Programmazione I-B 2020-21

Laboratorio T2

Attilio Fiandrotti
attilio.fiandrotti@unito.it

5 Novembre 2020

Outline

- Soluzione esercizi serie numeriche 29 Ottobre
- Lettura input da tastiera
- Esercizio combinazioni carte

Soluzione esercizi 29 Ottobre

Esercizio: serie numeriche

- Di seguito sono elencate serie numeriche ed il valore cui convergono

$$\sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q} \tag{1}$$

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} \tag{2}$$

$$\sum_{k=0}^{\infty} \frac{1}{(2k+1)^2} = \frac{\pi^2}{8} \tag{3}$$

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{\pi}{4} \tag{4}$$

$$\sum_{k=0}^{\infty} \frac{z^k}{k!} = e^z \tag{5}$$

Esercizio: serie numeriche

- Completare la classe Serie.java per le serie 2-4 (la serie 1 serva d'esempio)
- Completare la classe SerieTest.java con test sui metodi aggiunti in Serie.java secondo l'esempio fornito
- I risultati dei metodi che implementeranno le serie numeriche non produrranno necessariamente un valore esatto (la sommatoria avrà un numero finito di termini). Si sviluppi un opportuno test che verifichi un risultato quando esso è in un intervallo ritenuto ragionevole

Esercizio: serie numeriche - Osservazioni

- Le serie 2-5 saranno da troncare ad un *n* parametro della funzione
- Le serie 2-5 prevedono una divisione con risultato frazionario
  - utilizzeremo l'operatore divisione «/» built-in di Java
  - restituiremo un tipo *float*
- La serie 5 richiede l implementazione della funzione fattoriale

Esercizio: serie 1

$$\sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q}$$

```
1  public static int primaSerie(int q, int n) {
2      int s = 0;
3      int k = 0;
4      while (k < n + 1) {
5        s = s + Aritmetica.pot(q, k);
6        k = k + 1;
7      }
8      return s;
9  }
```

Esercizio: serie 1

$$\sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q}$$

```
1  public class SerieTest {
2    public static void main(String[] args) {
3      final float ERRORE = 0.5f; // errore ritenuto ragionevole
4      int q, n; // parametri di input
5      float output, outputAtteso;
6
7      q = 2; n = 4;
8      outputAtteso = (1 - Aritmetica.pot(q, n + 1))/(1 - q);
9      output = Serie.primaSerie(q, n);
10     System.out.println("Serie.s1(" + q + "," + n + ") returns " + output +
11                        " vs. " + outputAtteso + " -> " +
12                        SerieTest.compareEpsilon(output, outputAtteso, ERRORE));
13   }
14
15   public static boolean compareEpsilon (float a, float b, float epsilon) {
16       float diff = a - b;
17       if (diff < 0) {diff = -diff;}
18       if (diff < epsilon)
19           return true;
20       return false;
21   }
```

Esercizio: serie 1 – V2

$$\sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q}$$

- Ottimizzo il codice esplicitando la potenza come prodotto

```
public static int primaSerieV2(int q, int n) {
  int s = 1;
  int num = 1;
  int k = 1;
  while (k <= n + 1) {
    num = Aritmetica.perU(num, q);
    s = s + num;
    k = k + 1;
  }
  return s;
}
```

Termine per k=0 ($q^0=1$)

$$\sum_{k=0}^{n} q^k = 1 + \sum_{k=1}^{n} q^k$$

Esercizio: serie 2

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

accumulatore e risultato di tipo *float*

```
public static float secondaSerie(int n) {
  float s = 0;
  int den = 0;
  int k = 1;
  while (k < n + 1) {
    den = Aritmetica.pot(k, 2);
    s = s + (1 / (float)den);
    k = k + 1;
  }
  return s;
}
```

Cosa succede senza cast a *float ?*

Esercizio: serie 2

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

```
// Test seconda serie
n = 100;
outputAtteso = Aritmetica.quo(Aritmetica.pot((float)Aritmetica.pi, 2), 6);
output = Serie.secondaSerie(n);
System.out.println("Serie.s2(" + n + ") returns " + output +
                   " vs. " + outputAtteso +
                   " -> " + SerieTest.compareEpsilon(output, outputAtteso, ERRORE));
```

```
public static float pot(float a, int b) {
  float res = 1;
  int i = 0;
  while (i < b) {
    res = res * a;
    i = i + 1;
  }
  return res;
}
```

Esercizio: serie 3

$$\sum_{k=0}^{\infty} \frac{1}{(2k+1)^2} = \frac{\pi^2}{8}$$

```
public static float terzaSerie(int n) {
   float s = 0;
   int den = 0;
   int k = 0;
   while (k < n + 1) {
      den = Aritmetica.pot(Aritmetica.piu(Aritmetica.perU(k, 2), 1), 2);
      s = s + (1 / (float)den);
      k = k + 1;
   }
   return s;
}
```

Esercizio: serie 4

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{\pi}{4}$$

```
public static float quartaSerie(int n) {
   float s = 0;
   int num = 0, den = 0;
   int k = 0;
   while (k < n + 1) {
      num = Aritmetica.pot(-1, k);
      den = Aritmetica.piu(Aritmetica.perU(2, k), 1);
      s = s + (num / (float)den);
      k = k + 1;
   }
   return s;
}
```

Esercizio: serie 5

$$\sum_{k=0}^{\infty} \frac{z^k}{k!} = e^z$$

```
public static float quintaSerie(int z, int n) {
  float s = 0;
  int num = 1, den = 0;
  int k = 0;
  while (k < n + 1) {
    num = Aritmetica.pot (z, k);
    den = fattoriale(k);
    s = s + ((float)num / (float)den);
    k = k + 1;
  }
  return s;
}
```

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

```
public static int fattoriale(int n) {
  int k = 1;
  int ret = 1;
  while (k <= n) {
      ret = Aritmetica.perU(ret, k);
      k = k + 1;
  }
  return ret;
}
```

Esercizio: serie 5 – V2

$$1 + \sum_{k=1}^{\infty} \frac{z^k}{k!} = e^z$$

```
public static float quintaSerieV2(int z, int n) {
  float s = 1;
  int num = 1, den = 1;
  int k = 1;
  while (k < n + 1) {
    num = Aritmetica.perU(num, z);
    den = Aritmetica.perU(den, k);
    s = s + ((float)num / (float)den);
    k = k + 1;
  }
  return s;
}
```

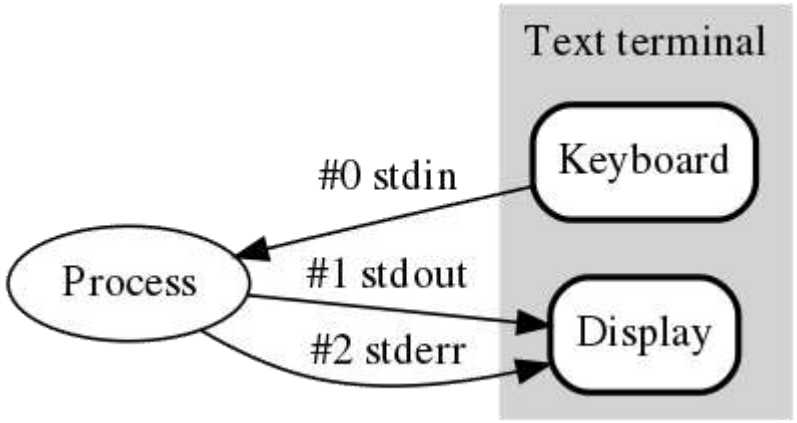
$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

```
public static int fattoriale(int n) {
  int k = 1;
  int ret = 1;
  while (k <= n) {
      ret = Aritmetica.perU(ret, k);
      k = k + 1;
  }
  return ret;
}
```

Lettura input da tastiera

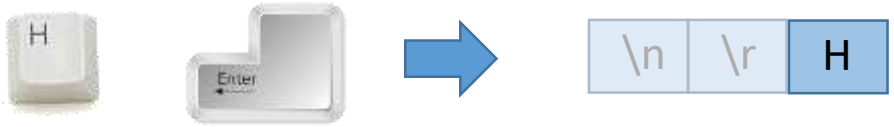
Gli *standard streams*

- Ogni programma Java dispone di tre «*canali*» di I/O
- 0) Standard input (System.in)
  - Lettura caratteri da tastiera
- 1) Standard output (System.out)
  - Stampa caratteri a schermo
- 2) Standard error (System.err)
  - Stampa caratteri a schermo (debug, errore)

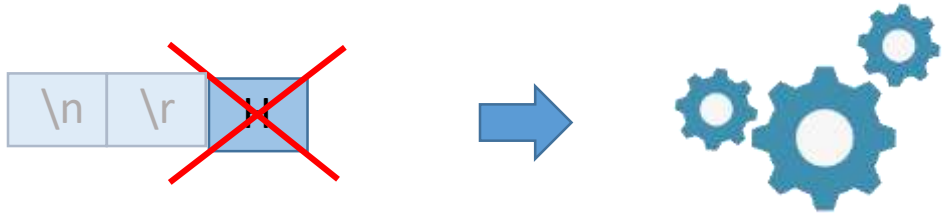


Lettura di un singolo carattere

- System.in.read() permette di leggere *sequenze* di caratteri
  - Per ora leggeremo un singolo carattere
- Il carattere immesso é memorizzato in una *coda* (FIFO)



- Premendo <Enter>, System.in.read() si sblocca e ritorna un carattere
  - La coda si svuota del carattere ritornato *



Lettura di un singolo carattere – *SIn.read()*

- System.in.read() può scatenare una *eccezione*
  - Useremo il wrapper *read()* fornita dalla classe *SIn.java*

Bloccante in attesa di tasto <Enter>

Destion dell' eccezione con *try-catch*

```
public static int read() {
    int result = -1; //To keep the compiler happy
    try {
        result = System.in.read();
    }
    catch(IOException e)  {
        System.out.println(e.getMessage());
        System.out.println("Fatal error.");
        System.exit(0);
    }
    return result;
}
```

Lettura di un singolo carattere, esempio

Bloccante in attesa di <Enter>

```
public static void main (String[] args){
    System.out.print("In attesa di input: ");
    int tasto = SIn.read();
    System.out.println("Carattere " + tasto + " -> " + (char)tasto);
}
```

Casting a *char*
(vedi tab. ASCII)

```
> java SystemInReadTest
In attesa di input: H
Carattere 72 -> H
```

Lettura di un singolo carattere

- SIn.*read()* ritorna un *int*
  - E' necessario casting a (char) per stampare a schermo o concatenare la stringa
- Codifica carattere *ASCII (7 bit)*
  - Primi 32 caratteri speciali
  - Restanti caratteri stampabili
  - Es: «H» = 72

Ascii	Char	Ascii	Char	Ascii	Char	Ascii	Char
0	Null	32	Space	64	@	96	`
1	Start of heading	33	!	65	A	97	a
2	Start of text	34	"	66	B	98	b
3	End of text	35	#	67	C	99	c
4	End of transmit	36	$	68	D	100	d
5	Enquiry	37	%	69	E	101	e
6	Acknowledge	38	&	70	F	102	f
7	Audible bell	39	'	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Horizontal tab	41	)	73	I	105	i
10	Line feed	42	*	74	J	106	j
11	Vertical tab	43	+	75	K	107	k
12	Form feed	44	,	76	L	108	l
13	Carriage return	45	-	77	M	109	m
14	Shift in	46	.	78	N	110	n
15	Shift out	47	/	79	O	111	o
16	Data link escape	48	0	80	P	112	p
17	Device control 1	49	1	81	Q	113	q
18	Device control 2	50	2	82	R	114	r
19	Device control 3	51	3	83	S	115	s
20	Device control 4	52	4	84	T	116	t
21	Neg. acknowledge	53	5	85	U	117	u
22	Synchronous idle	54	6	86	V	118	v
23	End trans. block	55	7	87	W	119	w
24	Cancel	56	8	88	X	120	x
25	End of medium	57	9	89	Y	121	y
26	Substitution	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	File separator	60	<	92	\	124	|
29	Group separator	61	=	93	]	125	}
30	Record separator	62	>	94	^	126	~
31	Unit separator	63	?	95	_	127	Forward del.

Lettura di un singolo carattere, esempio

```
public static void main (String[] args){
    System.out.print("In attesa di input: ");
    while(true) {
        int tasto = SIn.read();
        System.out.println("Carattere " + tasto + " -> " + (char)tasto);
    }
}
```

Chiamata a SIn.read()
da un loop infinito

Lettura di un singolo carattere, esempio

```
public static void main (String[] args){
    System.out.print("In attesa di input: ");
    while(true) {
        int tasto = SIn.read();
        System.out.println("Carattere " + tasto + " -> " + (char)tasto);
    }
}
```

Caratteri «invisibili» cod. 13 e 10

```
> java SystemInReadTest
In attesa di input: H
Carattere 72 -> H
Codice carattere 13 ->
Codice carattere 10 ->
```

Lettura di un singolo carattere, esempio

```
public static void main (String[] args){
    System.out.print("In attesa di input: ");
    while(true) {
        int tasto = SIn.read();
        System.out.println("Carattere " + tasto + " -> " + (char)tasto);
    }
}
```

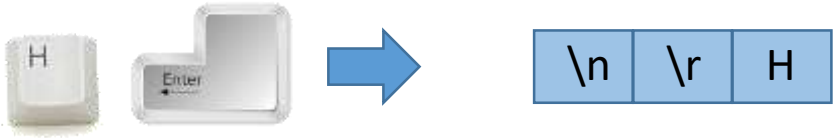
Caratteri «invisibili» cod. 13 e 10

```
> java SystemInReadTest
In attesa di input: H
Carattere 72 -> H
Codice carattere 13 ->
Codice carattere 10 ->
```

Ascii	Char	
0	Null	
1	Start of heading	
2	Start of text	
3	End of text	
4	End of transmit	
5	Enquiry	
6	Acknowledge	
7	Audible bell	
8	Backspace	
9	Horizontal tab	
10	Line feed	«\n»
11	Vertical tab	
12	Form feed	
13	Carriage return	«\r»

Lettura di un singolo carattere, dettagli

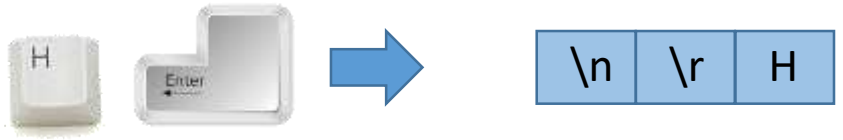
- In Windows il <Return> é codificato tramite CR («\r») + LF («\n»)
- Questi caratteri *speciali* sono accodati al carattere inserito nella coda



- SIn.read() memorizza nella coda *anche* questi caratteri *speciali*
- La coda va svuotata perché sia pronta per la successiva *read()*

Lettura di un singolo carattere, dettagli

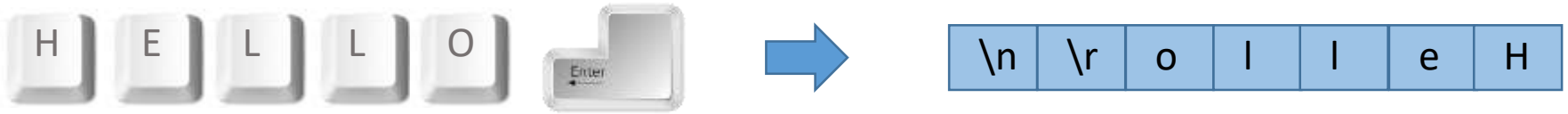
- Estraggo dalal coda ‘\r’ ed ‘\n’ ma li ignoro




```
public static void main (String[] args){
    System.out.print("In attesa di input: ");
    while(true) {
        int tasto = SIn.read();
        if ((char)tasto != '\r' && (char)tasto != '\n')
            System.out.println("Carattere " + tasto + " -> " + (char)tasto);
    }
}
```

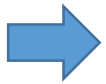
Lettura di una stringa – readWord()

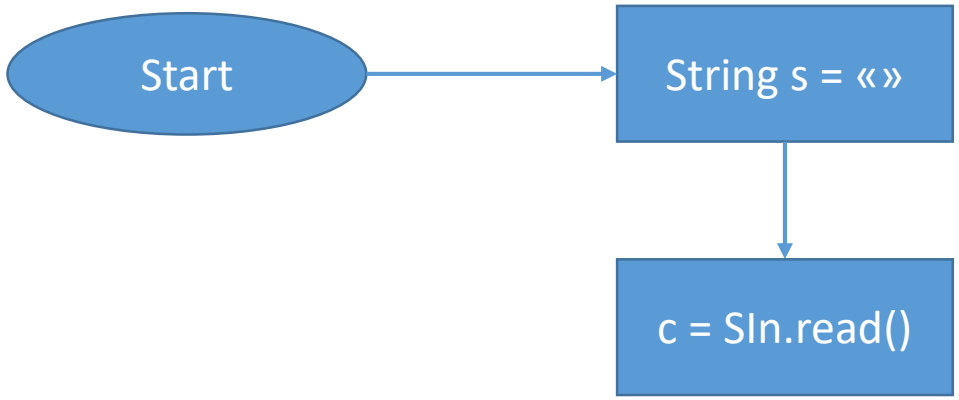
- Abbiamo tutti gli elementi per leggere una stringa di testo
- L'utente preme una sequenza di caratteri
- I carattaeri sono inseriti nella coda



- Ogni *SIn.read()* estrae un carattere dalla coda
- Chiamata a *Sin.read()* in un ciclo while
  - I caratteri stampabili sono concatenati in una stringa
  - I caratteri speciali vengono utilizzati per determinare la fine del ciclo

Lettura di una stringa – readWord()



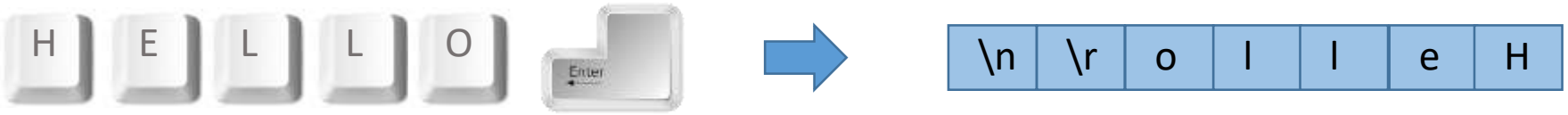


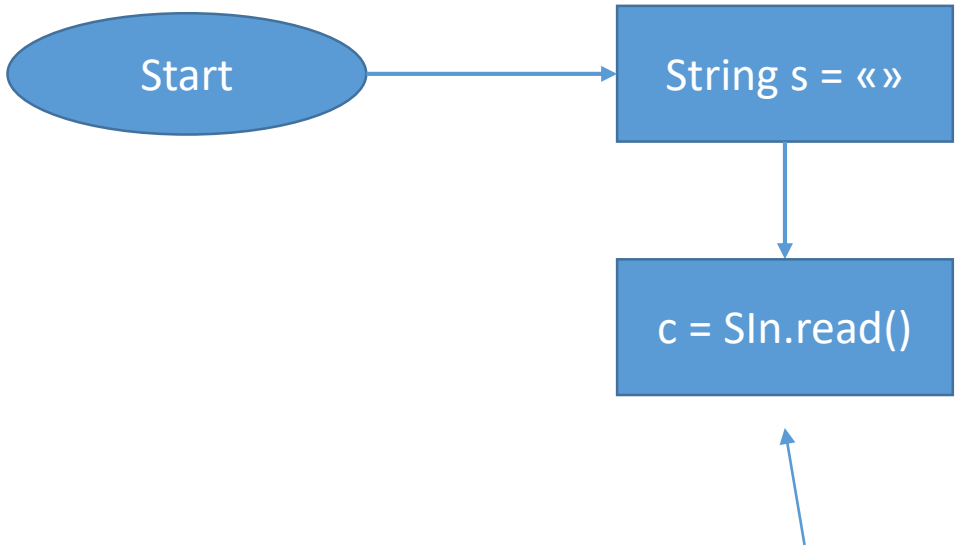
Coda contenete 5 caratteri

SIn.read() in attesa di <Enter>

s= «»

Lettura di una stringa – readWord()

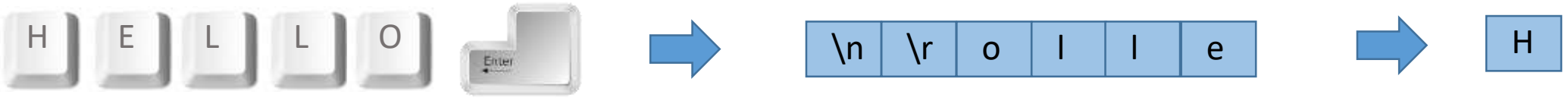


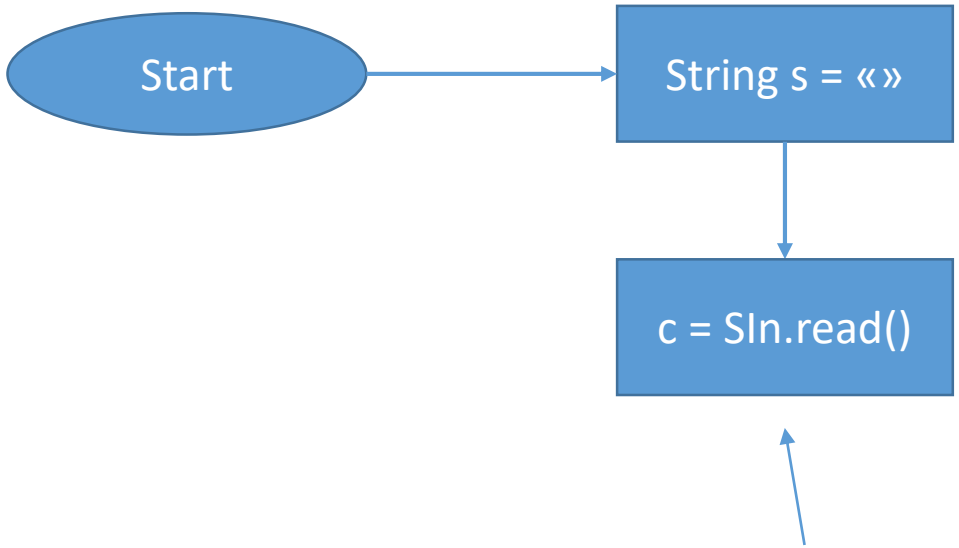


- Effetto pressione tasto <Enter>
  - Inserimento nella coda di ‘\r’ e ‘\n’

s= «»

Lettura di una stringa – readWord()

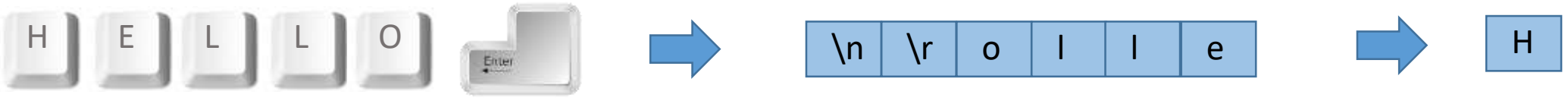


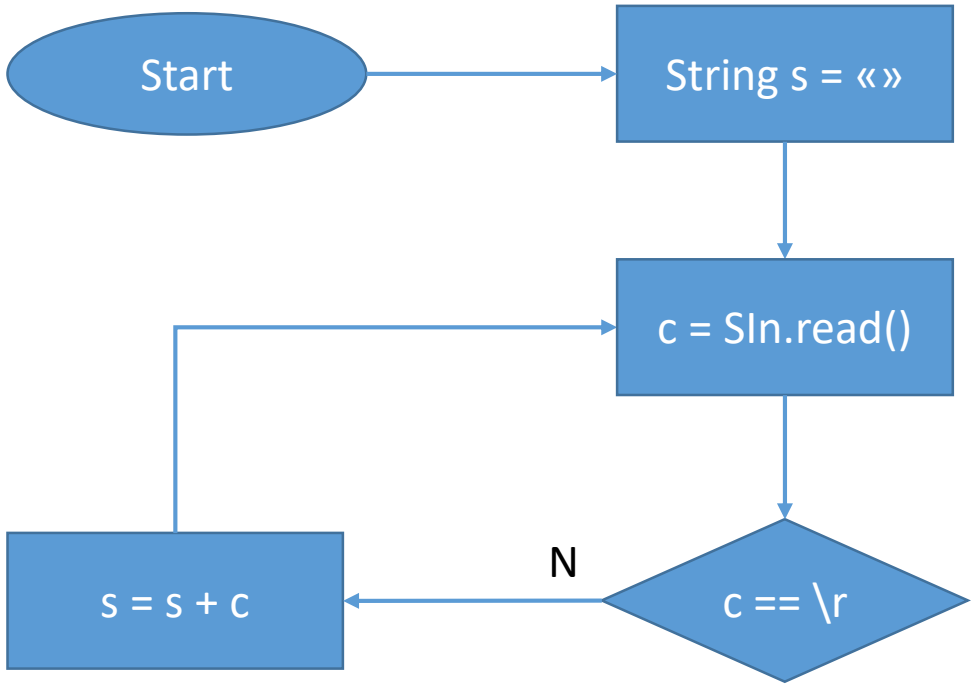


- Effetto pressione tasto <Enter>
  - Inserimento nella coda di '\r' e '\n'
  - *S In.read()* si sblocca ed estrae dalla coda 'H'

s= «»

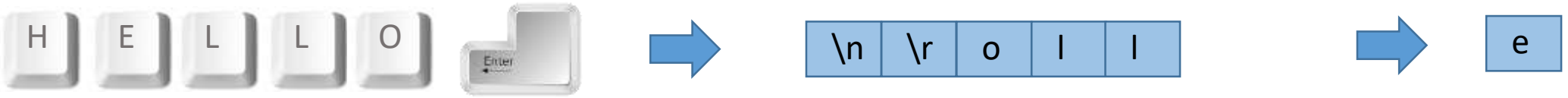
Lettura di una stringa – readWord()

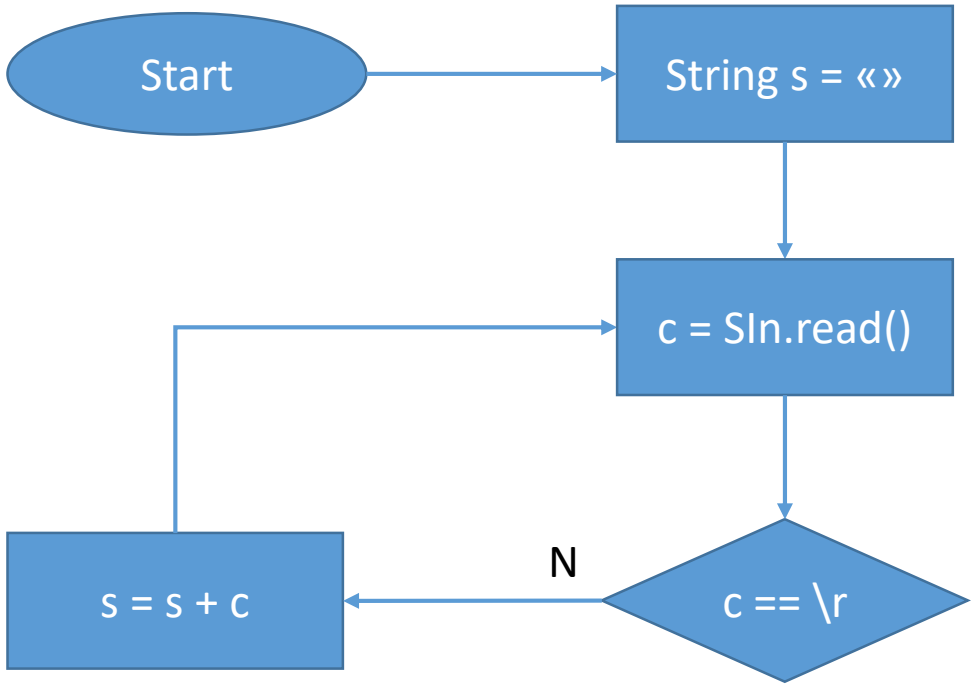




s= «H»

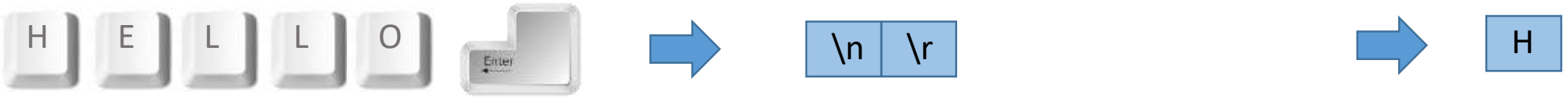
Lettura di una stringa – readWord()

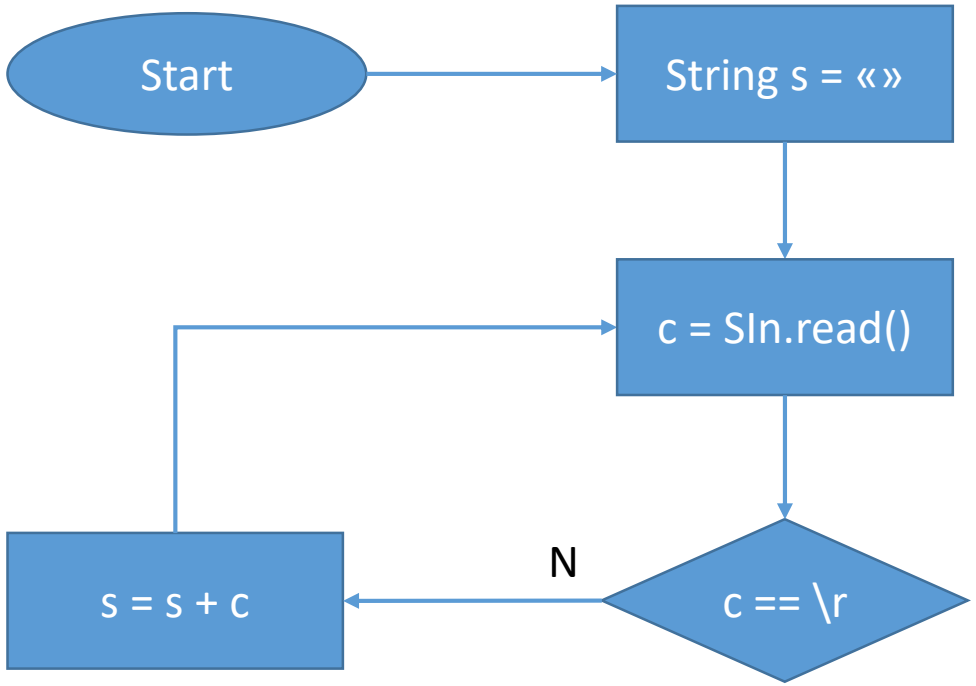




s= «He»

Lettura di una stringa – readWord()



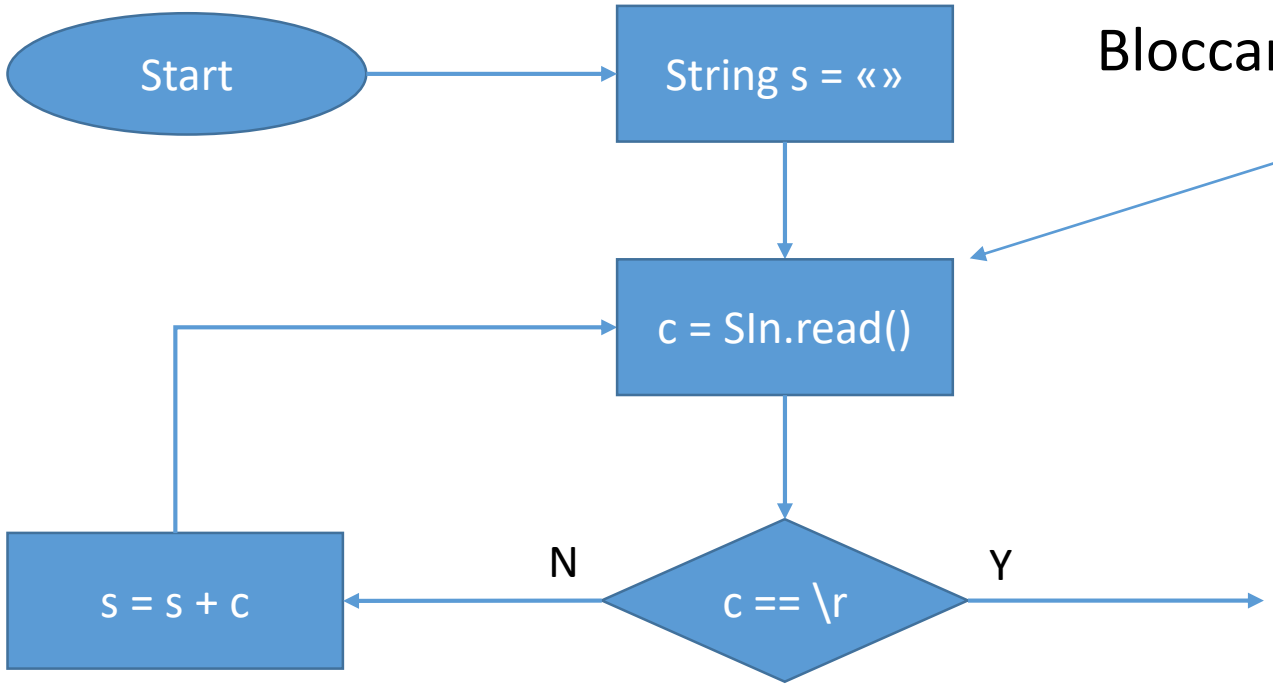


s= «Hello»

Lettura di una stringa – readWord()

\n

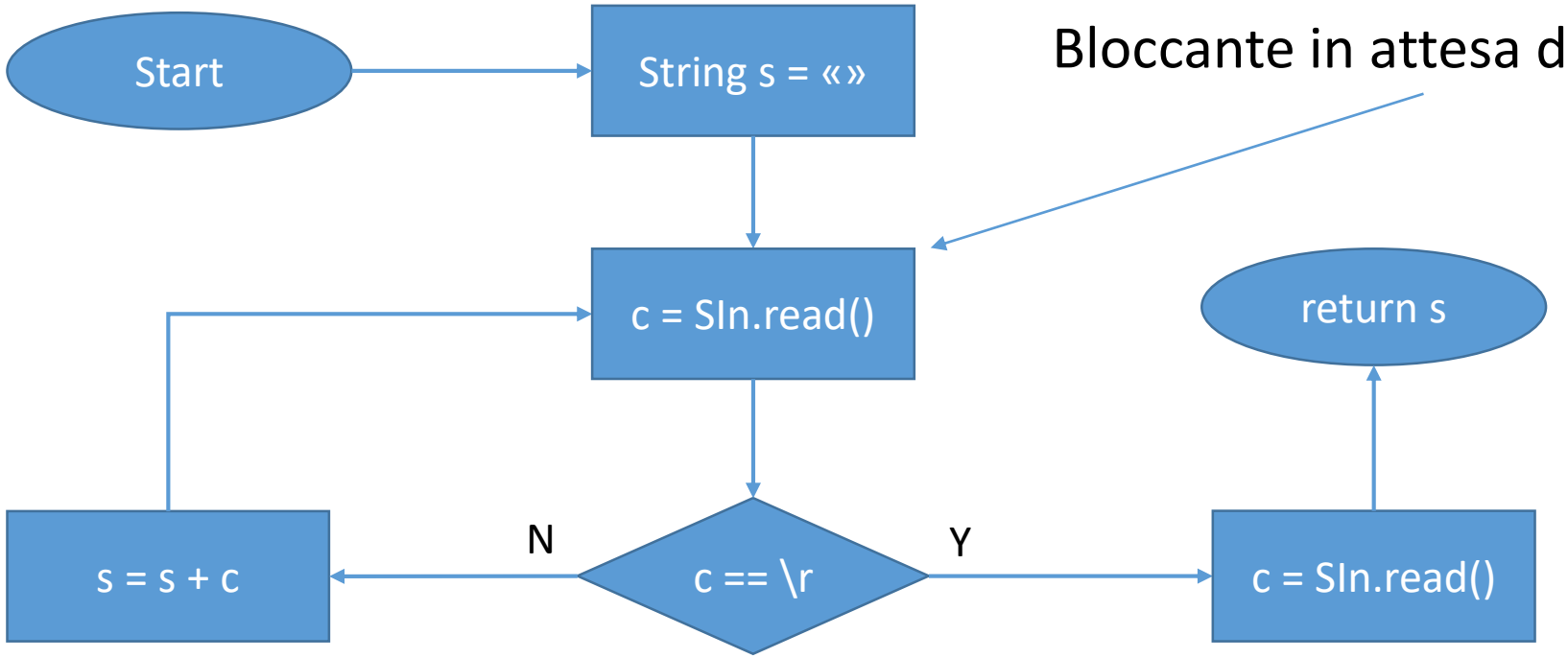
\r



s= «Hello»

Scarto ‘\r’ in *c*

Lettura di una stringa – readWord()

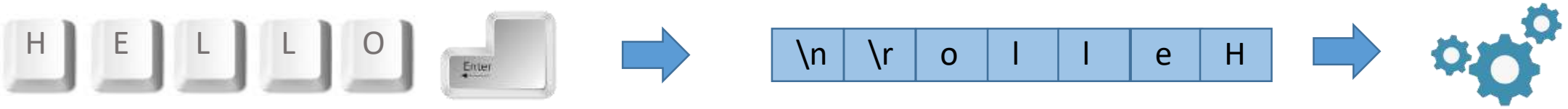


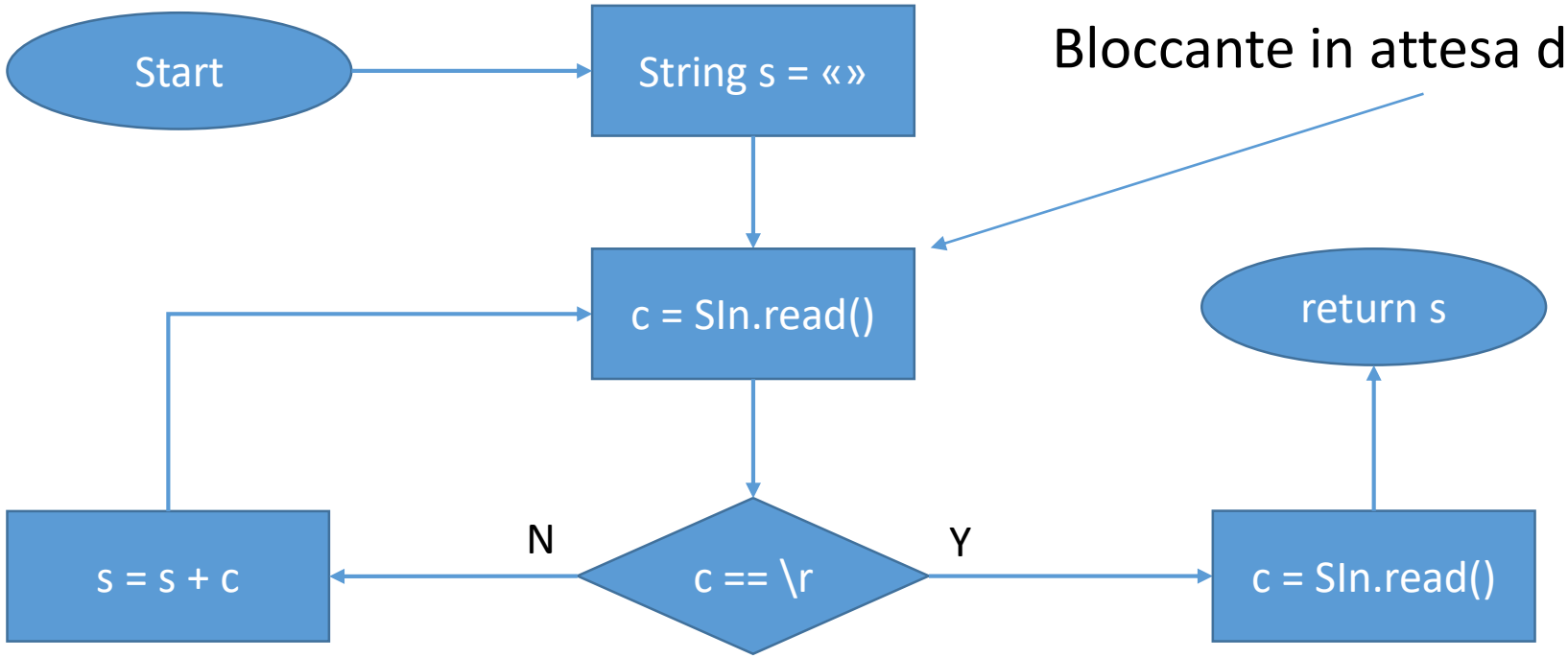
Bloccante in attesa di <Enter>

Estraggo '\n' in *c* e lo scarto

s= «Hello»

Lettura di una stringa – readWord()





Bloccante in attesa di <Enter>

Estraggo e scarto '\n'

Lettura di una stringa – soluzione

```
public static void main (String[] args){
    System.out.print("In attesa di input: ");
    char c = (char)-1;
    String s = "";
    boolean flagCont = true;
    while(flagCont) {
        c = (char)SIn.read();
        if (c == '\r') {
            SIn.read(); //scarto '\n'
            flagCont = false;
        } else {
            s = s + c;
        }
    }
    System.out.println("Stringa risultante: " + s);
}
```

Lettura di una stringa – readWord()

```
public static String readWord() {
    String result = "";
    char next;

    next =  readChar();
    while (Character.isWhitespace(next))
        next =  readChar();

    while (!(Character.isWhitespace(next))) {
        result = result + next;
        next =  readChar();
    }

    if (next == '\r') {
        next = readChar();
        if (next != '\n')
        {
            System.out.println("Fatal Error in method readWord of class SavitchIn.");
            System.exit(1);
        }
    }
    return result;
}
```

Lettura di numeri da tastiera

- readWord() ritorna *stringhe di caratteri*
- I numeri sono stringhe di caratteri a tutti gli effetti
- E' necessario convertire la stringa in *int* o *float*
- Conversione a *int*

```
int sInt =  Integer.valueOf(s).intValue();
```

- Conversione a *float*

```
float sFloat =  Float.valueOf(s).floatValue();
```

Lettura di caratteri da tastiera - 1

- I singoli caratteri immessi sono memorizzati in una coda
- *SIn.read()* é bloccante in attesa di <Enter>
- Premendo <Enter>, *SIn.read()* ritorna il *primo* carattere nella coda
- Chiamate successive a *SIn.read()* ritorneranno i caratteri successivi
- Scartare i caratteri speciali (\n, \r) al termine della coda
- Svuotata la coda, la successiva SIn.read() sarà nuovamente bloccante

Lettura di caratteri da tastiera - 2

- La classe Sin mette a disposizione dei metodi di alto livello per
- Leggere una stringa di testo

```
String s = SIn.readWord();
```

- Leggere un numero intero

```
int sInt =  Integer.valueOf(s).intValue();
```

- Leggere un numero reale

```
float sFloat =  Float.valueOf(s).floatValue();
```

Esercizi

Esercizio: combinazioni di carte

- Un giocatore riceve *k* carte da un mazzo di *n* carte. Si scriva un programma che richieda all'utente di inserire il numero di carte *k* ricevute e calcoli il numero di differenti combinazioni di k carte che può ricevere. Il programma verifichi che l input inserito dall'utente sia corretto (es, k < n) e, se necessario, lo richieda nuovamente finché questo non é corretto.

Esercizio: combinazioni - Approccio

- Il numero di possibili combinazioni di *k* elementi da un set di *n* é dato dal coefficiente binomiale

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- Il fattoriale n! é calcolabile come

$$n! = \prod_{k=1}^{n} k = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$$

Esercizio: combinazioni - Approccio

- Si implementino nella classe Aritmetica i nuovi metodi fattoriale() e binomiale()
- Si sviluppi l‘opportuna classe di test che si occupa di interagire con l’utente tramite i metodi della classe SIn ed esegua il calcolo richiesto
- Si ponga attenzione a che tutti risultati intermedi siano correttamente rappresentabili in formato interoc on segno a 32 bit (int)