2 `CheckPaths` [LPRO 12pt.]

Si vuole implementare la classe `CheckPaths` che consente di verificare se alcuni path specificati in un file esistano e siano del *tipo richiesto*.

La classe sarà utilizzata in tutti quei progetti che necessitano di una struttura per poter funzionare (es. Netbeans stesso).

Ecco un esempio del file `ZZZ_Check.txt`:

```
d .\nbproject
d .\src
f .\build.xml
f .\manifest.mf
d .\questo di sicuro è un errore
f .\anche questo
```

Ogni riga è formata da una lettera, uno spazio e un path. La prima lettera indica il *tipo richiesto*:

- `'f'`: il path punterà ad un file
- `'d'`: il path punterà a una directory

La classe contiene un solo metodo, `toCheck`, che:

- Riceve in argomento il `Path` del file di configurazione
- Restituisce un `boolean`
- Compito:
  - Il metodo legge ogni singola riga del file e controlla se il path indicato esiste e se corrisponde al *tipo richiesto*
  - Il programma verifica l'esistenza e il *tipo* di tutti i path indicati
  - Il metodo stampa a video `"OK"` oppure `"KO!"` seguiti dal path
  - Al termine restituisce `true` se tutti i path hanno dato esito positivo, `false` se almeno uno ha dato esito negativo
  - In caso di errori di sistema viene stampato un generico `"Error"` e viene restituito `false`

Scrivete la classe, il metodo, e invocatelo dal `main` (usando l'esempio dato). Il metodo stamperà il seguente risultato (se avete creato il progetto NetBeans come indicato in aula, ovvero con Ant):

```
OK  .\nbproject
OK  .\src
OK  .\build.xml
OK  .\manifest.mf
KO! .\questo di sicuro è un errore
KO! .\anche questo
```

Stampate a video quanto restituito dal metodo `toCheck` (in questo esempio sarà `false`).

2 `Creation` [LPRO 12pt.]

Si vuole implementare la classe `Creation` che consente di creare files e directory effettuando un controllo prima di procedere. La classe sarà utilizzata in tutti quei progetti che necessitano di una struttura per poter funzionare (es Netbeans stesso).

Ecco un esempio del file `ZZZ_Create.txt`:

```
.\resources\a.txt f
.\resources\b.txt f
.\resources\dir1 d
.\resources\dir2 d
.\resources\f.txt f
```

Ogni riga è formata da un path, uno spazio e una lettera. La lettera finale indica se il path rappresenta un file (`'f'`) o una directory (`'d'`).

La classe contiene un solo metodo, `toCreate`, che:

- Riceve in argomento una `String` rappresentante il path del file
- Restituisce un `boolean`
- Compito:
  - Il metodo legge ogni singola riga del file
  - Se il path esiste stampa `"KO!"` seguito dal path
  - Se il path non esiste (date sempre per scontato che i path parent siano esistenti):
    - Se l'ultima lettera è una `'f'` viene generato un file
    - Se l'ultima lettera è una `'d'` viene generata una directory
    - Il metodo stampa a video `"OK"` seguito dal path
  - Al termine restituisce `true` se è riuscito a creare tutti path indicati, `false` se almeno uno esisteva già
  - In caso di errori di sistema viene stampato un generico `"Error"` e viene restituito `false`

Scrivete la classe, il metodo, e invocatelo dal `main` (usando l'esempio dato) e stampando a video il valore restituito dal metodo.

Alla prima esecuzione, siccome nessun path esiste, dovreste ottenere il seguente output:

```
OK  .\resources\a.txt
OK  .\resources\b.txt
OK  .\resources\c
OK  .\resources\d
OK  .\resources\f.txt
true
```

Alla seconda esecuzione invece otterrete:

```
KO! .\resources\a.txt
KO! .\resources\b.txt
KO! .\resources\c
KO! .\resources\d
KO! .\resources\f.txt
false
```

2 `ListFiles` [LPRO 12pt.]

Si vuole implementare la classe `ListFiles` che consente di elencare tutti i files e le directory contenute in un path indicando se essi siano files o directory.

Il codice Java che permette di ottenere l'elenco dei files è: `baseDir.toFile().list();`

La classe contiene un solo metodo, `toList`, che:

- Riceve in argomento un `Path` rappresentante il percorso da monitorare
- Riceve in argomento un `Path` rappresentante il file di output che si dovrà generare
- Restituisce un `boolean`
- Compito:
  - Il metodo verifica se il file di output esiste (secondo argomento del metodo), in caso positivo si ferma restituendo `false` (così come in caso di errori di sistema)
  - Ottiene l'elenco dei files contenuti nel path indicato dal primo argomento e:
    - Se è un file scriverà nel file di output una `'f'` seguita da uno spazio e dal percorso assoluto del file.
    - Se è una directory scriverà nel file di output una `'d'` seguita da uno spazio e dal percorso assoluto del file.
    - In entrambi i casi stamperà anche a video l'informazione.
  - Restituisce `true`.

Dal `main`, invocate il metodo fornendo gli argomenti e stampando il valore ritornato.

Esempio (il nostro progetto NetBeans si trova in `c:\temp\Espe4`), se invochiamo il metodo fornendo il percorso relativo `"."` e il file `".\ZZZ_List.txt"`, ci troveremo a schermo:

```
d C:\temp\Espe4\build
f C:\temp\Espe4\build.xml
d C:\temp\Espe4\dist
f C:\temp\Espe4\manifest.mf
d C:\temp\Espe4\nbproject
d C:\temp\Espe4\resources
d C:\temp\Espe4\src
d C:\temp\Espe4\test

true
```

E nel percorso della directory `"."` (ovvero `c:\temp\Espe4`) avremo il file `ZZZ_List.txt`.

Se lo eseguiamo una seconda volta otterremo un semplice: ~~~ false ~~~

In quanto il programma rileva l'esistenza di:

```
ZZZ_List.txt
```