Istituto Statale "E.Torricelli"
Liceo Scientifico Tecnologico-Tecnico Industriale

a.s. 2025/2026

Compiti Estivi
Scienze e Tecnologie Applicate

classi 2 ITI

riferimenti al materiale utilizzato nel corso dell'anno

ESERCIZI DI RINFORZO PER LA PROGRAMMAZIONE JAVA

Il percorso è suddiviso in aree tematiche incrementali. Ogni esercizio richiede una triplice richiesta: **Diagramma di flusso (Flow-chart)**, **Pseudocodice** e **Codice Java** commentato.

AREA 1: Strutture di Controllo e Logica Iterativa

*Questa sezione si concentra sui costrutti fondamentali della programmazione imperativa (*if-else*,* while*,* do-while*), sull'uso di accumulatori, contatori e flag di terminazione (sentinelle).*

Esercizio 1: Menu Ciclico e Analisi Statistica di "N" Valori

- Sviluppare un programma Java che presenti un menu interattivo continuo utilizzando un ciclo do-while e una struttura switch. Il menu deve proporre due opzioni:
  1. **Analisi Numerica**: Chiede all'utente quanti numeri desidera inserire ("N"). Attraverso un ciclo, acquisisce i valori calcolando e visualizzando in tempo reale il valore **massimo**, il valore **minimo** e la **media aritmetica** di tutti i numeri digitati.
  2. **Uscita**: Termina l'esecuzione del programma.

  - Il menu deve ripresentarsi automaticamente al termine delle operazioni della prima opzione e ciclare finché l'utente non seleziona esplicitamente l'uscita.

Esercizio 2: Somma e Prodotto con Condizioni di Terminazione Alternata

- Scrivere un programma che permetta l'inserimento continuo di numeri interi da parte dell'utente, offrendo due modalità distinte di esecuzione:
  - **Modalità Somma**: Il programma accumula i valori e si interrompe non appena l'utente digita il valore 0 (zero), mostrando il risultato finale.
  - **Modalità Prodotto**: Il programma moltiplica i numeri digitati. Gli studenti dovranno individuare il valore sentinella ideale per segnalare la fine della digitazione (es. -1 o un valore specifico) e mostrare il prodotto totale, prestando particolare attenzione al valore di inizializzazione dell'accumulatore per non annullare le moltiplicazioni.

Esercizio 3: Calcolo di Prodotto e Potenza tramite Somme e Prodotti Ripetuti

- Realizzare un algoritmo che implementi le seguenti funzioni matematiche senza utilizzare i metodi preconfezionati della libreria Math, ma basandosi esclusivamente su cicli e operatori elementari:
  - **Moltiplicazione**: Dati due numeri interi digitati da tastiera, calcolare il loro prodotto utilizzando il solo operatore di somma (+) all'interno di una struttura iterativa.
  - **Potenza**: Dati una base e un esponente intero digitati dall'utente, calcolare la potenza tramite prodotti ripetuti. Gli studenti dovranno gestire accuratamente i casi particolari matematici (esponente uguale a zero, base uguale a zero).

Esercizio 4: Divisibilità, Parità e Calcolo del Resto Condizionato

- Sviluppare un algoritmo che acquisisca due numeri interi da tastiera e verifichi:

1. Se ciascun numero sia pari o dispari.
2. Se i due numeri siano fra loro divisibili (se il primo è multiplo del secondo o viceversa).
3. Successivamente, il programma deve calcolare e visualizzare il resto della divisione tra i due numeri, ma **solo a condizione che siano entrambi pari o entrambi dispari**. Se una coppia è mista (un pari e un dispari), il programma deve segnalare l'impossibilità di procedere con il calcolo del resto.

AREA 2: Validazione dei Dati e Logica di Controllo Avanzata

Questa sezione allena gli studenti alla robustezza del codice, alla gestione dei limiti di inserimento (tentativi) e all'applicazione di regole logico-matematiche complesse per la validazione degli input.

Esercizio 5: Controllo Accesso con Tentativi Limitati

- Simulare la verifica di correttezza di un codice di identificazione (PIN) numerico. Il programma deve richiedere il codice all'utente e confrontarlo con una costante predefinita nel codice.
- Se il codice è corretto, visualizza un messaggio di successo e termina.
- Se il codice è errato, permette all'utente di riprovare, consentendo un **massimo di 3 tentativi**. Superato il terzo tentativo errato, il programma deve bloccarsi visualizzando un messaggio di allarme (es. "Account Bloccato").

Esercizio 6: Validatore di Date e Controllo Anno Bisestile

- Scrivere un programma che riceva in input tre valori interi distinti: giorno, mese e anno. L'algoritmo deve verificare l'attendibilità e la correttezza della data fornita, implementando i controlli per:
  - I mesi con 30 e 31 giorni.
  - Il mese di febbraio, determinando se l'anno inserito sia bisestile (un anno è bisestile se è divisibile per 4 ma non per 100, oppure se è divisibile per 400).
  - Il programma restituirà in output un messaggio di conferma ("Data valida") o di errore ("Data non valida").

Esercizio 7: Assegnazione della Stagione di Nascita

- Acquisita da tastiera una data di nascita valida (espressa in giorno e mese), l'algoritmo deve indicare in quale stagione astronomica è nato l'utente, gestendo con precisione i giorni di transizione (i solstizi e gli equinozi):
  - Primavera: dal 21 marzo al 20 giugno
  - Estate: dal 21 giugno al 22 settembre
  - Autunno: dal 23 settembre al 21 dicembre
  - Inverno: dal 22 dicembre al 20 marzo

AREA 3: Risoluzione di Equazioni e Radici Intere

Esercizi orientati alla corretta formalizzazione delle formule matematiche e alla gestione dei domini numerici nel software.

Esercizio 8: Radice Quadrata Intera

- Sviluppare un algoritmo che calcoli e visualizzi la radice quadrata intera (approssimata per difetto) di un numero digitato dall'utente, senza usare funzioni di libreria. Il programma deve applicare rigidamente le regole sui radicali nel sistema dei numeri reali, impedendo l'introduzione di un radicando negativo e richiedendo l'input fino a che non viene inserito un valore valido.

Esercizio 9: Risoluzione di Equazioni di Primo e Secondo Grado

- Creare un programma che permetta di risolvere i modelli algebrici lineari e quadratici:
  - **Primo Grado ("ax + b = 0")**: Acquisiti i coefficienti "a" e "b", calcolare e visualizzare la radice dell'equazione, gestendo i casi di equazione indeterminata ("0x = 0") o impossibile ("0x = b").
  - **Secondo Grado ("ax^2 + bx + c = 0")**: Acquisiti i coefficienti "a", "b" e "c", calcolare il discriminante ("\Delta"). In base al valore del delta, determinare e visualizzare le due radici reali e distinte, le due radici reali e coincidenti, o segnalare l'assenza di radici reali (delta negativo).

AREA 4: Teoria, Architettura e Strumenti (Esercizio di Tracciamento)

Una traccia pensata consolidare la comprensione dell'allocazione in memoria (Stack e Heap) e il funzionamento dei comandi da riga di comando indicati nella dispensa e a lezione.

Analisi del Codice e Tabella di Traccia (Trace Table)

- Dato il seguente frammento di codice Java conforme agli standard indicati a lezione:
  - commentarlo in modo significativo
  - disegnare la relativa **Tabella di Traccia**, ipotizzando che l'utente digiti in sequenza i valori: 4, 7, 2, 0.

```
1.  import java.io.*;
2.
3.  public class Tracciamento {
4.      public static void main(String[] args) throws IOException {
5.          InputStreamReader A = new InputStreamReader(System.in);
6.          BufferedReader T = new BufferedReader(A);
7.
8.          int contatore = 0;
9.          int somma = 0;
10.         int numero;
11.
12.         do {
13.             System.out.print("Inserisci un numero (0 per uscire): ");
14.             String num=T. readLine();
15.             numero = Integer.valueof(num).intValue();
16.
17.             if (numero % 2 == 0 && numero != 0) {
18.                 somma += numero;
19.                 contatore++;
20.             }
21.         } while (numero != 0);
22.
23.         System.out.println("Conteggio pari: " + contatore);
24.         System.out.println("Somma complessiva: " + somma);
25.     }
26. }
```

- *Quesito teorico correlato:* Descrivere in quale area di memoria (Stack o Heap) viene allocata la variabile primitiva numero e dove viene allocato l'oggetto puntato dalla variabile reference T. Spiegare inoltre a cosa serve il comando javac Tracciamento.java eseguibile da riga di comando.

DOMANDE TEORICHE

Di seguito una raccolta mirata di 20 domande teoriche a risposta aperta, formulate per verificare la comprensione concettuale profonda degli argomenti trattati a lezione in merito all'architettura della memoria, al ciclo di vita del software Java e alla logica di programmazione imperativa.

Queste tracce richiedono risposte accurate e argomentate: ti esorto a motivare il funzionamento strutturale delle istruzioni apprese a lezione anziché limitarti alla sola memorizzazione della sintassi

Gruppo A: Architettura della Memoria e Gestione dei Dati (Stack vs Heap)

1. **La distinzione dei tipi in memoria** Spiega in quale area della memoria centrale (Stack o Heap) viene allocata la variabile primitiva int numero e in quale area viene invece allocata la struttura creata da new InputStreamReader(System.in). Qual è la differenza fondamentale tra queste due aree?
2. **Il ciclo di vita degli oggetti e il Garbage Collector** Cosa accade nella memoria Heap quando un oggetto non possiede più alcuna variabile reference che lo punti? Descrivi il ruolo e il funzionamento del *Garbage Collector*.
3. **Variabili primitive vs Variabili reference** Qual è la differenza fondamentale tra il contenuto memorizzato in una variabile di tipo primitivo (come double) e il contenuto di una variabile reference? Aiutati con un esempio concreto basato sul codice.
4. **Il ciclo dei sotto-processi (Thread)** Quando un programma Java viene eseguito, si parla di "processo". Quali sono i due sotto-processi (thread) principali che si attivano contemporaneamente e quali compiti distinti svolgono?

Gruppo B: Ciclo di Vita del Software e Compilazione

5. **Dal codice sorgente all'esecuzione** Descrivi dettagliatamente cosa accade quando da riga di comando si esegue il comando javac Esercizio.java e cosa succede quando successivamente si digita java Esercizio. Quali file vengono coinvolti o generati?
6. **Il concetto di Bytecode e portabilità** Che cos'è il *Bytecode*? Per quale motivo l'architettura di Java, basata sulla Java Virtual Machine (JVM), rende i programmi "indipendenti dalla piattaforma" (portabili)?
7. **Errori di compilazione vs Errori di runtime** Spiega la differenza tra un errore intercettato dal compilatore (javac) e un errore che si manifesta esclusivamente durante l'esecuzione del programma (runtime), portando un esempio per ciascuna tipologia.

Gruppo C: Analisi Strutturale dell'Input con BufferedReader

8. **Anatomia dell'istruzione di input** Analizza l'istruzione numero = Integer.valueOf(T.readLine()).intValue(); spiegando accuratamente la funzione e l'ordine di esecuzione di ciascuna delle tre sotto-operazioni: T.readLine(), Integer.valueOf(...) e .intValue().
9. **Il ruolo delle classi involucro (Wrapper)** A cosa servono le classi involucro come Integer o Double all'interno di un linguaggio orientato agli oggetti come Java? Perché non possiamo usare direttamente il tipo primitivo int per invocare metodi di conversione del testo?
10. **La gestione obbligatoria delle eccezioni (**IOException**)** Per quale motivo l'utilizzo del metodo T.readLine() impone al programmatore di inserire la clausola throws IOException nella firma del metodo main? Cosa rappresenta questa eccezione a livello logico?
11. **Analisi del rischio:** NumberFormatException Se l'utente esegue l'istruzione numero = Integer.valueOf(T.readLine()).intValue(); ma digita da tastiera la parola "quindici" invece del valore numerico 15, cosa succede al programma a runtime? Come si chiama l'anomalia sollevata?

Gruppo D: Logica Algoritmica e Strutture di Controllo

12. **Il Teorema di Jacopini-Böhm** Enuncia il principio cardine del Teorema di Jacopini-Böhm. Quali sono le tre sole strutture di controllo necessarie e sufficienti per rappresentare qualsiasi algoritmo computabile?
13. **Cicli pre-condizionali vs Cicli post-condizionali** Metti a confronto la struttura iterativa while con la struttura do-while. In quale scenario applicativo sei obbligato a scegliere il do-while per garantire la corretta interazione con l'utente?

14. **Inizializzazione degli accumulatori (Somma vs Prodotto)** Perché la variabile utilizzata come accumulatore per calcolare una somma totale viene inizializzata a 0, mentre l'accumulatore per un prodotto progressivo deve essere necessariamente inizializzato a 1? Cosa accadrebbe se si invertissero i valori?
15. **Il concetto di "Valore Sentinella"** Che cos'è un valore sentinella all'interno di un ciclo iterativo? Porta l'esempio di un algoritmo in cui lo 0 agisce da sentinella e spiega come evitare che la sentinella stessa alteri il risultato finale del calcolo.
16. **L'importanza dell'operatore modulo (**%**)** Spiega l'utilità algoritmica dell'operatore modulo (%) nella risoluzione di problemi informatici. In che modo viene impiegato per verificare sia la parità di un numero, sia la divisibilità reciproca tra due valori?
17. **Evitare il ciclo infinito (Loop)** Qual è la condizione logica ed operativa affinché un ciclo while o do-while giunga a terminazione? Cosa deve accadere tassativamente alle variabili di controllo interne al blocco di istruzioni?

Gruppo E: Logica Booleana e Strutture di Selezione

18. **Tabelle di verità e cortocircuito logico** Data l'espressione condizionale if (numero % 2 == 0 && numero != 0), spiega come si comporta l'operatore logico && (AND). Cosa si intende per "valutazione in cortocircuito" se la prima condizione risulta falsa?
19. **Selezione nidificata vs Selezione multipla (**switch**)** In quali casi specifici è preferibile e strutturalmente più elegante utilizzare l'istruzione switch al posto di una lunga serie di costrutti if-else if-else nidificati? Quali limitazioni possiede lo switch riguardo ai tipi di dato valutabili?
20. **Validazione dei dati e domini matematici** Perché nell'algoritmo di risoluzione di un'equazione di secondo grado o nel calcolo di una radice quadrata la validazione dei dati di input (es. il controllo del segno del radicando o del discriminante (Delta) deve precedere qualsiasi operazione di calcolo reale? Quale principio di robustezza del software esprime questa pratica?