LABORATORIO DI PROGRAMMAZIONE
CORSO DI LAUREA IN SICUREZZA DEI SISTEMI E DELLE RETI INFORMATICHE
UNIVERSITÀ DEGLI STUDI DI MILANO
2025–2026

Indice

Ultima revisione: 1 dicembre 2025.

Nota

Per svolgere gli esercizi in questa lezione dovete usare solo la programmazione imperativa. Java è un linguaggio di programmazione (principalmente) orientato agli oggetti, perciò è comunque necessario usare *qualche* costrutto della programmazione a oggetti. In particolare, dovete creare una *classe* che contiene il codice tramite il costrutto `public class <ClassName>`. Sostituite `<ClassName>` con il nome del file. Inoltre le definizioni delle funzioni devono essere preceduti da `public static`

.

Parte 1. **Array**

Esercizio 1

*Lettura di un array***.**

Tempo: 10 min.

Scrivere un programma che legga in input un array e ne mostra il contenuto in output. Come prima cosa, il programma chiede all'utente il numero di elementi di cui si compone l'array, poi legge gli elementi uno per uno, di tipo `int`. In seguito, mostrate in output il contenuto dell'array letto.

Esercizio 2

*Lettura di un array – versione con metodi***.**

Tempo: 10 min.

Modificate il programma nell'Esercizio 1, definendo un metodo `int[] readArray()` per la lettura dell'array, e un metodo `void printArray(int[] array)` che mostra il contenuto dell'array in output.

Esercizio 3

*Invertire un array***.**

Tempo: 10 min.

Scrivere un programma che legga in input un array e stampi in output l'array invertito. Usate il metodo `int[] readArray()` definito nell'Esercizio 2.

Ricordatevi di risolvere l'esercizio a partire dalle parti più importanti. Quindi, se la lettura dell'input vi mette in difficoltà, iniziate definendo il valore dell'array staticamente.

Esercizio 4

*Invertire un array***.**

Tempo: 10 min.

Modificate il programma nell'Esercizio 3, definendo un metodo `int[] reverse(int[])` per l'inversione dell'array. Il metodo non deve modificare l'array in input, ma deve ritornare un array nuovo.

Input. Un array `v[N]` di `int` o `float` o `double`.

Output. L'array `v[N]` riordinato (ordine non decrescente).

```
scambio = true;
while (scambio==true)
    scambio=false;
    for ( i=0; i<=N-2; i++ )
        if ( v[i]>v[i+1] )
            scambia v[i] e v[i+1]
            scambio=true;
```

FIGURA 1. Pseudocodice dell'algoritmo BubbleSort.

ESERCIZIO 5

Ordinare un array.

Tempo: 20 min.

Scrivete un programma che implementi l'algoritmo di ordinamento BubbleSort secondo lo pseudocodice riportato in Figura 1, definendo un metodo `void sortArray(int[] array)`. Il programma legge in input un array usando il metodo definito nell'Esercizio 2.

Parte 2. Stringhe

Documentazione

La documentazione ufficiale del JDK è disponibile all'indirizzo `https://docs.oracle.com/en/java/javase/25/docs/api/index.html`. Per conoscere le funzionalità relative alle stringhe, potete fare riferimento a `https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/String.html`.

Esercizi su stringhe

Come anticipato, è possibile risolvere (alcuni de) gli esercizi proposti usando la programmazione oggetti, ovvero invocando dei *metodi* sulle variabili di tipo `String`. Per prendere dimestichezza con il linguaggio, è consigliabile usare *prima* un approccio imperativo. Ad esempio, se dovete lavorare sugli elementi di una stringa uno per uno, potete usare il metodo `charAt(int)` che restituisce l'i-esimo carattere della stringa. Data una variabile `str` di tipo `String`, la sintassi per invocare il metodo è `str.charAt(0)`. Allo stesso modo, se dovete costruire una stringa a partire da un insieme di `char`, potete dichiarare un

array di tipo `char` e inserire gli elementi man mano. Per definire una variabile `String` a partire da `char[]` potete fare `String str2 = new String(array)`, dove `array` è un array di tipo `char`.

ESERCIZIO 6

Trasformazione in caratteri minuscoli.

Tempo: 10 min.

Scrivere un programma Java che legga in input da tastiera una stringa e restituisca in output la stringa ottenuta trasformando la stringa in caratteri maiuscoli.

ESERCIZIO 7

Accesso posizionale.

Tempo: 10 min.

Scrivere un programma che crei un oggetto di tipo `String`, contenente il proprio nome, e stampi il primo e l'ultimo carattere della stringa. Descrivere ciò che avviene in memoria durante l'esecuzione del programma, aiutandosi con un disegno.

ESERCIZIO 8

Semplici operazioni – 1.

Tempo: 10 min.

Scrivere un programma che riceva una stringa in input e stampi la stringa ottenuta invertendo la prima e l'ultima lettera.

ESERCIZIO 9

Semplici operazioni – 2.

Tempo: 10 min.

Scrivere un programma che riceva una stringa in input e stampi la stringa invertita.

ESERCIZIO 10

Stringa palindroma.

Tempo: 15 min.

Scrivere un programma che riceva una stringa in input e stampi `Palindroma`, se la stringa è palindroma, `Non palindroma`, altrimenti. Definite un metodo `boolean isPalindroma(String)` per lo scopo.

Parte 3. ArrayList

ESERCIZIO 11

Massimo e minimo.

Tempo: 10 min.

Scrivete un programma che legge in input una sequenza di numeri interi, fino a che l'utente non inserisce un valore specifico, definito da voi a tempo di compilazione. Il programma calcola il minimo e il massimo di questa sequenza. Implementate una versione *senza* memorizzare l'intera sequenza, e una versione in cui usate la struttura dati più adatti per memorizzare l'intera sequenza, e il calcolo del massimo (e minimo, resp.) è implementato in un metodo chiamato `getMax` (`getMin`, resp.).

Esercizio 12

Frequenze di parole.

Tempo: 25 min.

Scrivete un programma che legge in input una stringa contente una frase e calcola il numero di occorrenze di ciascuna parola nella frase, indipendentemente da maiuscole e minuscole. Non dovete considerare spazi e segni di punteggiatura. Suddividete il codice in funzioni, e usate delle strutture dati dinamiche.

Fate riferimento alla documentazione online per capire i metodi necessari per operare sulle stringhe. Avete una stringa che comprende l'intera frase, e dovete spezzarla in parole, quindi dividere la stringa di partenza dove ci sono degli spazi.

Non potete usare classi *collection* eccetto liste dinamiche. È meglio una lista o un array?