Metodologie di Programmazione

Lezione 8: La classe String

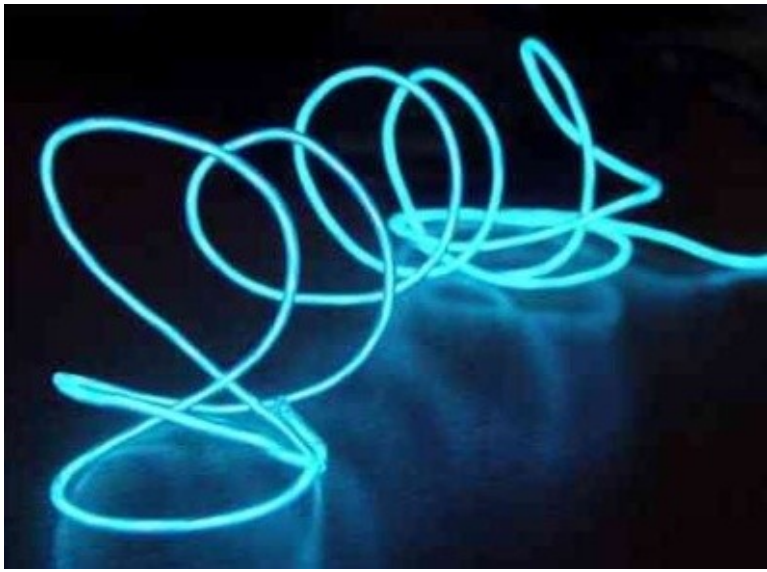
Lezione 8: Sommario

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- La classe java.lang.String
- Metodi per:
  - manipolare le stringhe
  - ottenere informazioni dalla stringa
  - confrontare stringhe
  - ecc.
- Esercizi con le stringhe

La classe java.lang.String

- Una classe fondamentale, perché è relativa a un tipo di dato i cui letterali sono parte della sintassi del linguaggio e per il quale il significato dell'operatore + è ridefinito
- Non richiede import perché appartiene al package "speciale" java.lang



Ottenere la lunghezza di una stringa

- La classe String è dotata del metodo length
- Ad esempio:

```
String s = "ciao";
System.out.println(s.length());
```

- Stamperà il valore 4, che è pari al numero di caratteri di cui è costituita la stringa s

Stringa tutta in maiuscolo o tutta in minuscolo

- Con i metodi toLowerCase() e toUpperCase() si ottiene un'**altra** stringa tutta minuscola o maiuscola, rispettivamente
- La stringa su cui viene invocato il metodo **non viene modificata**
- Ad esempio:

```
String min = "Ciao".toLowerCase(); // "ciao"
String max = "Ciao".toUpperCase(); // "CIAO"
```

Ottenere singoli caratteri

- E' possibile ottenere il k-esimo carattere di una stringa con il metodo charAt
- **Importante:**
  - il primo carattere è in posizione **0**
  - l'ultimo carattere è in posizione **stringa.length()-1**
- Esempio:

```
String s = "ciao";
System.out.println(s.charAt(2));
```

- stamperà il carattere 'a'
- Notate che charAt restituisce un carattere (tipo char), **non** una stringa

Ottenere una sottostringa

- E' possibile ottenere una sottostringa di una stringa con il metodo substr(startIndex, endIndex), dove:
  - startIndex è l'indice di partenza della sottostringa
  - endIndex è l'indice successivo all'ultimo carattere della sottostringa
- Ad esempio:

```
String s = "ciao";
System.out.println(s.substr(1, 3));
```

- Stamperà la stringa "ia", dalla posizione 1 ('i') alla posizione 3 ('o') **esclusa**
- Esiste anche una versione substr(startIndex) equivalente a substr(startIndex, stringa.length())

Concatenare stringhe

- La concatenazione tra due stringhe può essere effettuata con l'operatore "speciale" + oppure mediante il metodo concat(s)

```
String s3 = s1+s2;
String s4 = s1.concat(s2);
```

- Tuttavia, se si devono concatenare parecchie stringhe, è bene utilizzare la classe StringBuilder, dotata dei metodi append e insert

```
StringBuilder sb = new StringBuilder();
sb.append(s1).append(s2);
String s5 = sb.toString();
```

Cercare in una stringa

- Si può cercare la (prima) posizione di un carattere c con indexOf(c)
  - restituisce -1 se il carattere non è presente
- E' possibile anche cercare la prima posizione di una sottostringa con indexOf(s)
- Ad esempio:

```
int k = "happy happy birthday".indexOf('a');
int j = "din din don don".indexOf("don");
int h = "abcd".indexOf('e');
```

- k varrà 1, j varrà 8, mentre h varrà -1
- Con i metodi startsWith e endsWith è possibile verificare prefissi o suffissi della stringa

Sostituire caratteri e sottostringhe

- Con il metodo replace è possibile sostituire tutte le occorrenze di un carattere o di una stringa all'interno di una stringa
- Esempio:

```
// s1 vale "uno due tre"
String s1 = "uno_due_tre".replace('_', ' ');
// s2 vale "uno two tre"
String s2 = "uno due tre".replace("due", "two");
```

Confrontare stringhe

- Le stringhe, come peraltro tutti gli altri oggetti, vanno **SEMPRE** confrontate con il metodo equals
- Che differenza c'è tra equals e ==?
- L'operatore == confronta il riferimento (diciamo, l'indirizzo in memoria), quindi è true **se e solo se** si confrontano gli stessi oggetti fisici
- L'operatore equals confronta la stringa carattere per carattere e restituisce true se le stringhe contengono la stessa sequenza di caratteri
- Ad esempio:

```
String s1 = "ciao", s2 = "ci"+"ao", s3 = "hello";
System.out.println(s1 == s2);       // potrebbe restituire false
System.out.println(s1.equals(s2)); // restituisce true
System.out.println(s1.equals(s3)); // restituisce false
```

Spezzare le stringhe

- Il metodo split prende in input un'espressione regolare s (senza entrare in dettagli, è sufficiente pensarla come una semplice stringa) e restituisce un array di sottostringhe separate da s
- Esempio:

```
String[] parole = "uno due tre".split(" ");
// parole contiene l'array new String[] { "uno", "due", "tre" }
```

Riprendiamo l'esercizio Menù


```
public class Menu
{
	...

	/**
	 * Costruisce un menu' vuoto (senza opzioni)
	 */
	public Menu()
	{
		menuText = "";
		optionCount = 0;
	}

	/**
	 * Visualizza il menu' su console
	 */
	public void display()
	{
		System.out.println(menuText);
	}

	/**
	 * Aggiunge un'opzione alla fine del menu'
	 * @param option l'opzione da aggiungere
	 */
	public void addOption(String option)
	{
		optionCount++;
		menuText += optionCount + ") " + option + "\n";
	}
}
```

Quindi se volessi aggiungere un metodo Menu.getOption()?

- Aggiungere un metodo getOption alla classe Menu che, dato un intero k, restituisca la k-esima opzione

```
/**
 * Restituisce la k-esima opzione del menù
 * @param k la posizione dell'opzione
 * @return la stringa associata all'opzione specificata
 */
public String getOption(int k)
{
    String[] opzioni = menuText.split("\n");
    String opzione = opzioni[k-1];
    int posizione = opzione.indexOf(") ");
    return opzione.substring(posizione+2);
}
```

Esempio: la Stringa “magica”

- Progettare una classe Stringa42, costruita a partire da 3 stringhe in input, che concateni le stringhe inframezzate da spazi e conservi solo i primi 42 caratteri della stringa risultante
- La classe deve poter:
  - restituire la stringa di lunghezza massima 42
  - restituire l’iniziale di tale stringa
  - restituire un booleano che indichi se la stringa è pari al numero “magico” 42
  - restituire un booleano che indichi se la stringa contiene il numero “magico” 42

La Stringa “magica”: soluzione


```
public class Stringa42
{
	private String stringa;

	public Stringa42(String s1, String s2, String s3)
	{
		// equivalente a s1+" "+s2+" "+s3
		stringa = s1.concat(" ").concat(s2).concat(" ").concat(s3);

		// massima lunghezza 42
		if (stringa.length() > 42) stringa = stringa.substring(0, 42);
	}

	public String getStringa()
	{
		return stringa;
	}

	public char getIniziale()
	{
		return stringa.charAt(0);
	}

	public boolean isMagic()
	{
		return stringa.equals("42");
	}

	public boolean containsMagic()
	{
		return stringa.indexOf("42") != -1;
	}

	public static void main(String[] args)
	{
		Stringa42 s = new Stringa42("La risposta", "e'", "42");
		System.out.println(s.getIniziale());
		System.out.println(s.getStringa());
		System.out.println(s.isMagic());
		System.out.println(s.containsMagic());
	}
}
```

Esercizio: punti e segmenti

- Progettare una classe Punto per la rappresentazione di un punto nello spazio tridimensionale
- E una classe Segmento per rappresentare un segmento nello spazio tridimensionale
- Scrivere una classe di test che crei:
  - due oggetti della classe Punto con coordinate (1, 3, 8) e (4, 4, 7)
  - un oggetto della classe Segmento che rappresenti il segmento che unisce i due punti di cui sopra