

Selezione e iterazione

Settimana V

Scopo della lezione

Laboratorio di Informatica Generale

- Ripassare i concetti di selezione e iterazione.
- Dare nozioni di programmazione strutturata.
- Verificare come uno stesso programma possa essere scritto in forme differenti.

V.2

Programmazione strutturata

- Si basa su un piccolo insieme di strutture di controllo predefinite
 - **Sequenza**: le istruzioni sono eseguite in ordine sequenziale a meno che il loro flusso non sia interrotto da una delle rimanenti strutture di controllo.
 - **Selezione**: permette di modificare il flusso di esecuzione delle istruzioni scegliendo tra due o più alternative.

V.3

Programmazione strutturata

- **Iterazione**: permette di ripetere ciclicamente un'istruzione o una sequenza di istruzioni.
- **Invocazione di metodo**: il flusso di esecuzioni è temporaneamente trasferito all'inizio del metodo invocato, terminato il quale ritorna all'invocante, partendo dall'istruzione successiva all'invocazione stessa.

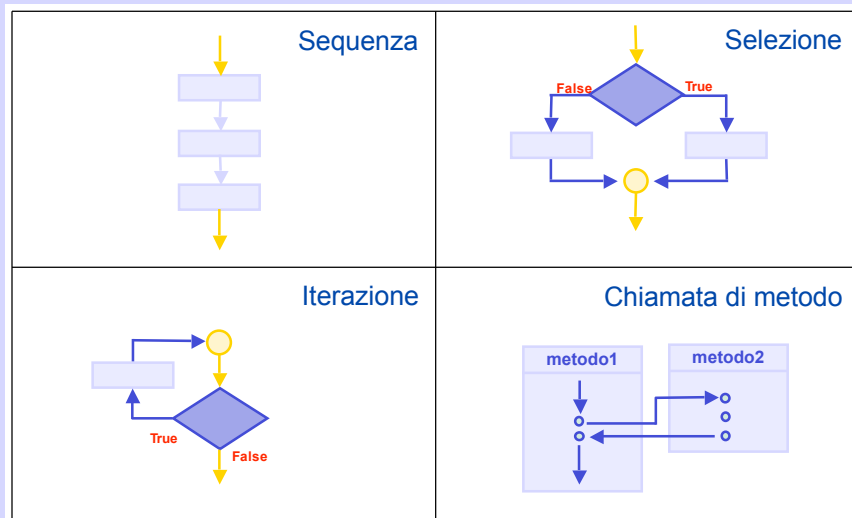
V.4

Programmazione strutturata

- Non importa quanto grande e complesso sia il programma, il suo flusso di controllo potrà sempre essere espresso come combinazione di questi quattro costrutti.

V.5

Costrutti



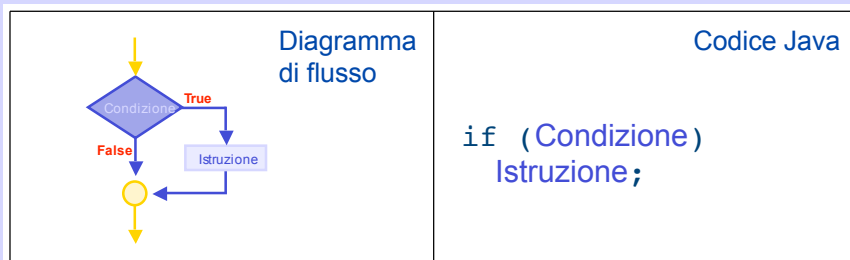
V.6

Flusso di controllo

- La sequenza è stata vista nella scorsa lezione, mentre vedremo in seguito le chiamate di metodo.
- Il flusso di esecuzione del programma è gestito da istruzioni di controllo che permettono di scegliere un cammino tra molti.
- Il cammino viene scelto in base al soddisfacimento di alcune condizioni.

V.7

If-then



- Se l'espressione booleana che rappresenta **Condizione** è valutata true, allora verrà eseguita Istruzione. Altrimenti Istruzione non verrà eseguita.

V.8

Espressioni booleane

- Sono espressioni che assumono valori booleani, cioè che valgono vero o falso.
- Esempi di espressioni booleane:

true	false
isSleeping	(1+1)==2

- == è l'operatore di uguaglianza in Java,
!= è la negazione dell'operatore di uguaglianza (i.e. l'operatore di diversità).

V.9

Espressioni booleane

I dati boolean hanno solo due possibili valori: true e false.

$o_1 \ || \ o_2$ è true se uno dei due operandi è true.

$!o_1$ è true quando o_1 è false.

$o_1 \ \&\& \ o_2$ è true solo se entrambi o_1 e o_2 sono true.

$o_1 \ \wedge \ o_2$ è true se solo uno tra o_1 e o_2 è true.

		AND	OR	XOR	NOT
o_1	o_2	$o_1 \&\& o_2$	$o_1 \ \ o_2$	$o_1 \wedge o_2$	$!o_1$
true	true	true	true	false	false
true	false	false	true	true	false
false	true	false	true	true	true
false	false	false	false	false	true

V.10

Ordine di precedenza

Ordine di precedenza	Operatore	Operazione
1	()	Parentesi
2	!	NOT
3	&&	AND
4	^	XOR
5		OR

In un'espressione mista, la valutazione del NOT precede quella dell'AND, che precede quella dell'XOR, che precede quella dell'OR.

AND è valutato prima dell'OR perché ha una precedenza più alta.

ESPRESSIONE

true || true && false

(true || true) && false

true || (true && false)

VALUTAZIONE

true || false → true

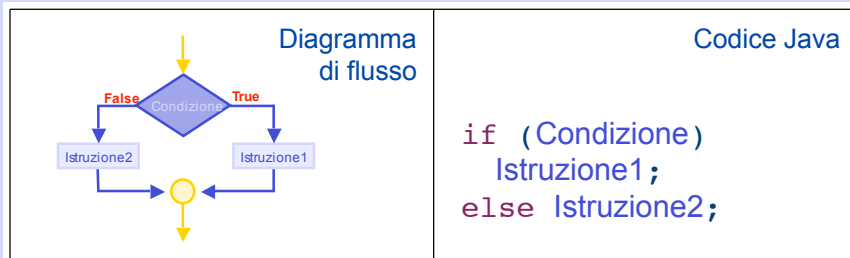
true && false → false

true || false → true

Le parentesi possono annullare la relazione di precedenza.

V.11

If-then-else



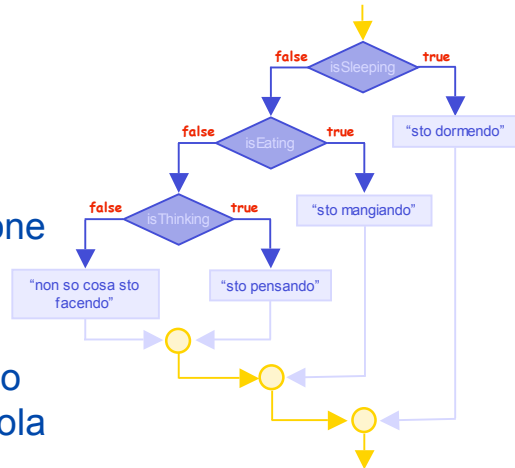
- Se l'espressione booleana che rappresenta **Condizione** è valutata true, allora verrà eseguita Istruzione1, altrimenti verrà eseguita Istruzione2.

V.12

Selezione multipla

Diverse istruzioni if-then-else possono essere messe in cascata per formare una struttura di selezione multipla.

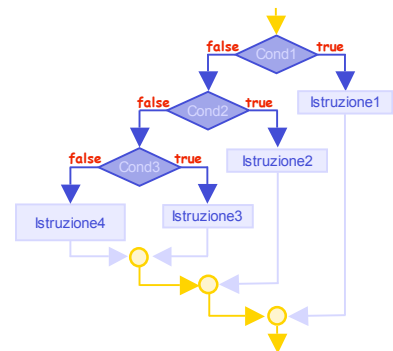
Nota: questa struttura ha un solo ingresso ed una sola uscita



V.13

Selezione multipla

Diagramma di flusso



Codice Java

```

if (Cond1)
    Istruzione1;
else if(Cond2)
    Istruzione2;
else if (Cond3)
    Istruzione3;
else Istruzione4;
  
```

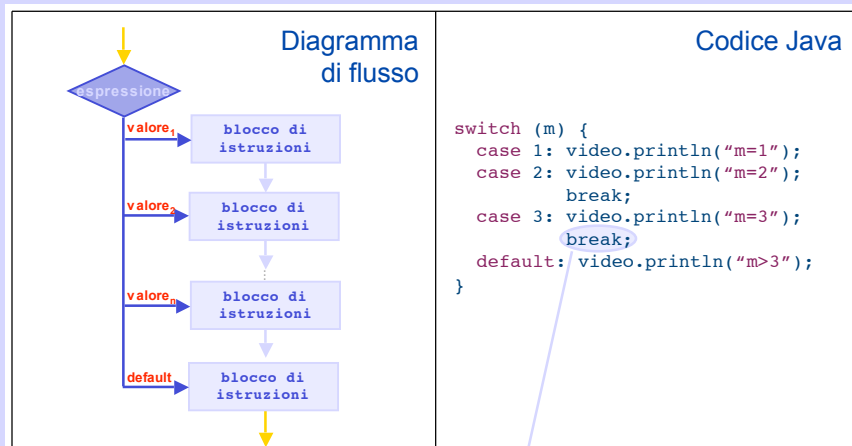
V.14

Switch

- Sintassi: `switch (Espressione) {`
`case valore_1: blocco_istruzioni_1`
`...`
`case valore_n: blocco_istruzioni_n`
`default: blocco_istruzioni_default`
`}`
- Semantica: i) valuta *Espressione*, ii) passa il controllo all'etichetta *case* il cui valore corrisponde a quello ottenuto valutando *Espressione* (o alla clausola di *default* quando nessuna etichetta corrisponde al valore calcolato), iii) esegue tutte le istruzioni fino alla fine del costrutto.

V.15

Switch



l'istruzione `break` trasferisce il controllo fuori dallo `switch`.

V.16

Esempi

if

```
if (isEating)
    return "Sta mangiando";
```

if-then-else

```
if (isEating)
    System.out.println("Sta mangiando");
else
    System.out.println("Non sta mangiando");
```

Selezione Multipla

```
if (isSleeping)
    System.out.println("Sta dormendo");
else if (isEating)
    System.out.println("Sta mangiando");
else if (isThinking)
    System.out.println("Sta pensando");
else
    System.out.println("Errore: non so che sta facendo");
```

V.17

Il problema dell'else pendente

- Il programmatore deve stare attento ad associare ogni `else` con il corrispondente `if`.
- Ogni `else` è associato all'`if` più vicino non ancora associato.
- L'indentazione (che il compilatore ignora) dovrebbe riflettere la logica dell'istruzione.

V.18

Il problema dell'else pendente

Indentazione Corretta

```
if (condition1)
    if (condition2)
        System.out.println("One");
    else
        System.out.println("Two");
```

Indentazione Errata

```
if (condition1)
    if (condition2)
        System.out.println("One");
else
    System.out.println("Two");
```

V.19

Il problema dell'else pendente

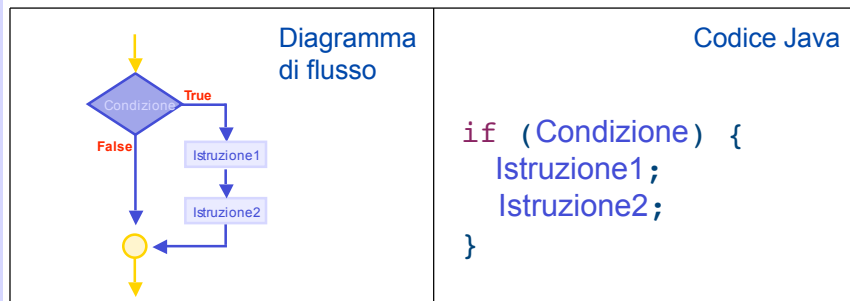
- Nel caso si voglia eludere questa regola è necessario utilizzare le parentesi graffe:

```
if (condition1) {
    if (condition2)
        System.out.println("One");
}
else
    System.out.println("Two");
```

- Implementate delle variazioni compilabili delle tre selezioni precedenti e verificate come vengono eseguite

V.20

If-then



- Le parentesi graffe possono essere usate anche per eseguire più istruzioni all'interno di un'istruzione `if`.

V.21

Iterazione

- Struttura di controllo pensata per ripetere una sequenza di istruzioni.
- Se il numero di iterazioni è noto si usa un ciclo numerato:

Contare il numero di volte
che la lettera 'a' ricorre
in un documento.

Stampare i numeri
da 1 a 5000

inizializza totalA a 0
per ogni carattere nel documento
se il carattere è una 'a'
aggiungi 1 a totalA
restituisce totalA come risultato

per ogni numero n da 1 a 5000
stampa n

V.22

Iterazione

- Se il numero di iterazioni è sconosciuto si usa un ciclo condizionato:

Cercare in un file
la scheda di uno
studente

ripeti i seguenti passi
leggi una scheda da file
finché non trovi la scheda di Paolo

Calcolare il numero medio di
orsi avvistati mensilmente

Inizializzare sumOfBears e
numOfMonths a 0
Ripeti i seguenti passi
leggi un numero dalla tastiera
aggiungilo a sumOfBears
aggiungi 1 a numOfMonths
Finché l'utente non vuole fermarsi
dividi sumOfBears per numOfMonths
restituisci la media

V.23

for



- Per eseguire più istruzioni all'interno di un ciclo `for` possono essere usate le parentesi graffe.

V.24

Variabile di ciclo

```
for (int k = 0; k < 100; k++) // per 100 volte
    System.out.println("ciao"); // stampa "ciao"
```

- Se **k** è dichiarata nell'istruzione **for**, non può essere usata fuori dal ciclo

```
for (int k = 0; k < 100; k++)
    System.out.println("ciao");
System.out.println("k = " + k); // Errore, k non è stata dichiarata
```

```
int k = 0; // dichiarazione dell'indice
for (k = 0; k < 100; k++)
    System.out.println("Ciao");
System.out.println("k = " + k); // uso corretto
```

V.25

for

- Il ciclo **for** inizializza un indice a un **valore iniziale** e poi conta 0 o più iterazioni fino a che non viene raggiunto un **limite**.

```
public static void main() {
    for (int k = 10; k > 0; k--)
        System.out.println(k);
    System.out.println("BOOOM!!!!!!");
} // main()
```

- La **condizione** verifica se il limite è stato raggiunto, l'**aggiornamento** modifica l'indice.

V.26

for

- Implementare un ciclo for che
 - Stampi i numeri da 1 a 10
 - Stampi i numeri da 10 a 1
 - Stampi i numeri pari da 1 a 10
 - Stampi i numeri da 10 a -10
 - Calcoli e in seguito stampi la somma dei primi n numeri, dove n è il valore di una variabile inserito dall'utente a inizio programma

V.27

Cicli infiniti

- Cicli che non raggiungono mai il proprio limite.

```
public static void main() {  
    for (int k = 10; k > 0; k++)  
        System.out.println(k);  
    System.out.println("BOOOM!!!!!!");  
} // main()
```

- Come è possibile modificare il codice precedente in modo che il ciclo termini?

V.28

Cicli annidati

- Sono cicli inseriti all'interno di altri cicli.

```
for (int row = 1; row <= 5; row++) {      // per ogni riga
    for (int col = 1; col <= 6 - row; j++) // stampa la riga
        System.out.print('#');
    System.out.println();                  // e vai a capo
}
```

Riga	Colonne
1	6-1=5
2	6-2=4
3	6-3=3
4	6-2=2
5	6-1=1

```
# # # # #
# # # #
# # #
# #
#
```

V.29

Cicli annidati

- Modificate il codice precedente in modo da ottenere i seguenti output

```
# # # # #
# # # #
# # #
# #
#
```

```
# #
# #
# #
# #
# #
# #
# #
```

```
#
# #
# #
# #
# #
# #
# #
```

```
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
```

V.30

Il problema 3N+1

- A partire da un intero positivo N_0 , generare la sequenza definita dalla seguente regola ricorsiva

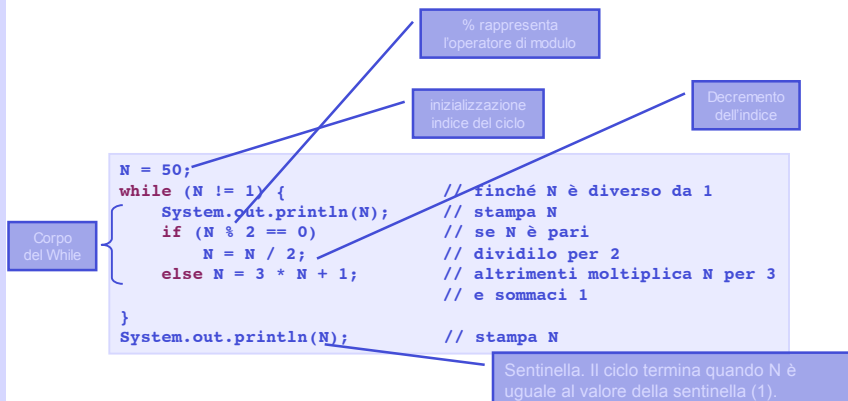
$$N_{i+1} = \begin{cases} 3N_i + 1 & \text{se } N \text{ è dispari} \\ N/2 & \text{se } N \text{ è pari} \end{cases}$$

- Ad esempio
 - 10,5,16,8,4,2,1
 - 7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1

V.31

While

- Non conoscendo a priori il numero di iterazioni non è opportuno usare `for`:



V.32

do-while

- Quanti giorni sono necessari per perdere metà del proprio peso se si dimagrisce del 2% ogni giorno?

```
import prog.io.*;

public class PerdiPeso {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        double matGrass = 100.0;      // inizializza il peso
        int nDays = 0;                // inizializza il contatore dei giorni
        do {                          // continua ad
            matGrass = matGrass - matGrass * 0.02;
            // aggiornare il peso
            nDays = nDays + 1;        // incrementare il n. di giorni
        } while (matGrass > 50.0);    // finché non ha perso il 50% del peso
        video.println(nDays);        // stampa il numero di giorni
    } // main
}
```

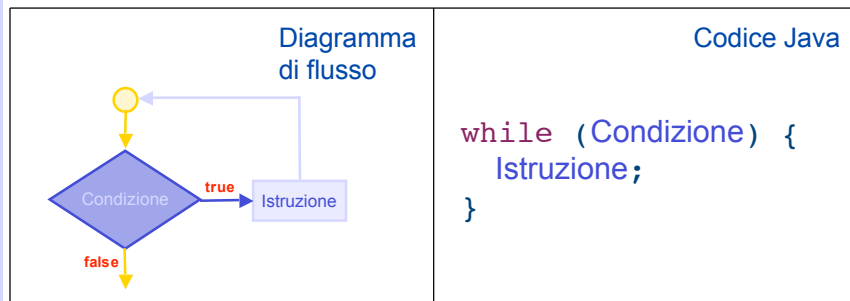
V.33

do-while

- Modificate il programma precedente in modo che i valori di matGrass e nDays vengano stampati ad ogni esecuzione del corpo del ciclo

V.34

While



- Istruzione viene eseguita fin tanto che Condizione viene valutata `true`.

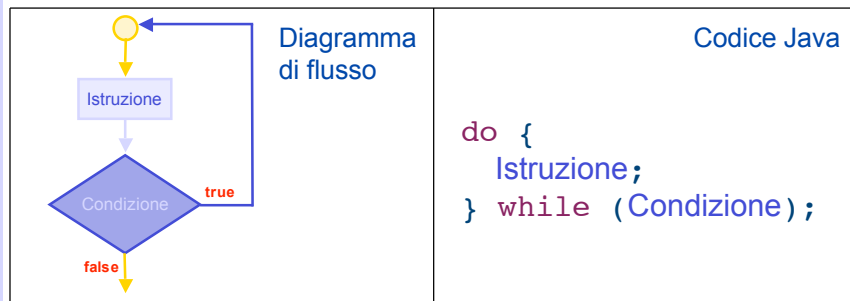
V.35

While

- Non sono previste sezioni per l'inizializzazione e l'aggiornamento di indici.
- Il corpo del ciclo può non venire mai eseguito.
- Può originare un ciclo infinito.

V.36

Do-while



- Istruzione viene eseguita fin tanto che Condizione viene valutata `true`.

V.37

Do-while

- Non sono previste sezioni per l'inizializzazione e l'aggiornamento di indici.
- Il corpo del ciclo viene eseguito almeno una volta.
- Può originare un ciclo infinito.

V.38

Calcolo della media

- Calcolare la media degli esami sostenuti
 - I voti (numeri interi) saranno introdotti dalla tastiera. I dati termineranno quando si introdurrà il valore 9999.
 - Verificare che il voto introdotto sia corretto (da 18 a 30), facendolo reimmettere fino a che non risulti corretto.
 - Suggerimento: usare do-while per gestire la verifica/reimmissione dell'input.

V.39

Calcolo della media

```
import prog.io.*;

public class MediaVoti {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        int esame, totale=0, i=0;
        do {
            i = i + 1; /* i++ */
            do {
                video.print("Immetti voto "+i+": ");
                esame = tastiera.readInt();
            } while (((esame<18) || (esame>30)) && (esame != 9999));
            if (esame != 9999) totale = totale + esame;
        } while(esame != 9999);
        video.println("La media è: "+totale/(i-1));
    }
}
```

V.40

Calcolo della media

Input/Output

```
import prog.io.*;

public class MediaVoti {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        int esame, totale=0, i=0;
        do {
            i = i + 1; /* i++ */
            do {
                video.print("Immetti voto "+i+": ");
                esame = tastiera.readInt();
            } while (((esame<18) || (esame>30)) && (esame != 9999));
            if (esame != 9999) totale = totale + esame;
        } while(esame != 9999);
        video.println("La media è: "+totale/(i-1));
    }
}
```

V.41

Calcolo della media

Controllo validità voto

```
import prog.io.*;

public class MediaVoti {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        int esame, totale=0, i=0;
        do {
            i = i + 1; /* i++ */
            do {
                video.print("Immetti voto "+i+": ");
                esame = tastiera.readInt();
            } while (((esame<18) || (esame>30)) && (esame != 9999));
            if (esame != 9999) totale = totale + esame;
        } while(esame != 9999);
        video.println("La media è: "+totale/(i-1));
    }
}
```

V.42

Calcolo della media

```
import prog.io.*;

public class MediaVoti {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        int esame, totale=0, i=0;
        do {
            i = i + 1; /* i++ */
            do {
                video.print("Immetti voto "+i+": ");
                esame = tastiera.readInt();
            } while (((esame<18) || (esame>30)) && (esame != 9999));
            if (esame != 9999) totale = totale + esame;
        } while(esame != 9999);
        video.println("La media è: "+totale/(i-1));
    }
}
```

Calcolo totale e media

V.43

Calcolo dei numeri primi

- Calcolare e stampare i primi 100 numeri primi.
 - I possibili numeri sono generati automaticamente e non immessi da tastiera.
 - Suggerimento: separare la verifica di primalità dal calcolo della sequenza.

V.44

Calcolo dei numeri primi

```
import prog.io.*;

public class CalcoloPrimi {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        boolean èPrimo;
        int i=0, p=1, div;

        do {
            p++;
            div = 2;
            èPrimo = true;
            while ((div <= Math.sqrt(p)) && èPrimo) {
                if (p%div == 0) èPrimo = false;
                else div++;
            }
            if (èPrimo) {
                i++;
                video.println(i+": "+p+" è primo!");
            }
        } while (i<100);
    }
}
```

V.45

Calcolo dei numeri primi

```
import prog.io.*;

public class CalcoloPrimi {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        boolean èPrimo;
        int i=0, p=1, div;

        do {
            p++;
            div = 2;
            èPrimo = true;
            while ((div <= Math.sqrt(p)) && èPrimo) {
                if (p%div == 0) èPrimo = false;
                else div++;
            }
            if (èPrimo) {
                i++;
                video.println(i+": "+p+" è primo!");
            }
        } while (i<100);
    }
}
```

Verifica di primalità

Radice Quadrata.
Std Java library

Java è
Unicode

//inizializzazione

//verifica primalità

V.46

Calcolo dei numeri primi

```

1: 2 è primo!    21: 73 è primo!    41: 179 è primo!    61: 283 è primo!    81: 419 è primo!
2: 3 è primo!    22: 79 è primo!    42: 181 è primo!    62: 293 è primo!    82: 421 è primo!
3: 5 è primo!    23: 83 è primo!    43: 191 è primo!    63: 307 è primo!    83: 431 è primo!
4: 7 è primo!    24: 89 è primo!    44: 193 è primo!    64: 311 è primo!    84: 433 è primo!
5: 11 è primo!   25: 97 è primo!    45: 197 è primo!    65: 313 è primo!    85: 439 è primo!
6: 13 è primo!   26: 101 è primo!   46: 199 è primo!    66: 317 è primo!    86: 443 è primo!
7: 17 è primo!   27: 103 è primo!   47: 211 è primo!    67: 331 è primo!    87: 449 è primo!
8: 19 è primo!   28: 107 è primo!   48: 223 è primo!    68: 337 è primo!    88: 457 è primo!
9: 23 è primo!   29: 109 è primo!   49: 227 è primo!    69: 347 è primo!    89: 461 è primo!
10: 29 è primo!  30: 113 è primo!   50: 229 è primo!    70: 349 è primo!    90: 463 è primo!
11: 31 è primo!  31: 127 è primo!   51: 233 è primo!    71: 353 è primo!    91: 467 è primo!
12: 37 è primo!  32: 131 è primo!   52: 239 è primo!    72: 359 è primo!    92: 479 è primo!
13: 41 è primo!  33: 137 è primo!   53: 241 è primo!    73: 367 è primo!    93: 487 è primo!
14: 43 è primo!  34: 139 è primo!   54: 251 è primo!    74: 373 è primo!    94: 491 è primo!
15: 47 è primo!  35: 149 è primo!   55: 257 è primo!    75: 379 è primo!    95: 499 è primo!
16: 53 è primo!  36: 151 è primo!   56: 263 è primo!    76: 383 è primo!    96: 503 è primo!
17: 59 è primo!  37: 157 è primo!   57: 269 è primo!    77: 389 è primo!    97: 509 è primo!
18: 61 è primo!  38: 163 è primo!   58: 271 è primo!    78: 397 è primo!    98: 521 è primo!
19: 67 è primo!  39: 167 è primo!   59: 277 è primo!    79: 401 è primo!    99: 523 è primo!
20: 71 è primo!  40: 173 è primo!   60: 281 è primo!    80: 409 è primo!    100: 541 è primo!

```

V.47

Tavole di verità

- Calcolare e visualizzare la tavola di verità per l'espressione

$$X_1 \wedge (X_2 \vee \neg X_1) \wedge \neg X_3$$

- Suggerimenti:

0	false
1	true
*	$\wedge$
+	$\vee$
1-x	$\neg X$

V.48

Tavole di verità

```
import prog.io.*;

public class TavolaVerita { // 0 falso e 1 vero

    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        int x1, x2, x3; // una variabile per ogni valore di verità

        video.println
            ("Tavola di Verità per: f = x1 and (x2 or not x1) and not x3.\n");
        video.println("x1 x2 x3 | f\n-----|--");
        for (x1=0;x1<=1;x1++) // for annidati: uno per operando della
            formula
                for (x2=0;x2<=1;x2++)
                    for (x3=0;x3<=1;x3++)
                        video.println
                            (" "+x1+" "+x2+" "+x3+" | "+(x1*(x2+(1-x1))*(1-x3)));
    }
}
```

V.49

Tavole di verità

```
malchiod% java TavolaVerita
Tavola di Verità per: f = x1 and (x2 or not x1) and not x3.

x1 x2 x3 | f
-----|--
0 0 0 | 0
0 0 1 | 0
0 1 0 | 0
0 1 1 | 0
1 0 0 | 0
1 0 1 | 0
1 1 0 | 1
1 1 1 | 0
malchiod%
```

V.50

Tavole di verità

- Modificate il programma precedente in modo che, invece di stampare una tavola di verità, dica se l'espressione di partenza è una tautologia, una contraddizione o una formula soddisfacibile.
- Verificate il funzionamento delle due versioni del programma per altre espressioni

V.51

Giochiamo a carte

Problema: scrivere un algoritmo che

- legga da tastiera due caratteri;
- i caratteri codificano le carte da gioco come segue:

A	asso	K	re
2-9	carte da 2 a 9	Q	quadri
0	dieci	C	cuori
J	fante	P	picche
D	donna	F	fiori

- stampi la descrizione completa

Suggerimento: usare il costrutto switch.

V.52

Giochiamo a carte

```
import prog.io.*;
public class Queen {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        char seme, punto;

        punto = tastiera.readChar("punto: ");
        seme = tastiera.readChar("seme: ");
        switch (punto) {
            case '0': video.print("dieci");      break;
            case 'A': video.print("asso");      break;
            case '2': video.print("due");       break;
            case '3': video.print("tre");       break;
            case '4': video.print("quattro");   break;
            case '5': video.print("cinque");    break;
            case '6': video.print("sei");       break;
            case '7': video.print("sette");     break;
            case '8': video.print("otto");      break;
            case '9': video.print("nove");      break;
            case 'J': video.print("fante");     break;
            case 'D': video.print("donna");     break;
            case 'K': video.print("re");        break;
        }
    }
}
```

V.53

Giochiamo a carte

```
    }
    video.print(" di ");
    switch (seme) {
        case 'C': video.print("cuori");
            break;
        case 'Q': video.print("quadri");
            break;
        case 'F': video.print("fiori");
            break;
        case 'P': video.print("picche");
            break;
    }
    video.println();
}
}
```

V.54

Giochiamo a carte

```
[19:10]cazzola@ulik:esercizi>java Queen
punto: A
seme: Q
asso di quadri
[19:10]cazzola@ulik:esercizi>java Queen
punto: J
seme: C
fante di cuori
[19:11]cazzola@ulik:esercizi>java Queen
punto: Q
seme: C
donna di cuori
```

V.55

Esercizi

- Scrivere un programma `MediaPesataVoti` che calcola la media degli esami pesata sui crediti, modificando opportunamente `MediaVoti`. La media pesata sui crediti si ottiene moltiplicando ogni voto per il numero di crediti del corrispondente corso, sommando le quantità così ottenute e dividendo per il numero totale di crediti. La regola di fermata diventerà "è stato introdotto 9999 sia come voto che come numero di crediti".

V.56

Esercizi

- Modificare `CalcoloPrimi` in modo da usare il ciclo `do-while` anziché `while` per il test di primalità.
- Siano `a` e `b` due valori interi introdotti da tastiera. Scrivere un programma che calcoli $a*b$ utilizzando solo le operazioni aritmetiche `+` e `-`.

V.57

Esercizi

- Ricontrollate tutti gli esercizi svolti questa settimana e verificate che tutte le varianti introdotte delle strutture di selezione e di iterazione siano state implementate. In caso contrario modificate opportunamente alcuni degli esercizi proposti in modo che richiedano l'utilizzo delle varianti non implementate.

V.58