Unità B2 – Accesso diretto	**MODULO L1**

(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- *Token*
- **StringBuffer**
- **RandomAccessFile**
- **StringTokenizer**
- Accesso diretto

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Cosa consente *l'accesso diretto ad un file*?
2. Quali sono le classi Java per la gestione *dell'accesso diretto ad un file*?
3. Quali classi e quali metodi si possono utilizzare per realizzare *record di lunghezza fissa*?
4. Cosa significa *cancellazione logicai?*

B2) Competenza

1. Su *quali procedure* occorre intervenire, per modificare un'applicazione da accesso sequenziale ad *accesso diretto*?
2. Quali sono le *classi utilizzate* da un'applicazione che gestisce file con la tecnica di accesso diretto?
3. Perché, per gestire correttamente l'accesso diretto, occorre trattare *record di lunghezza fissa*?

(C) ESERCIZI DI COMPRENSIONE

1. Un'applicazione che gestisce un file con accesso …….. tratta record di lunghezza …….; in questo modo, il sistema operativo può calcolare la posizione …… di un record conoscendone la posizione ……
2. Per l'accesso diretto, Java utilizza la classe …….. che consente di aprire uno stream in …….., in …….. o in …….. e …….. Con l'accesso diretto è opportuno che ogni …….. sia caratterizzato da un …….. progressivo rispetto al quale il file risulti ……..
3. La cancellazione ……... consiste nel modificare un ……. del record in modo da renderlo invisibile alle successive operazioni di …….. e ……. La cancellazione …….. evita l'uso di un file …… in cui copiare temporaneamente i dati dell'archivio.
4. La classe …… consente di manipolare stringhe e viene utilizzata per creare record a lunghezza ……. .
5. La classe …….. consente di scomporre una stringa in piccoli blocchi autonomi detti …….
6. Scrivere l'algoritmo corrispondente al metodo **public void** inserimento (**int** c):

7. Scrivere l'algoritmo corrispondente al metodo **public** Libro ricerca (**int** cod)

8. Scrivere l'algoritmo corrispondente al metodo **public void** eliminazione (**int** c)

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

	Vero	Falso
Il metodo **append**(**String** s) appartiene alla classe **StringBuffer**		
Il metodo **nextToken**(**String** s) appartiene alla classe **StringBuffer**		
La classe **StringBuffer** ha gli stessi metodi della classe **String**		
Il metodo **read()** della classe **RandomAccessFile** legge un carattere		
Con **write**(**byte** b[]) si scrive sul file un array di byte		
Il metodo **seek** (……) riceve un argomento **int**		

10. Completare la tabella con i prototipi opportuni dei metodi della classe **RandomAccessFile**:

Tipo di dato	Input	Output
byte		
Intero		
reale		
carattere		
array di byte		
stringa		

11. Completare la tabella con i prototipi opportuni dei metodi della classe **StringBuffer**.

	Prototipo del metodo
Costruttore	
Accodamento stringa	
Cancellazione di una sottostringa	
Conversione a **String**	

12. Completare la tabella con i prototipi opportuni dei metodi della classe **StringTokenizer**.

	Prototipo del metodo
Costruttore	
Successivo *token*	

13. Completare le parti mancanti nel seguente listato, che inserisce in un file un oggetto di classe Libro.

```
public void inserimento (int cod)
{Libro l=new Libro();
 try
   {        RandomAccessFile fout =new RandomAccessFile(filename, "…..");  ;
            aut=JOptionPane.showInputDialog("……..: ");
            edit=JOptionPane.showInputDialog("……: ");
            tit=JOptionPane.showInputDialog("……..: ");
            pr=Double.parseDouble(JOptionPane.showInputDialog("……: "));
            Libro l = new Libro (……);
```

```
            scriviRec (…….);
            fout……..();
    }
    catch (IOException e)
      {     System.out.println("Errore in inserimento!");   }
    }
```

14. Completare le parti mancanti nel seguente listato, che elimina da un file un oggetto di classe Libro.

```
public void eliminazione (int cod)
{   StringBuffer rec= new StringBuffer(100);
    try
    {   Libro ……= null;
        RandomAccessFile raf = new RandomAccessFile(filename, "rw");
        raf.seek((……..)(cod-1)*100);
        l=leggiRec(raf);
        l.setCodice(0);
        raf.seek((long)(cod-1)*100);
        rec…….(0 + ";" + aut +";" + edit + ";" + tit + ";" + pr + ";");
        for (int i=rec.........(); i<99; i.........)
              rec…..........(" ");
        rec.append("\n");
        raf.writeBytes(rec.toString());
    }
    catch (IOException e)
    {   System.out.println("Fine file");  }
}
```

(D) ESERCIZI DI APPLICAZIONE

1. Aggiungere alla classe *Libreria* il metodo

   public *modifica* (**int** *cod*, Libro *l*)

   che aggiorna, se esiste, l'oggetto di classe Libro avente codice *cod*, con il nuovo oggetto *l*, passato come parametro, *con controllo di esistenza*.
2. Aggiungere alla classe *Libreria* il metodo

   public *stampaAutore* (**Stringa** *autore*)

   che stampa l'elenco di tutti i libri di un dato autore, il cui nome è passato come parametro.
3. Si supponga di voler riscrivere l'applicazione, facendo usodi un file ordinato. Descrivere le modifiche progettuali e implementative da apportare.