Risolutore di puzzle – Parte 1

Programmazione concorrente e distribuita
Progetto A.A. 2014/2015

1 Descrizione del progetto

La parte 1 del progetto consiste nell'implementare un programma che ricostruisce un puzzle partendo dall'insieme dei pezzi mischiati.
Il puzzle consiste in una serie di tessere, ognuna delle quali rappresenta un carattere di un testo. Una volta che il puzzle è costruito correttamente sarà possibile leggere il testo corrispondente partendo dalla prima riga in alto e scorrendo ogni riga da sinistra verso destra.
Il programma

- prende in input un file con l'elenco dei pezzi mischiati
- restituisce in output il testo corrispondente, il puzzle ricostruito e la dimensione del puzzle.

2 Rappresentazione del puzzle

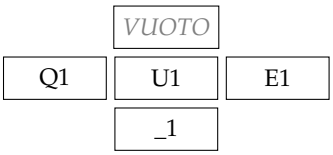
Il puzzle ha un certo numero di righe R e colonne C e contiene RxC pezzi.
Il puzzle rappresenta un testo ed ogni suo pezzo rappresenta un carattere del testo.
Si assume che il testo contiene solo caratteri alfanumerici e punteggiatura; non contiene il carattere di tabulazione e il ritorno a capo.
Esempio di puzzle:

Q	U	E	S	T
O		È		U
N		P	U	Z
Z	L	E		D
I		E	S	E
M	P	I	O	.

Ogni pezzo ha un identificativo univoco (ID) — che può essere una stringa qualsiasi — ed informazione locale sui pezzi che gli stanno vicino nelle quattro direzioni. In particolare ogni pezzo conosce gli ID dei vicini; nel caso in cui un pezzo sia sul bordo del puzzle si assume che nella direzione del bordo l'informazione sia rappresentata dalla stringa ("VUOTO"). Assumendo che gli ID dei pezzi del puzzle di esempio siano:

Q1	U1	E1	S1	T1
O1	_1	È1	_2	U2
N1	_3	P1	U3	Z1
Z2	L1	E2	_4	D1
I1	_5	E3	S1	E4
M1	P2	I2	O2	.1

prendendo il pezzo con ID=U1 del puzzle di esempio (carattere U) si ha che l'informazione locale del carattere è:



3 Requisiti obbligatori

La parte 1 del progetto deve soddisfare i seguenti requisiti obbligatori:

- Il programma deve gestire puzzle di dimensione qualsiasi.
- Il programma non avrà thread né sarà distribuito.
- Il programma non avrà un'interfaccia grafica.
- Allegato al programma ci sarà una relazione.
- Il programma si chiamerà **PuzzleSolver** e sarà avviato con la seguente linea di comando dalla root del progetto
  `PuzzleSolver nome_file_input nome_file_output`
- 6. Il programma dovrà essere in grado di processare un file di input con le seguenti caratteristiche:
  - Codifica del file di input: UTF-8.
  - Formato del file di input:
    Ogni riga rappresenta un pezzo, per cui ogni riga sarà una stringa così composta
    `id_pezzo \t carattere \t id_nord \t id_est \t id_sud \t id_ovest`
    dove `\t` è il carattere di tabulazione.
- Il programma dovrà prodirre un file di output con le seguenti caratteristiche:
  - Codifica del file di input: UTF-8.
  - Formato del file di output:
    La prima riga contiene il testo rappresentato dal puzzle, non essendo permessi nel testo caratteri di ritorno a capo, il testo sarà scritto in una sola riga. La seconda riga del file sarà vuota.
    Dalla terza riga sarà scritto il puzzle nella sua forma"tabellare" (RxC colonne in cui ogni cella è un carattere, non saranno aggiunti dei caratteri per generare i bordi della tabella).
    La riga sucessiva sarà vuota.
    La riga successiva conterrà la dimensione della tabella nel formato `R C`.

4 Valutazione del progetto - Relazione

La valutazione del progetto terrà conto dell'aderenza del progetto ai principi basilari della programmazione ad oggetti in Java: encapsulation e information hiding, modularizzazione del codice, riutilizzo del codice, polimorfismo, organizzazione delle classi.
Al fine di permettere la valutazione del progetto, è richiesto quindi di consegnare, oltre al programma, una **breve relazione** che illustri in modo **preciso** e **sintetico**:

- quali scelte effettuate realizzano quali principi della programmazione ad oggetti.
- come sono organizzate tra loro le classi del progetto (es. ereditarietà, interfacce, classi astratte, classi interne) e perché sono state organizzate in quel modo.
- una breve spiegazione della logica dell'algoritmo di ricostruzione del puzzle.
- una breve spiegazione dei test fatti per dimostrare la correttezza del programma.

NOTA: La valutazione del progetto dipenderà in **ugual misura dalla qualità della relazione e dalla correttezza e qualità del codice**. Si osserva inoltre che la valutazione del progetto terrà conto della capacità di risolvere il problema assegnato in modo semplice ma esauriente. Sono quindi preferibili i progetti che implementano in modo semplice e chiaro una soluzione corretta, mentre viene **SCORAGGIATA l'aggiunta di funzionalità e aspetti grafici non richiesti dalla presente specifica**.

5 Consigli

Nelle successive due fasi di progetto sarà richiesto di rendere concorrente e distribuito l'algoritmo di ricostruzione del puzzle. In questa prima fase è dunque consigliabile ideare un algoritmo di ricostruzione che si presti poi ad essere parallelizzato.

5.1 Manipolazione dei file

Per manipolare i file è utile leggere l'articolo all'url
`http://docs.oracle.com/javase/tutorial/essential/io/file.html`
in particolare la sezione intitiolata "Buffered I/O Methods for Text Files".
Di seguito si riporta una classe di esempio — compatibile con Java 7 o superiore — che legge e scrive da file.
Si prega di notare l'uso del costrutto **try-with-resources** di Java 7, una spiegazione chiara di tale costrutto si trova all'url `http://tutorials.jenkov.com/java-exception-handling/try-with-resources.html`

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

/*
 * Questa programma prende come parametri due file,
 * legge il contenuto dal primo e lo scrive nel secondo
 * rimuovendo i caratteri di ritrono a capo.
 */
public class FileTestJava7 {

  private static Charset charset = StandardCharsets.UTF_8;

  public static void main(String[] args) {
    String inputFile = args[0];
    String outputFile = args[1];
```

```
    Path inputPath = Paths.get(inputFile);
    Path outputPath = Paths.get(outputFile);

    String inputContent = readContent(inputPath);
    writeContent(outputPath, inputContent);
  }

  private static String readContent(Path inputPath) {
    StringBuilder content = new StringBuilder();
     try (BufferedReader reader = Files.newBufferedReader(inputPath,
         charset)) {
           String line = null;
           while ((line = reader.readLine()) != null) {
              content.append(line);
           }
        } catch (IOException e) {
           System.err.println(e);
        }
     return content.toString();
  }

  private static void writeContent(Path file, String content) {
    try (BufferedWriter writer = Files.newBufferedWriter(file, charset)) {
       writer.write(content);
    } catch (IOException e) {
      System.err.println(e);
    }
  }
}
```

Per completezza si riporta la classe con le medesime funzionalità compatibile con Java 6.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

/*
 * Questa programma prende come parametri due file,
 * legge il contenuto dal primo e lo scrive nel secondo
 * rimuovendo i caratteri di ritrono a capo.
 */
public class FileTestJava6 {
  public static void main(String[] args) {
    String inputPath = args[0];
    String outputPath = args[1];

    String inputContent = readContent(inputPath);
    writeContent(outputPath, inputContent);
  }

  private static String readContent(String inputPath) {
```

```
    StringBuilder content = new StringBuilder();
    BufferedReader br = null;

    try {
      br = new BufferedReader(new InputStreamReader(new FileInputStream(
          inputPath), "UTF-8"));
      String line = null;
      while ((line = br.readLine()) != null) {
        content.append(line);
      }
    } catch (IOException e) {
      e.printStackTrace();
    } finally {
      try {
        if (br != null)
          br.close();
      } catch (IOException ex) {
        ex.printStackTrace();
      }
    }
    return content.toString();
  }

  private static void writeContent(String file, String content) {
    BufferedWriter writer;
    try {
      writer = new BufferedWriter(new OutputStreamWriter(
          new FileOutputStream(file), "UTF-8"));
      writer.write(content);
      writer.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

6 Regole per la consegna del progetto

Il progetto dovrà essere realizzato da ogni singolo studente in modo INDIPENDENTE.

6.1 Come verrà effettuato il test operativo del progetto

Per quanto riguarda la valutazione del progetto, questo verrà eseguito localmente su un sistema operativo Linux, dove vi è una installazione di Java 7.
Al fine di standardizzare la procedura di valutazione il programma dovrà aderire a quanto specificato nella sezione "'Requisiti obbligatori"'.
ATTENZIONE: se un progetto non compila o non soddisfa i i requisiti obbligatori sarà considerato insufficiente.

6.2 Cosa consegnare

La cartella principale del pacchetto da consegnare, dovrà essere chiamata "'parte1"'. Nel caso di consegna contemporanea di più parti, il pacchetto avrà più cartelle, una per ciascuna parte.

6.3 Come e quando consegnare

Il progetto va consegnato dalle macchine del laboratorio invocando il comando

```
consegna programmazione3-14-15
```

dalla directory contenente i file da consegnare. **Non** saranno accettate altre modalità di consegna (ad es. via email). È possibile consegnare remotamente il progetto usando il server `ssh.studenti.math.unipd.it`.
Le date di consegna del progetto saranno indicate sul sito del corso.