Unità A1 – I file testo	**MODULO L3**

(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- *stream*
- file testo
- file binari
- file di tipi primitivi
- bufferizzazione
- **InputStreamReader**
- **OutputStreamWriter**
- **Reader**
- **Writer**
- **FileReader**
- **FileWriter**
- **read()**
- **write** (**char** c)
- **close ()**
- **BufferedReader**
- **BufferedWriter**
- **String readLine***()*
- **write (String** s)
- **void newLine()**
- **void close()**

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Cosa vuol dire *processo produttore* e *processo consumatore* e come intergaiscono?
2. Cosa si intende con *stream*?
3. Quali *tipi di file* consente di usare Java?
4. Quali sono le classi degli *stream standard*?
5. Quali sono le classi degli *stream su disco*?
6. In cosa consiste e quali vantaggi presenta la *bufferizzazione*?

B2) Competenza

1. Quali sono i metodi degli *stream di testo standard*?
2. Quali sono i metodi degli *stream di testo bufferizzati*?
3. Qual è lo schema di un'applicazione che *legge dati da un file*?
4. Qual è lo schema di un'applicazione che *scrive dati da un file*?

(C) ESERCIZI DI COMPRENSIONE

1. I file sono trattati, in tutti i moderni linguaggi di programmazione, con il nome astratto di ……, che significa flusso. Si possono avere stream di ……. e stream di ……. Lo stream è visto come un ……. di comunicazione tra il processo …….. e il processo ……
2. Uno stream di ……. è un canale che trasferisce dati dalla sorgente (un organo di …….) al programma; uno stream di …… è un canale che trasferisce dati dal programma al destinatario (un organo di …….).
3. Uno stream di testo è un flusso formato da sequenze di …… Può essere stream di testo ……., se riguarda organi di input e output (…….. e ……), oppure stream di testo su disco, se tratta dati su memoria ……
4. Le classi principali sono ……. e …….. . .Dalla classe …… derivano le classi …… e quella bufferizzata ……, per le operazioni di lettura; dalla classe …….. derivano le classi …… e quella bufferizzata ……
5. La bufferizzazione è utile perché consente di leggere una ……. di caratteri anziché un solo carattere, semplificando la struttura del programma. Quando si usa la bufferizzazione, occorre …… la classe relativa allo ……. con quella relativa al …….
6. Associare le proposizioni di sinistra con le classi corrispondenti sulla destra:

1		Classe per stream standard di input	**A**	**FileReader**
2		Classe per stream di input su disco	**B**	**InputStreamReader**
3		Classe per stream standard di output	**C**	**BufferedReader**
4		Classe per bufferizzazione di input	**D**	**OutputStreamWriter**
5		Classe per stream di output su disco	**E**	**FileWriter**
6		Classe per bufferizzazione di output	**F**	**BufferedWriter**

7. Completare la tabella con le classi opportune.

	Input	**Output**
Stream standard		
Stream su disco		
Stream per bufferizzazione		

8. Per ciascuna delle proposizioni riportate, indicare se vera o falsa.

	Vero	Falso
InputStreamReader utilizza lo *stream* standard di input		
FileReader utilizza lo *stream* di input su disco		
FileReader utilizza lo *stream* di input su disco		
OutputStreamWriter utilizza lo *stream* standard di input		
BufferedReader si usa solo con gli *stream* standard		
FileWriter utilizza lo *stream* di input su disco		
BufferedWriter esegue la scrittura bufferizzata su disco		

9. Completare la tabella con i prototipi dei metodi relativi alle classi degli *stream* standard.

	Sintassi	Descrizione
Lettura		
Scrittura		
Chiusura		

10. Completare la tabella con i prototipi dei metodi relativi alle classi degli *stream* su disco.

	Sintassi	Descrizione
Lettura		
Scrittura		
Chiusura		

11. Completare la tabella con i prototipi dei metodi relativi alle classi di bufferizzazione:

	Sintassi	Descrizione
Lettura		
Scrittura		
Chiusura		

12. Disegnare la gerarchia di classi che eredita da **Object** le sottoclassi **Reader** e **Writer** con le relative sottoclassi

(D) ESERCIZI DI APPLICAZIONE

1. Scrivere una applicazione Java a menu, che consenta di registrare alcuni dati dei clienti che devono ancora saldare il debito e di visualizzare l'elenco completo dei dati presenti nell'archivio. Si supponga che i dati necessari per i clienti siano cogonme, nome e importo fattura.
2. Dato un file testo *elenco.txt*, contenente righe di testo, stamparlo a video, con interlinea doppia.
3. Creare un'applicazione *rubrica.java* che consenta di memorizzare in un file testo *rubrica.dat* i dati di un certo numero di persone. Prevedere le seguenti classi:
   Nominativo
   – attributi: *cognome*, *nome* e *telefono*;
   – metodi:
     - costruttori
     - *getNominativo*()
     - *setNominativo()*
     - *leggi*() (lettura da input)

   Rubrica:
   – attributi: *file di input* e *file di output*;
   – metodi:
     - costruttore
     - *apriScrittura*()
     - *apriLettura*()
     - *elenco*() (stampa elenco nominativi)
     - *inserisci*() (registra un nominativo)
     - *ricerca()* (ricerca di un mominativo)

   Analizzare lo scopo dei vari metodi e scegliere per ciascuno un prototipo adatto.
2. Scrivere un'applicazione che registri in un file testo *voti.txt* una serie di voti interi (da 1 a 10) di studenti, terminata con uno zero. Successivamente, deve essere stampata a video la sola sequenza dei voti maggiori o uguali a 6 e la loro media.
3. Scrivere la stessa applicazione dell'esercizio precedente, ma registrando i dati in un file binario. Osservare l'occupazione in byte su disco. Quali considerazioni si possono trarre?
4. Scrivere un'applicazione per creare un file *canzoni.txt* contenente i dati di canzoni, come autore, titolo, anno di edizione, luogo di edizione, nome del produttore, nome dell'interprete, durata.

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 1

Titolo: Implementazione classe TextFile
Obiettivi: utilizzo classi **FileWriter**, **InputStreamReader** e metodi relativi

1) Predisporre mediante il **Blocco note** di Windows o altro editor un file contenente il seguente testo:

   "Quando puoi misurare cio' di cui parli, ed esprimerlo in numeri, ne hai una certa conoscenza; quando non puoi misurarlo, quando non puoi esprimerlo in numeri, la tua conoscenza e' insoddisfacente".
   (Lord Kelvin)
   (*inserire una riga vuota*)

   Salvare il testo nel file *testo1.txt*.
2) Attivare l'ambiente di sviluppo (TextPad, Eclipse, ecc)
3) Creare una classe *TextFile* con attributo stringa *fileName*
4) Aggiungere alla classe i seguenti metodi, verificandone, volta per volta, il corretto funzionamento su un oggetto *tf* di classe *TextFile*:
   a. *ContaCaratteri()*, un metodo restituisce il numero di caratteri presenti in *testo1.txt*. Verificare il valore ottenuto in stampa con l'occupazione del file visualizzabile mediante l'opzione **File → Proprietà**;
   b. *AccodaFile()* che, aprendo in accodamento il file *testo1.txt*, consenta di accodare altro testo. Testare il funzionamento del metodo, inserendo, in esecuzione, il seguente testo:

      "Come va che la matematica essendo fondamentalmente un prodotto del pensiero umano indipendente dalla esperienza spiega in modo così ammirevole le cose reali?" (A. Einstein)

      e verificare, aprendolo con il **Blocco note**, che il file *testo1.txt* contenga entrambe le massime inserite; eseguire di nuovo l'applicazione *ContaCaratteri* e verificare l'occupazione del file anche mediante **File → Proprietà**;
   c. *ContaRighe()*, un metodo che conta il numero di righe del file, tenendo conto che il carattere di nuova riga è '\n';
   d. *CopiaFile(***String** *fname)*, un metodo che copia, carattere per carattere, il file *testo1.txt* nel file di nome *fname*.
   e. *getFileName()*, metodo che restituisce il valore dell'attributo *fileName*;
   f. *setFileName(***String** *fname)*, che imposta l'attributo *fileName* con la stringa *fname*.

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 2

Titolo: Caricare un testo in un file testo e accodare ad esso un altro testo letto da input.
Obiettivo: utilizzo classi **FileWriter**, **InputStreamReader**, **BufferedReader** e metodi relativi

1) Predisporre mediante il **Blocco note** di Windows o altro editor un file contenente il seguente testo:

```
"Il fatto che le macchine siano elettriche e' solo perche' con
l'elettricita' si possono inviare piu' rapidamente le informazioni. Le
analogie tra elaboratore e mente umana sono soprattutto analogie
matematiche di funzionamento".                                  (A.
Turing)
  (inserire una riga vuota)
```

Salvare il testo nel file *testo2.txt*.

2) Procedere analogamente a quanto fatto nell'Esercitazione 1, creando però una classe *BufferedTextFile* con attributo stringa *fileName* ed aggiungendovi gli stessi metodi della classe *TextFile*, verificandone, volta per volta, il corretto funzionamento su un oggetto *btf* di classe *BufferedTextFile*. I metodi vanno costruiti mediante la tecnica della *bufferizzazione*. Per il metodo *AccodaFile()* il testo da accodare è il seguente:

```
Un giorno le macchine riusciranno a risolvere tutti i problemi, ma mai
nessuna di esse potra' porne uno.
                                        (Albert Einstein)
```

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 3

Titolo: Scrivere un'applicazione *scriviNumeri.java* che registri in un file testo, *numeri.txt*, gestito a caratteri, una sequenza di numeri immessi da input e terminata da uno zero. Successivamente, scrivere un'altra applicazione *leggiNumeri.java* che riaprendo il file *numeri.txt* rilegga i singoli caratteri memorizzati e ricostruisca i numeri immessi, calcolandone la somma.
Obiettni: utilizzo classi **FileWriter**, **PrintWriter InputStreamReader**, **BufferedReader,** e metodi relativi

Creazione *scriviNumeri.java*

1. Aprire l'ambiente di sviluppo e creare l'applicazione *scriviNumeri.java*.
2. Creare l'oggetto *tastiera* mediante **InputStreamReader** e **BufferedReader**;
3. Creare, dentro un blocco **try-catch** lo stream per scrivere sul file con:
   fw = **new FileWriter** ("numeri.txt", **true**); // apre file in scrittura
   f = **new PrintWriter**(fw); // crea oggetto per scrivere
4. Scrivere un ciclo precondizionato che legge i singoli valori *num* da tastiera e li scrive nel f(num);
5. Dopo il ciclo chiudere lo stream *f*.
6. Eseguire l'applicazione e verificare tramite un editor qualunque, la corretta registrazione dei valori immessi.

Creazione *leggiNumeri.java*

1. Aprire l'ambiente di sviluppo e creare l'applicazione *leggiNumeri.java*.
2. Creare, dentro un blocco **try-catch** lo stream per leggere dal file con:
   fr = **new FileReader** ("numeri.txt"); // apre file in lettura
3. Azzerare l'accumulatore intero *somma*;
4. Azzerare una variabile *numero* in cui costruire il numero dai caratteri letti via via;
5. Prevedere un ciclo precondizionato di lettura che termini con -1 nel quale si legge il singolo carattere *c*; nel ciclo, s c vale 13 (carattere INVIO) si incrementa *somma* con *numero*, altrimenti si incrementa *numero* con il valore del carattere *c* convertito in cifra secondo le potenze di 10.
6. Eseguire l'applicazione e verificare la correttezza del valore della somma dei numeri presenti nel file.