ESERCIZIO 1 (FILE DI TESTO)

- Scrivere su un file di testo righe inserite da console, fino a quando non viene inserita la linea vuota. Passare il nome del file come parametro al programma.
  - Bisogna incapsulare **`System.in`** in un **`InputStreamReader`** che a sua volta è incapsulato da un **`BufferedReader`**
  - Si può inoltre incapsulare **`FileWriter`** in un **`PrintWriter`**
- Come verifica, leggere lo stesso file di testo e stamparne a video il contenuto.
  - Si può incapsulare **`FileReader`** in un **`BufferedReader`**
- Creare una copia del file il cui nome è dato dalla concatenazione di "Copia_di_" e il nome originale del file.
- Gestire tutte le operazioni su file sotto controllo di eccezioni.

ESERCIZIO 2 (FILE BINARIO)

- Leggere valori interi dal file binario IN.DAT (sul web)
  - Si può incapsulare **FileInputStream** in un **DataInputStream**
- Se gli interi letti appartengono a [-5;5] scriverli su un altro file binario OUT.DAT
- Come verifica, rileggere OUT.DAT e stampare a video i numeri interi contenuti
- N.B.: il metodo **readInt()** della classe **DataInputStream** per segnalare la fine del file solleva l'eccezione **EOFException** che deve essere gestita (ad es. chiudendo tutti gli stream).

ESERCIZIO 3 (FILE BINARIO)

- Leggere interi e caratteri da riga di comando (args[])
- Nel caso in cui args[i] sia un intero → scrittura dell'intero sul file "interi.bin"
- Nel caso in cui args[i] sia un char → scrittura del carattere sul file "caratteri.bin"
- Controllare che i file creati siano corretti leggendone il contenuto e stampandolo a video

ESERCIZIO 4 (FILE DI TESTO)

- Definire la classe **`MyBufferedReader`** che estende la classe **`BufferedReader`**, implementando il metodo

  `public String readNRows(int nRows)`
- Il metodo **`readNRows`** legge dallo stream **`nRows`** righe e restituisce una stringa formata dalle righe lette concatenate tra loro, se **`nRows`** righe possono essere lette.
- Se non sono presenti abbastanza righe nello stream il metodo lancia l'eccezione **`RowsNotFoundException`**.
- Definire la classe **`RowsNotFoundException`** che estende la classe **`Exception`**.

ESERCIZIO 4 cont. (FILE DI TESTO)

- Definire la classe **`MyFileWriter`** che estende la classe **`FileWriter`**, che contiene la variabile

```
private int row;
```

- Implementare i costruttori in modo da azzerare la variabile **`row`**.
- Ridefinire il metodo

```
public void write(String str)
```

che dovrà scrivere prima di **`str`** il numero di riga attuale seguita da una *tabulazione*, *:* e uno *spazio*.

- Ridefinire il metodo

```
public void close()
```

che dovrà scrivere nel file il numero totale di righe scritte e infine chiudere il file.

ESERCIZIO 4 cont. (FILE DI TESTO)

- Il metodo *main* dovrà, usando le classi implementate, leggere da un file di testo (di contenuto arbitrario) tutte le righe e scriverle in un secondo file.
- Una volta eseguita la copia del file, chiudere e riaprire lo stream di lettura e leggere N righe (N <= righe totali) del file di input scrivendole in un terzo file usando il metodo **`readNRows(int nRows)`**.
- N.B.: Provare a inserire un valore di righe tale da scatenare l'eccezione **`RowsNotFoundException`**.

ESERCIZIO 5 (FILE DI TESTO)

- Si realizzi una classe **astratta** **`RGBObject`** che rappresenta la tipologia di oggetti grafici il cui colore è espresso tramite valori RGB. Tale classe ha attributi **`red`** (intero), **`green`** (intero), **`blue`** (intero) e definisce il metodo:
  `public boolean eqColor(RGBObject x)`
  che restituisce un booleano.

- Si realizzi poi un componente software **`RGBRectangle`**, che deriva da **`RGBObject`**, con attributi aggiuntivi di tipo intero **`width`** (larghezza del rettangolo) e **`height`** (altezza del rettangolo). Tale componente codifica un metodo costruttore a 5 argomenti (larghezza, altezza e i tre componenti del colore) e implementa il metodo **`toString()`** (operazione di override). Implementa inoltre il metodo
  `public boolean eqColor(RGBObject x).`

ESERCIZIO 5 cont. (FILE DI TESTO)

Tale metodo riceve il riferimento a un'istanza **`x`** della classe **`RGBObject`** e restituisce **`true`** se l'oggetto passato per riferimento ha gli stessi componenti di colore (red, green e blue) di quello su cui è invocato il metodo stesso. Diversamente restituisce **`false`**.

- Si realizzi poi un metodo **`main`** in una classe **`Prova`** che:

1. Crei 3 oggetti **`r1, r2 e r3`** istanze della classe **`RGBRectangle`**, con attributi:

	width	**height**	**red**	**green**	**blue**
r1	100	50	125	80	200
r2	32	10	220	76	88
r3	155	156	125	80	200

ESERCIZIO 5 cont. (FILE DI TESTO)

2. Utilizzando il metodo **`eqColor(RGBObject x)`** verifichi quali oggetti **`RGBRectangle`** hanno gli stessi valori come componenti di colore e ne scriva su file di testo (risultati.txt) i valori di larghezza e altezza.

Esempio di file di testo in output:

```
width: 100      height: 50
width: 155      height: 156
```

Nota: per stampare i valori di larghezza e di altezza di un oggetto **`RGBRectangle`**, implementare il metodo **`toString()`** della stessa classe come segue:

```
public String toString() {
      return ("width: " + this.width + "\theight: " +
this.height);
}
```