Contents

1 Perchè? 1
1.1 Per i commenti 1
1.2 Per il tokenizer 2

2 Come? 2

1 Perchè?

il parser realizzato per `Jelly` è un po' alla cazzo di cane, fatto ai fini principali di andare avanti brutalmente senza fregarsene un cazzo di indietreggiare una sega, perchè mi andava

per fare ciò si è ritenuto necessario avere queste funzionalità, e visto che sono comuni a più parti del codice, si è ritenuto necessario estrarle in una classe a parte

le funzionalità riguardo al controllare prefissi di una stringa di caratteri sono utilizzate, per adesso, nei seguenti momenti del codice

1.1 Per i commenti

- quando si trova un delimitatore di commenti inline (il `//` per intenderci), si sa che non ci saranno caratteri significativi dopo quello
- quando si trova un delimitatore di inizio commento multiline (il `/*`), si sa che non ci saranno caratteri significativi finchè non si trova il delimitatore di fine commento multiline (il `*/`)

quando si stanno iterando i caratteri del codice, il `//`, o `/*`, o `*/` di interesse sarà osservabile come prefisso della sequenza di caratteri, sarà quindi utile poter chiamare per tirare del codice a cazzo, qualcosa del genere

```
public class SignificantCharactersIterator extends PrefixIterator {
    @Override
    public char next() {
        this.ignoreComments();
        return super.next();
    }

    private void ignoreComments() {
        if(isPrefix("/*")) {
            ignoreUntilPrefix("*/");
        }
```

```
        if(isPrefix("//")) {
            ignoreUntilNewline();
        }
    }
    /* ignoreUntil... non fa altro che chiamare
             * this.next() finchè non si realizza qualche condizione */
}
```

1.2 Per il tokenizer

si prendano la stringa di codice java

```
for(i=0;i<10;++i){System.out.println(i);}
```

questo codice fa schifo, ma è valido un parser che vede questa merda però come deve orientarsi, non è che a quel poveraccio arriva così bellino con la sintassi, gli arriva una cagata tipo

```
"for(i=0;i<10;++i){System.out.println(i);}"
```

e lì so' cazzi, come cazzo si orienta?

ci sono certe stringhe che quando il parser prova a leggerle si deve capire che "ok, questa va a parte", quindi se trovo un `"("` o un `";"` mentre cerco un token devo capire che `"("` non è l'inizio di token tipo `"(i=0;"`, devo metterlo subito da parte

questo, nuovamente, si realizza abbastanza facile se si possono fare check su prefissi, basta fare un

```
if(isPrefix("(")) {
    pijatelo();
}
```

2 Come?

le funzioni che il `PrefixIterator` deve poter fare in fretta sono

- `isPrefix(String s)`
- al massimo `getPrefix(int len)` (questo da vedere se effettivametne serve o meno)

il `PrefixIterator` è un'iterator di `Character`, e funge principalmente da estensione a questo con funzionalità extra per controllare, come si può intuire

dal nome, certi prefissi essendo fatto per controllare prefissi di una sequenza iterata, il prefixIterator agirà più da wrapper o (devo vedere se c'entra col pattern) decorator su un altro `Qualcosa implements Iterator<Character>`