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

Indice

Tratti dagli esercizi del corso del prof. Vincenzo Marra.
Ultima revisione: 20 novembre 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.

Parte 1. Algoritmo Euclideo

Esercizio 1

L'algoritmo euclideo: prima versione.

Tempo: 20 min.

La Figura 1 mostra un'implementazione parziale dell'algoritmo euclideo delle sottrazioni successive per il calcolo del massimo comun divisore di due numeri naturali. Completatela.

Euclide.java

```
1  /*********************************************************
2   * Calcola il m.c.d di due interi a, b > 0 applicando    *
3   * l'algoritmo delle sottrazioni successive di Euclide. *
4   *********************************************************/
5
6  public class Euclide {
7
8          public static void main(String []args){
9                  int a=214, b=128; //Deve essere a,b > 0
10                 System.out.printf("Il m.c.d. di %d e %d e': ",a, b);
11
12                 while (a != b)   //Fino a quando a e b sono diversi...
13                 if (a > b)   //Se a > b,
14                         ....     //sostituisci a con a-b.
15                 else         //Altrimenti,
16                         ....     //sostituisci b con b-a.
17
18                 System.out.printf("%d\n", a);
19         }
20
21 }
22
23
24
```

Figura 1. L'algoritmo euclideo delle sottrazioni successive in Java.

Esercizio 2

L'algoritmo euclideo: validazione dell'input.

Tempo: 20 min.

Modificate la vostra soluzione all'Esercizio 1 in modo che il programma si assicuri che i valori di `a` e `b` inseriti dall'utente soddisfino le precondizioni: $a, b > 0$. Al termine dell'esecuzione, mostrate il numero di iterazioni necessarie per arrivare alla soluzione.

Parte 2. Verifica di identità

Tre sommatorie facili

Sia dato un intero $n \geqslant 1$. Valgono le tre identità:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{1}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \tag{2}$$

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4} \tag{3}$$

Esercizio 3

Somma dei primi n numeri naturali.

Tempo: 10 min.

Scrivere un programma che chieda in ingresso all'utente un intero $n \geqslant 1$. Se l'intero inserito non soddisfa questa disuguaglianza, il programma chiede il reinserimento fino a che la condizione non sia verificata. Il programma visualizza poi la somma dei primi n numeri naturali calcolata con un ciclo `for`, assieme alla stessa somma calcolata tramite la formula al membro destro di (1).

Esercizio 4

Somma dei primi n quadrati.

Tempo: 10 min.

Scrivere un programma che chieda in ingresso all'utente un intero $n \geqslant 1$. Se l'intero inserito non soddisfa questa disuguaglianza, il programma chiede il reinserimento fino a che la condizione non sia verificata. Il programma visualizza poi la somma dei primi n quadrati calcolata con un ciclo `for`, assieme alla stessa somma calcolata tramite la formula al membro destro di (2).

Esercizio 5

Somma dei primi n cubi.

Tempo: 10 min.

Scrivere un programma che chieda in ingresso all'utente un intero $n \geqslant 1$. Se l'intero inserito non soddisfa questa disuguaglianza, il programma chiede il reinserimento fino a che la condizione non sia verificata. Il programma visualizza poi la somma dei primi n cubi calcolata con un ciclo `for`, assieme alla stessa somma calcolata tramite la formula al membro destro di (3).

Parte 3. Calcolatrice

Esercizio 6

Calcolatrice con menu.

Tempo: 30 min.

Per la lettura di un dato di tipo `double` dovete usare il metodo `nextDouble()` della classe `Scanner`.

Scrivete un programma che chieda all'utente di inserire due numeri (operandi) di tipo `double`. Il programma visualizza poi il menu seguente.

```
1. Addizione.
2. Sottrazione.
3. Moltiplicazione.
4. Divisione.
5. Esci.
>
```

dove `>` è un "prompt" che indica all'utente che la macchina è in attesa dell'input dell'utente. L'utente inserisce una scelta, che il programma acquisisce come dato di tipo `int`. Se la scelta è inesistente o sbagliata, il programma stampa un messaggio d'errore, e visualizza nuovamente il menu. Se la scelta è `5`, il programma termina. Se la scelta è `1`, `2` o `3`, il programma stampa il risultato dell'operazione corrispondente applicata agli operandi, e visualizza nuovamente il menu. Se la scelta è `4` e il divisore è nullo, il programma stampa un messaggio d'errore, e visualizza nuovamente il menu; se il divisore non è nullo, stampa il risultato della divisione applicata agli operandi, e visualizza nuovamente il menu. Ogni operazione dev'essere implementata tramite un metodo. Per esempio, nell'eseguire una somma la procedura `main` invocherà un metodo di prototipo `public static double somma(double,double)`.

Esercizio 7

Calcolatrice con menu e modifica degli operandi.

Tempo: 15 min.

Migliorate la calcolatrice scritta per l'Esercizio 6 in modo che l'utente possa modificare il valore degli operandi tramite il menu. In dettaglio, aggiungete al menu la voce:

```
0. Inserisci operandi.
```

ed eliminate la lettura iniziale degli operandi — il programma visualizzerà menu e prompt direttamente all'avvio. Se l'utente sceglie `0`, il programma richiede l'inserimento dei due operandi, e da quel punto in poi il programma eseguirà le operazioni richieste sui due dati immessi, fino a quando l'utente non si avvarrà nuovamente dell'opzione `0`. Se l'utente sceglie di eseguire una qualunque delle quattro operazioni prima di aver inserito gli operandi, il programma visualizza un messaggio d'errore, e visualizza nuovamente il menu. Ogni operazione dev'essere eseguita tramite metodi.

Dipartimento di Informatica, Università degli Studi di Milano,