore più vicino.

Molto programmi usano valori *costanti*, ossia valori che non vengono modificati e che hanno un significato ben preciso all'interno dell'elaborazione. Per dichiarare una costante bisogna utilizzare la parola riservata final. Esempio: final double PENNY_VALUE = 0.01;

Le operazioni aritmetiche basilari vengono eseguite anche in Java. Combinazioni di variabili, numeri letterali, operatori e invocazione di metodi viene chiamata espressione. Se tale espressione contiene alcuni valori interi e altri in virgola mobile il risultato sarà in virgola mobile, mentre se viene effettuata una divisione tra interi, allora il risultato sarà un intero. Per avere come risultato il resto di una divisione bisogna utilizzare l'operatore "%".
Per funzioni meno elementari (elevamento a potenza, estrazione di radice,…) è necessario invocare metodi matematici (vedi pagina 153).
Per arrotondare un double per assegnarlo ad una variabile intera, si usa l'operatore di cast, che rimuove la parte in virgola mobile del valore double. Esempio: double balance = 123.43; int dollars = (int) balance; avrà valore 123.

Per acquistare dati in entrata da testiera, viene utilizzato il pacchetto java.util.Scanner per poi creare un oggetto di tipo Scanner.

Esempio:

```
import java.util.Scanner
public class Prova
{
   public static void main*String[] args(
  {
     Scanner in = new Scanner(System in);
     double prezzo = in.nextDouble();   // prende in ingresso dati di tipo double e gli salva nella variabile prezzo
     int volume = in.nextInt(); // prende in ingresso dati di tipo int e gli salva nella variabile volume
     string descrizione = in.nextLine();  // prende in ingresso dati di tipo stringa e gli salva nella variabile descrizione
     string str = in.next(); /* prende in ingresso dati di tipo stringa fino al primo spazio trovato e gli salva nella variabile str esempio hello world viene reso solo la stringa hello*/
  }
}
```

Un modo per controllare se i valori inseriti da tastiera siano corretti avviene mediante l'utilizzo del metodo in.nextInt(); (in caso di interi) che memorizza il valore nella variabile solo se il dato inserito da tastiera è valido.
Un modo per stampare il risultato delle espressioni del programma con solo un numero di cifre decimali specificato, avviene con il metodo System.out.printf("%.3f",variabile); dove verrà visualizzato la variabile con soltanto due cifre decimali dopo la virgola. Comunque, il medoto prinf, non va a capo al termine della visualizzazione prodotta.

Oltre a questi tipi di dato, ci possono essere anche variabili di tipo stringa, ossia sequenze di caratteri racchiusi tra virgolette. Un metodo molto comune che viene applicato alle stringhe è quello che calcola la lunghezza di essa, cioè variabile.length(); . Per unire più stringhe, ossia per concatenarle, viene usato il simbolo + dell'addizione.
Un modo per iserire le virgolette all'interno di una stringa letterale si utilizza il \"Stringa\", questa sequenza viene chiamata sequenza di escape. Per inserire il backslash all'interno di una stringa viene utilizzato il \\ .
Per andare a capo in una stringa viene utilizzato il \n.
Per assegnare un valore ad un carattere (char) bisogna utilizzare solamente il 'carattere'. Esempio char risp = 'V';
Altro metodo importante è quello che restituisce il carattere di una stringa ad una determinata posizione di essa. La prima posizione di una stringa, quindi la prima lettera, è associato l'indice 0. Il metodo utilizzato è:
char start = name.charAt(0);
Data una stringa è possibile estrarne sottostringhe usando il metodo substring che restituisce una stringa costituita dai caratteri consecutivi della stringa iniziale, che partirà dal carattere dell'indice specificato nel metodo incluso, fino al carattere dell'indice specificato nel metodo escluso. Esempio:
String greeting = "Hello, World!";
String sub = greeting.substring(0,5); // questa variabile conterrà soltanto "Hello"
String sub2 = greeting.substring(7,12); // questa varriabile conterrà "World"

DECISIONI

Per prendere una decisione all'interno di un programma si usa l'enunciato if: quando una determinata condizione è verificata, viene eseguito un certo insieme di enunciati, altrimenti ne viene eseguito un altro insieme. Se la condizione non viene rispettata allora viene utilizzato l'else e se in questo ramo non c'è nessuna azione da compiere allora lo si può omettere. Se il corpo dell'if ha un solo enunciato allora le parentesi graffe non servono. Se nell'if e nell'else ci sono codici duplicati allora bisogna spostarlo al di fuori dell'enunciato.

Gli operatori relazionali, ossia quelli che verificano una relazione esistente tra due valori sono:

==	Uguale a	op1 == op2	op1 e op2 sono uguali
!=	Diverso da	op1 != op2	op1 e op2 sono diversi
<	Minore di	op1 < op2	op1 è minore di op2
>	Maggiore di	op1 > op2	op1 è maggiore di op2
<=	Minore o uguale di	op1 <= op2	op1 è minore o uguale di op2
>=	Maggiore o uguale di	op1 >= op2	op1 è maggiore o uguale di op2

Per verificare se due stringhe sono uguali si utilizza l'equals: if (string1.equals(string2)) …

Perché scrivere string1==string2 verifica se le due stringhe si trovano nella stessa posizione all'interno della memoria. Se le due stringhe non sono identiche, perché magari differiscono di lettere maiuscole, si utilizza di compareTo: string1.compareTo(string2)==0.

Per confrontare degli oggetti, ossia se hanno lo stesso contenuto, bisogna usare l'equals.

Un riferimento può avere il valore *null* se non si riferisce ad alcun oggetto. Per verificare se un riferimento ha il valore null si usa l'operatore ==.

Per prendere decisioni con più di 2 alternative vengono utilizzati gli

else if(condizione) dopo il primo if (condizione) e alla fine ci sarà l'else. In questi casi bisogna prima verificare le prime condizioni più specifiche e poi quelle più generiche.

Per confrontare un singolo valore rispetto a diversi valori alternativi si può usare l'enunciato switch. Ciascuna diramazione dell'enunciato deve terminare con l'istruzione break. (pagina 218-219).

A volte è necessario inserire l'enunciato if all'interno di un altro enunciato if e in questo caso si parla di enunciati annidati.

All'interno degli if è possibile dichiarare variabili che vengono eliminate non appena il programma esce dall'if. Peò se la variabile serve anche al di fuori del blocco bisogna definirla all'esterno. È anche possibile avere variabili locali con nomi identici se i loro ambiti di visibilità non hanno parti in comune.

Per le condizioni logiche vengono spesso utilizzate condizioni vere o false, ossia valori booleani. Per decisioni più complesse si ha bisogno di combinare valori booleani.

&&	and	AND booleano
||	or	OR booleano
!	not	NOT booleano

ITERAZIONI

L'enunciato while è un esempio di ciclo che esegue ripetutamente le operazioni al suo interno solo se la condizione risulta vera. Quando una variabile viene dichiarata all'interno del corpo del ciclo ad ogni nuova iterazione la variabile viene creata all'inizio ed eliminata alla fine. La sintassi è la seguente: while (condizione) { … }

Se invece si vuole far eseguire una sequenza di enunciati per un numero determinato di volte, viene usato il ciclo for : for (int i = valore ; condizione ; incremento o decremento di i) { …}. L'inizializzazione della variabile i viene eseguita una sola volta prima di entrare nel ciclo, la condizione viene verificata prima di ciascuna iterazione, l'aggiornamento viene eseguito dopo l'iterazione. Questo tipo di iterazione viene anche detta ciclo definito perché si sa a priori quante volte verrà eseguito il ciclo.

Quando invece si vuole eseguire il ciclo almeno una volta, verificando perciò le condizioni dopo aver eseguito il corpo, viene utilizzato il ciclo do con la seguente sintassi: do { …} while(condizione); viene spesso utilizzato nei programmi che controllano che i valori inseriti dagli utenti siano corretti.

Le iterazioni possono essere annidate, dove il ciclo esterno esamina le righe della tabella, mentre quello interno elabora le colonne della riga in esame.

Per generare numeri casuali, almeno all'apparenza, viene utilizzata la classe random. Per quanto riguarda gli interi si usa il metodo nextInt(n), dove n è un numero intero compreso tra zero incluso ed n escluso, mentre per i double si usa nextDouble(), cioè un numero in virgola mobile compreso tra zero incluso e 1 escluso.

ARRAY E VETTORI

Gli array sono il meccanismo fondamentale per realizzare raccolte di più valori dello stesso tipo. La dichiarazione dell'array avviene in questi due modi: double[ ] nome = new double[lunghezza]; oppure se si conoscono già i valori da inserire: int [ ] nome = {valori};. L'indice dell'array parte da 0 e finisce perciò con il valore della sua lunghezza meno 1. Per ispezionare l'array viene utilizzato un ciclo for dove la variabile i agisce da indice.

Per fare una copia dell'array non bisogna usare l' "=" perché in questo modo viene copiato il riferimento di memoria, viene quindi usato il metodo: double[ ] nome = Arrays.copyOf(array da copiare, lunghezza);. La lunghezza dell'array non può essere modificata durante l'esecuzione del programma, quindi se non si sa a priori la lunghezza dell'array, ne verrà creato uno con lunghezza molto elevata e verrà riempito solo in parte tenendo traccia con una variabile ausiliaria fino a che indice l'array è stato riempito.

Per scandire gli elementi degli array senza modificarli, viene usato il ciclo for esteso che esegue il ciclo per ogni elemento dell'array. for (double nomevariabile : nomearray) { corpo ciclo} la variabile quindi prende come elementi i valori dei dati negli array. Per vdere la lunghezza dell'array si usa il metodo: nomeArray.length;

Algoritmi principali per gli array pag 351 – 359.

Quando la grandezza dell'array non è definita, viene usato un vettore le cui dimensioni possono aumentare o diminuire in base alle necessità oltre la classe arrayList fornisce vari metodi per svolgere le operazioni più comuni. Per usare l'arrayList bisogna prima importare la classe con: import java.util.ArrayList;. Negli arrayList non possono essere utilizzati i tipi primitivi come parametro di tipo, ma vengono usati delle classi involucro:

Tipo primitivo	Classe involucro
byte	Byte
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

La conversione tra i tipi primitivi e le corrispondenti classi involucro è automatica e anche i valori di tipo involucro vengono estratti automaticamente per generare valori di tipo primitivo.
L'invocazione dell'arrayList avviene in questo modo:
ArrayList<ClasseInvolucro> nome = new ArrayList<ClasseInvolucro>( );
Per aggiungere un oggetto nell'ultima posizione del vettore si usa il metodo add: nomeVettore.add(valore); mentre se si vuole specificare anche la posizione tra due valori e quindi spostando tutti gli elementi di una posizione in avanti rispetto all'elemento aggiunto, viene usato in questo modo: nomeVettore.add(posizione, valore); In entrambi i casi la dimensione del vettore viene incrementata di uno.
Al contrario il metodo remove elimina l'elemento che si trova in una determinata posizione spostando quindi tutti gli elementi successivi indietro di una posizione: nomeVettore.remove(posizione);
Per modificare un valore viene invece usato il metodo set: nomeVettore.sett(posizione, valore);
Per sapere le dimensione del vettore si utilizza il metodo size: nomeVettore.size( );
Per ispezionare un elemento si usa il metodo get: nomeVettore.get(posizione);
Per copiare un vettore e non il suo riferimento, deve essere passato come parametro al costruttore il vettore originale: ArrayList<classeInvolucro> nome = new ArrayList<ClasseInvolucro>(nomeVettoreDaCopiare);
Algoritmi usati nei vettori pag 392 – 393.

Quindi se le dimensioni del "contenitore" non cambiano mai è meglio usare l'array e lo stesso se bisogna memorizzare valori di tipo primitivo facendo particolare attenzione all'efficienza, mentre per tutti gli altri casi si utilizza un vettore.

PROGETTAZIONE DI CLASSI

Nelle classi possono essere presenti metodi modificatori, ossia che modificano lo stato di un oggetto con cui viene invocato, e metodi d-accesso, ossia metodi che richiedono soltanto all'oggetto di calcolare un determinato risultato senza modificarne lo stato. Esistono alcune classi dette immutabili che sono progettate in modo da avere solamente metodi d'accesso, come la classe string. Importante è quindi salvare il risultato in delle variabili quando viene invocato un metodo di tale classe.
A volte occorre memorizzare valori che appartengono ad una classe e non ad un suo oggetto utilizzando una variabile statica. Un esempio è il numero di conto bancario che viene aggiornato ogni volta che viene creato un nuovo oggetto. La dichiarazione avviene in questo modo: private static int nomeVariabile = valore; se si tratta di una costante allora può essere public.
Nelle classi possono esserci metodi che non vengono invocati con degli oggetti come parametro implicito, ossia i metodi statici come per esempio i metodi della classe math. Per invocare un metodo statico bisogna prima scrivere il nome della classe che lo contiene: Math.sqrt(nomeVaribile/numero); Un altro esempio di metodo static è il metodo main, perché è il primo metodo eseguito prima ancora che vengano creati degli oggetti.

EREDITARIETÀ

È la relazione esistente tra una classe più generica detta superclasse e una più specifica detta sottoclasse. L'uso dell'ereditarietà consente di riutilizzare codice invece di ricopiarlo in quanto una sottoclasse eredita i metodi della superclasse. Perciò quando si crea una nuova sottoclasse, bisogna specificare cosa la rende diversa dalla sua superclasse, un oggetto della sottoclasse possiede automaticamente tutte le variabili esemplari che sono dichiarate nella superclasse, quindi occorre dichiarare soltanto le variabili che non sono presenti. Lo stesso vale per i metodi che quindi andranno definiti solo quelli nuovi oppure che modificano l'implementazione dei metodi ereditati sovrascrivendoli. Per dichiiarare la sottoclasse bisogna utilizzare le parole "extends NomeSuperclasse":
public class NomeSottoclasse extends NomeSuperClasse { …}.
Le variabili private della superclasse sono inaccessibili e soltanto i metodi della superclasse possono accedervi, utilizzando per esempio il metodo getValore();.
Quando un metodo della superclasse viene sovrascritto, bisogna innanzitutto richiamarlo all'interno del nuovo metodo utilizzando la parola super: super.metodoSuperClasse();

In realtà tutte le classi che vengono create sono delle sottoclassi della classe *Object*, in cui sono presenti alcuni metodi molto generici come toString, equals,… Tali metodi come il toString deve essere sovrascritto nelle classi che progettiamo con la nostra versione personale, in quanto stamperà il codice hash dell'oggetto a cui viene applicato e non il suo contenuto.

ORDINAMENTO E RICERCA

Un algoritmo di ordinamento dispone gli elementi di una raccolta di dati in modo che siano memorizzati in qualche ordine specifico. Esistono più algoritmi per fare un ordinamento, uno dei più comuni è l'*ordinamento per selezione*: tale algoritmo ordina un array cercando ripetutamente l'elemento minimo della zona non ancora ordinata e una volta trovato lo pone all'inizio della zona non ancora ordinata. Una prima fase quindi è la ricerca dell'elemento più piccolo che viene messo nella prima posizione e l'elemento che in precedenza si trovava nella prima posizione verrà salvato in una variabile ausiliaria, perché sennò andrebbe perso, e posto nella posizione che occupava il numero più piccolo. Una seconda fase è la ricerca del numero più piccolo nella zona non ordinata, in questo caso dalla posizione 2 (array[1]) in poi dell'array, e una volta trovato, verrà posto, con un procedimento analogo a quello utilizzato nella prima fase, all'inizio della zona non ordinata. Verrà utilizzato questo procedimento fino alla lunghezza dell'array -1, in quanto l'ultimo numero è per forza di cose il più grande. Lo svantaggio di questo algoritmo è la sua lentezza con array molto grandi.
Un altro algoritmo di ordinamento è l'*insertion sort* che dato un array, lo ordina a mano a mano che lo scorre. Quindi parte dalla posizione 1 e confronta l'elemento in questa posizione con quello in posizione 0 e se è più piccolo allora gli scambia, utilizzando sempre una variabile ausiliaria per non perdere il valore più grande che altrimenti andrebbe perso. A questo punto confronta l'elemento in posizione 2 con quello in posizione precedente e ,se esso risulta più piccolo, allora lo scambia; successivamente confronta l'elemento appena scambiato, che in questo momento si trova in posizione 1, con l'elemento in posizione 0 e se dovesse risultare più piccolo allora verrà scambiato per essere messo in posizione 0 utilizzando sempre la variabile ausiliaria per non perdere il valore tra i due più grande. Viene eseguito questo processo per ogni elemento dell'array.

Allo stesso modo esistono diversi algoritmi per effettuare delle ricerche. Uno di questi avviene con la *ricerca lineare*, ossia un algoritmo che scorre ed esamina tutti gli elementi fino a quando trova la corrispondenza oppure arriva fino alla fine dell'elenco.
Un altro algoritmo effettua una *ricerca binaria*, ossia che viene ricercato un valore in un array ordinato determinando se si può trovare tale valore nella prima metà o nella seconda metà dell'array, ripetendo la ricerca e quindi dividendo l'array una nuova volta nella metà in cui si presuppone che esso sia presente. Questa operazione di divisione viene fatta finché non è più possibile effettuarla, e quindi non viene ritrovato il numero, oppure finché il numero viene trovato.

Codice ordinamento:

```
public class ordinamentoPerSelezione
{
    private int[] array;

    public ordinamentoPerSelezione(int[] anArray)
    {
      array = anArray;
    }

    public void sort()
    {
       for(int i = 0; i < array.length-1; i++)
        {
           int minimo = minPos(i);
           swap(minimo,i);
        }
    }

    private int minPos(int from)
    {
       int min = from;

       for(int i = from + 1; i < array.length; i++)
        {
           if(array[i] < array[min])
            {
               min = i;
            }
        }
       return min;

    }

   private void swap(int i, int j)
   {
       int temp = array[i];
       array[i] = array[j];
       array[j] = temp;
   }
}
```

```
public class InsertionSort
{
   private int[] a;

   public InsertionSort(int[] anArray)
   {
      a = anArray;
   }

   public void sort()
   {
      for (int i = 1; i < a.length; i++)
      {
         int next = a[i];
         int j = i;
         while (j > 0 && a[j - 1] > next)
         {
            a[j] = a[j - 1];
            j--;
         }
         a[j] = next;
      }
  }
}
```

Codice ricerca lineare:

```
public class LinearSearcher
{
   private int[] a;

   public LinearSearcher(int[] anArray)
   {
      a = anArray;
   }
   public int search(int v)
   {
      for (int i = 0; i < a.length; i++)
      {
         if (a[i] == v)
         {
           return i;
         }
      }
      return -1;
   }
 }
```