Tree school

Modulo 2

Switch, approfondimento array, metodi

M. Massotti

Congratulazioni, siete sopravvissuti alla teoria!

Switch

- **Potremmo avere l'esigenza di scegliere quale codice eseguire in base a al valore di una determinata variabile**
- **La soluzione accanto funziona ma è sicuramente di difficile lettura.**
- **In questo caso ci viene in soccorso il costrutto switch!**

```
if(a == 0)
    System.out.println("0");
else if(a == 1)
    System.out.println("1");
else if(a == 2)
    System.out.println("2");
else if(a == 3)
    System.out.println("3");
else if(a == 4)
    System.out.println("4");
else if(a == 5)
    System.out.println("5");
else if(a == 6)
    System.out.println("6");
else
    System.out.println("Caso non previsto");
```

Switch

Non questo di Switch!

Switch

- **Questo!**
- **Esplicitamente dividete i casi previsti**
- **A fine di ogni caso mettete un break altrimenti il codice continua a valutare i case.**
- **Buona regola è quella di definire il caso di default, e cioè quello dei casi non previsti, anche vuoto.**

```
switch(a){
    case 1:
        System.out.println("1");
        break;
    case 2:
        System.out.println("2");
        break;
    case 3:
        System.out.println("3");
        break;
    case 4:
        System.out.println("4");
        break;
    case 5:
        System.out.println("5");
        break;
    default:
        System.out.println("Caso non previsto");
        break;
}
```

Array in java - approfondimento

- **Dichiarare un array è molto semplice in Java, bisogna specificare il tipo e la dimensione voluta.**
- **Per accedere ad un elemento vi basta chiamare l'array e inserire tra le parentesi quadre l'indice che volete leggere.**
- **Ricordatevi sempre che il primo elemento è lo zero!**
- **Per leggere la dimensione di un array, vi basta scrivere il nome di un array e aggiungere .length**

```
int[] arrayInteri = new int[5];

arrayInteri[0] = 2;

System.out.println(arrayInteri[0]);

System.out.println(arrayInteri.length);
```

Stampare il contenuto di un array

- **Se provate a dare in pasto al println un array, non vi stampa l'array!**
- **Vi stamperà il suo indirizzo di memoria (abbastanza inutile).**
- **Ovviamente vi potete scrivere il vostro ciclo for che vi stampa l'array come volete ma java offre già una maniera per stamparlo**
- **Arrays.toString(nomeArray) vi stamperà il vostro array!**

```
System.out.println(Arrays.toString(arrayInteri));
```

Inizializzare un array direttamente

```
int[] interi = {1,2,3};
```

E se volessi cambiare la dimensione di un array?

- **Come abbiamo visto nella scorsa lezione un array è immutabile.**
- **Tuttavia qualora il programma si rendesse conto di aver bisogno di più spazio, può creare un array più grande, copiare il contenuto del vecchio al nuovo e sostituire il nuovo array al vecchio.**
- **ATTENZIONE: Dopo questa operazione arrayInteri e nuovoArray punteranno allo stesso array in memoria, perciò un cambiamento in nuovoArray si ripercuoterà in arrayInteri e viceversa.**
- **Per stare sicuri di non fare danno potete assegnare un nuovo array vuoto a nuovoArray**

```
int[] arrayInteri = new int[5];
int[] nuovoArray = new int [10];

for(int i = 0; i < arrayInteri.length; i++)
    nuovoArray[i] = arrayInteri[i];

arrayInteri = nuovoArray;

nuovoArray = new int [10]; //Così è safe :)
```

Matrici

- **Con java è possibile creare array multidimensionali**
- **In questo caso specifico stiamo creando una matrice 5 * 10**
- **È un po' come avere un array il cui tipo delle celle sono anche essi degli array.**
- **Quando usate una matrice fate molta attenzione a come scrivete i due indici, la posizione 2,3 è diversa dalla posizione 3,2!**
- **Se ad una matrice usate un solo indice invece che due, otterrete come risultato l'intera riga in forma di Array.**

```
int[][] matrice = new int [5][10];

matrice[0][0] = 5;

System.out.println(matrice[0][0]);

System.out.println(Arrays.toString(matrice[0]));

for(int i = 0; i<matrice.length; i++)
    for(int j = 0; j < matrice[0].length; j++)
        matrice[i][j] = 3;
```

Funzioni 1/2

- **Più che il concetto di funzione pura, in Java è presente il concetto di metodo.**
- **Capirete bene il concetto di metodo nei moduli successivi.**
- **Per il momento, quando volete raggruppare pezzi di codice scrivete usate la sintassi accanto**
- **Scrivete «public static» e successivamente il tipo dell'output, se la funzione non torna nulla scrivete void.**
- **Scrivete poi il nome della funzione con (tra parentesi tonde) gli input che prende, se non ne prende nessuno potete lasciare vuoto.**

```
public static int somma(int a, int b)
{
    return a + b;
}

public static void saluta(String nome)
{
    System.out.println("Ciao " + nome + "!");
}
```

Funzioni 2/2

- **Una funzione che restituisce un output non void deve avere almeno un'istruzione di return dove ritorna un output del tipo dichiarato**
- **È possibile avere più punti di uscita e quindi più return**
- **Può esistere solo una funzione con un dato nome che prende certi specifici tipi di dato come input, però può esistere una funzione con lo stesso nome ma che prende in input diversi tipi di dato.**

```
public static int operazioniSuNumero(int a, String operazione)
{
    switch (operazione)
    {
        case "quadrato":
            return a * a;
        case "cubo":
            return a * a * a;
        case "metà":
            return a / 2;
        default :
            return 0;
    }
}

public static String operazioniSuNumero(int a)
{
   return "Il numero selezionato è:" + a;
}
```

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Valori randomici

Random value

- **Ottenere un valore decimale casuale in Java è possibile tramite il metodo statico *random()* della classe *Math***
- **Questo metodo restituisce un valore decimale maggiore o uguale di 0 e minore di 1**
- **Possiamo facilmente ottenere valori interi o booleani**

```java
// Obtain a number between [0.0 - 1.0)
double randomValue = Math.random();

// Obtain a number between [0 - 49]
int randomInt = (int) (randomValue * 50);

// Obtain a boolean value
boolean randomBool = randomValue > 0.5;
```

La classe *Random*

- **Il package *java.util* fornisce inoltre una classe *Random* per la generazione di valori casuali**
- **La classe *Random* consente di creare direttamente valori casuali di tipo *int*, *float*, *boolean* e *double***

```
import java.util.Random;

Random rand = new Random();

// Obtain a number between [0.0 - 1.0)
double randomDouble = rand.nextDouble();

// Obtain a number between [0 - 49]
int randomInt = rand.nextInt(50);

// Obtain a boolean value
boolean randomBool = rand.nextBoolean();
```

Random in un intervallo

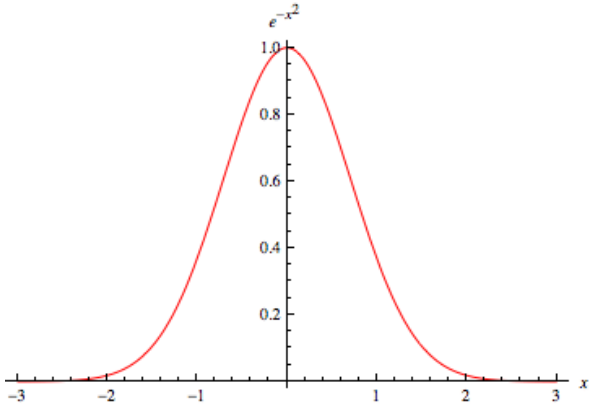
- **Se vi serve un valore casuale in uno specifico intervallo (per esempio compreso tra 34 e 41 per simulare la temperatura corporea di un uomo) potete sommare al valore base un numero casuale che rappresenta il range di distacco rispetto alla base**

```
int min = 34;
int max = 41;

// Obtain a number between [min - max]
int randomValueInRange = min + rand.nextInt(max - min + 1);
```

Random in un intervallo particolare 1/2

- **Riprendendo l'esempio della simulazione di temperature, sappiamo tutti che mediamente le persone hanno una temperatura intorno al 36,5 e pochissimi hanno una temperatura superiore al 40 o sotto il 35 (e meno male!)**
- **Il metodo *nextGaussian* della classe *Random* ci restituisce un valore casuale proveniente da una distribuzione gaussiana con valore medio 0 e standard deviation 1**
- **Tradotto in italiano, il 70% dei valori sarà nel range [-1, 1], Il 25% [1,2] o [-2, -1] e il restante 5% [-3,-2] e [2,3]**
- **Con la riga sotto, abbiamo un valore casuale nel range [34,40] con la maggior parte dei valori nel range [36,38]**




```
System.out.println(37 + rand.nextGaussian()));
```

Random in un intervallo particolare 2/2

- **Per avere invece un valore che può essere al 50% un valore, al 30% un altro e infine al 20% un altro ancora, vi basta prendere un numero random da 0 a 10 e controllare in che intervallo di valori si trova**

Valori pseudo-casuali

Valori pseudo-casuali

- **I valori generati dai metodi visti fino ad ora sono detti pseudo-casuali in quanto sono prodotti da un algoritmo che genera sequenze approssimabili a dei valori casuali**
- **La sequenza di valori generati varia in base al *seed* specificato durante la creazione dell'oggetto *Random*, allo stesso seed corrispondono sempre gli stessi valori**
- **Se non viene specificato alcun seed la classe utilizza come riferimento l'orologio di sistema, così che il seed sia sempre diverso e di conseguenza anche i valori generati**

```
Random rand = new Random(42);

int randomValue1 = rand.nextInt();
int randomValue2 = rand.nextInt();

// randomValue1 == randomValue2
```


Time to make practice

 Esercizio da secondi a giorni, ore, minuti e secondi

- **Scrivete una funzione che dato in input un numero di secondi, restituisce una stringa che dice «Giorni: numero di giorni, Ore: numero di ore etc...»**

 Da dispari a pari

- **Scrivete una funzione che dato un array di valori ne restituisce una copia dove i numeri dispari sono stati moltiplicati per 2.**

Estrarre un valore da un dado a *n* facce

- **Non tutti i dadi hanno 6 facce! Per ogni evenienza vogliamo simulare il lancio di un dado con un numero arbitrario di facce**
- **Scrivete un programma che estrae casualmente un valore da un dado ad *n* facce (un intero compreso tra 1 e n)**

Suggerimento: inserite il codice della vostra soluzione all'interno del metodo *lanciaDado* del codice che trovate a questo indirizzo:

https://pastebin.com/j43Bgjhg

 Tabelline

- **Scrivete una funzione che dato in input due numeri n ed m, crea un array lungo m che in ogni cella ha il valore di indice della cella moltiplicato per n**

🛴 Esercizio Fibonacci

Scrivere un metodo che stampa l'ennesimo elemento della sequenza di Fibonacci, nella quale ogni numero è definito dalla somma dei due precedenti, eccetto i primi due che sono per definizione 0 e 1. Esempio: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... E così via.
Assumete di iniziare da 0, dunque fibonacci(0) = 0, fibonacci(1) = 1, ... e così via.

Suggerimento: inserite il codice della vostra soluzione all'interno del metodo *fibonacci* del codice che trovate a questo indirizzo:

https://pastebin.com/8ifNRBZH

Altro suggerimento: usate gli array

Test: fibonacci(45) deve stampare 1134903170

🛴 Esercizio minimo e massimo

Dato un array di interi (inseriti dall'utente, separati da virgola e senza spazi), effettuare la conversione dei valori in interi e stampare in ordine minimo e massimo (separati da virgola).

Gli interi inseriti possono anche assumere valori negativi.

Suggerimento: implementate i metodi *parseNumbers* e *computeMinAndMax* del codice che trovate a questo indirizzo:

https://pastebin.com/T6wFn6Cy

Altro suggerimento: per il parsing da String a int, potete usare il metodo *parseInt()* della classe Integer (e l'autoboxing)

Esercizio Anagrammi

Due stringhe a e b sono dette anagrammi se contengono gli stessi caratteri con le stesse frequenze. Ad esempio, gli anagrammi di CAT sono CAT, ACT, TAC, TCA, ATC, e CTA.
Date due stringhe a e b in input, stampare «anagrammi» se sono anagrammi (case-insensitive), «non anagrammi» altrimenti.

Suggerimento: inserite il codice della vostra soluzione all'interno del metodo *anagrams* del codice che trovate a questo indirizzo:

https://pastebin.com/ipiNQFGf

 Estrarre una stringa casuale da un array

- **A partire da un array di stringhe, es. [‘Andrea’, ‘Marco’, ‘Melvin’, …], vogliamo estrarre uno dei valori presenti in esso**
- **Scrivete un programma che dato un array di stringhe, estrae casualmente una delle stringhe**
- **Provate a implementare diverse soluzioni del programma, utilizzando i diversi metodi per estrarre un valore casuale e provando ad estrarre la stringa con una probabilità non omogenea, es. ‘Andrea’ viene estratto il 50% delle volte, ‘Marco’ il 20%, …**