

# IL LABIRINTO DEGLI SCACCHI

---

*Una realtà difficile da comprendere*

Esame di stato 2014-2015



# Indice

---

## ❖ Informatica e Tpi :

- Struttura del gioco realizzato in Java
- Design pattern
- Socket
- Polimorfismo ed Override
- Multiplayer su piattaforme diverse

## ❖ Matematica:

- Probabilità
- Esempi

## ❖ Italiano e Storia :

- Le origini
- Brevi citazioni scacchistiche nella letteratura
- Il match del secolo : Boris Spasski VS Boddy Fisher

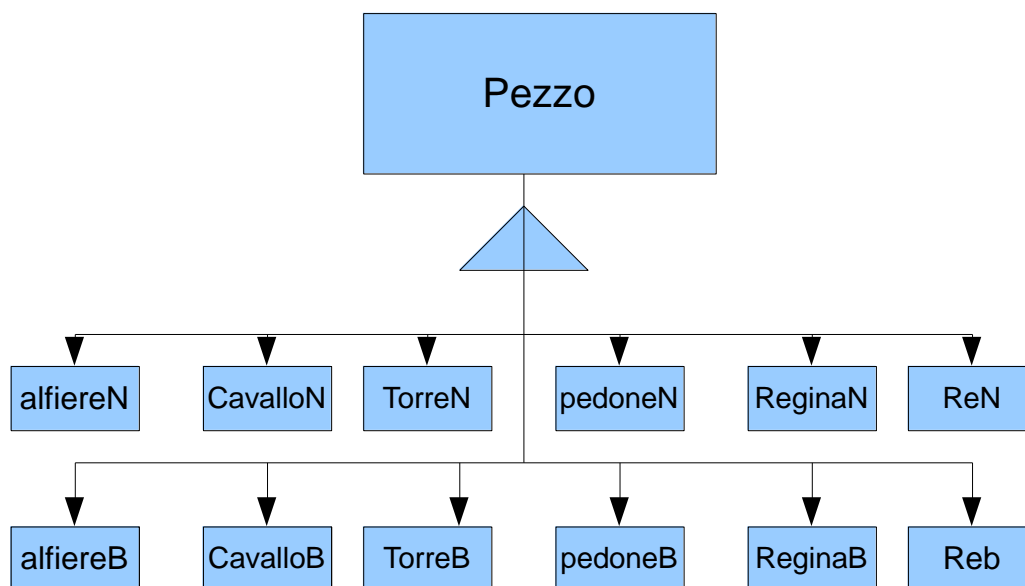
# Informatica & Tpi

Il programma java segue il modello MVC : divide, separa la parte algoritmica dalla parte grafica e la comunicazione, tra questi due ,viene affidato al controller.

Esiste una classe per ciascun pezzo nero e bianco ( pedone, torre, cavallo alfiere, regina e re). In totale ci sono 12 classi ( 6 classi per i pezzi neri e 6 classi per i pezzi bianchi).

C'è una classe madre , che ho chiamato "Pezzo" , dalla quale vengono estese tutti i pezzi elencati sopra. Nella classe Pezzo ci sono i *metodi comuni* ed , inoltre, è la tipologia della matrice del gioco.

La struttura generica è la seguente:



Questa è la classe "Pezzo" :

```
public abstract class Pezzo { // Classe generale (MADRE) ; Abstract= non è possibile fare l'istanza del pezzo
    protected String id; // Nome del pezzo
    protected String idgruppo; // ti dice se è bianco o nero
```

Nella classe Campo viene inizializzato la matrice da gioco 8x8 e nella figura riportata sotto si può notare che la tipologia della matrice è di tipo "Pezzo".

```
private Pezzo mat[][];

//costruttore: matrice 8x8
public Campo(){
    this.mat=new Pezzo[8][8];
    ApplicationContext.diz.put("campo",this);
}
```

Ecco alcune classe figlie estese dalla classe Pezzo :

```

public class alfiereB extends Pezzo {
    private ArrayList<Point>ls=new ArrayList<Point>();

    // costruttore : è un alfiere.
    public alfiereB(){
        super.id="alfiereB"; // prende id dalla classe madre
        super.idgruppo="B";
    }

public class pedoneB extends Pezzo {
    private ArrayList<Point>ls=new ArrayList<Point>();

    // costruttore : è un pedone bianco
    public pedoneB(){
        super.id="pedoneB"; // prende id dalla classe madre
        super.idgruppo="B";
    }
}

```

Il vantaggio principale sono i metodi override ; in questo modo le 2 classi estese diverse possono avere lo stesso nome di un metodo , ma hanno le istruzioni diverse.

Per esempio il metodo setIcona ,che mette un' icona specifica a una pedina .

(Regina)

```

public void setIcona(int riga,int colonna){
    JButton b=(JButton) ApplicationContext.diz.get((riga+colonna*8)+"" );
    if((riga+colonna)%2==0){ // pari chiaro
        b.setIcon(new ImageIcon("ReginaN_sfchiaro00000000.jpg"));
    }
    else { // scuro
        b.setIcon(new ImageIcon("ReginaN_sfscuro00000000.jpg"));
    }
}

```

(Alfiere)

```

public void setIcona(int riga,int colonna){
    JButton b=(JButton) ApplicationContext.diz.get((riga+colonna*8)+"" );
    if((riga+colonna)%2==0){ // pari chiaro
        b.setIcon(new ImageIcon("AlfereB_sfC0000.jpg"));
    }
    else { // scuro
        b.setIcon(new ImageIcon("AfB_sfS0000.jpg"));
    }
}

```

Ora, se vado a mettere le icone tramite la classe Controller, quest' ultimo va prendere il metodo specifico associato alla classe specifica. P.setIcona → se è una regina ,prende setIcona della regina; se è un alfiere prende setIcona del alfiere e così via.

```

public class Controller {

    Campo c;

    public void cambiastato() { // aggiorna la finestra ogni volta che viene richiamato.
        c = (Campo) ApplicationContext.diz.get("campo");

        for(int riga=0;riga!=8;riga++){
            for(int colonna=0;colonna!=8;colonna++){
                Pezzo p = c.getMatrice()[colonna][riga];
                p.setIcona(riga,colonna); // anche qui override! (prende il metodo specifico per ciascun pezzo)
            }
        }
    }
}

```

Il programma può essere implementato grazie ai socket, ottenendo in questo modo la comunicazione di 2 client gestite da un server su macchine diverse. Dal lato server, ho utilizzato i ServerSocket in ascolto su una porta. Il programma funziona in modo multi-threading, ovvero viene istanziata una connessione ogni volta che un client si connette. Attualmente il Server ha solo 3 classi: ServerSocket, Connessioni e Main.

### -Classe ServerSocket:

Questa classe è principale, che è **sempre in ascolto su una porta specifica** quando viene fatto partire il programma.

```

public ServerScacchi() {
    try {
        server = new ServerSocket(2000);
        while (true) {

```

Il suo compito principale è quello di creare le istanze Connessioni quando un client si connette e di farla partire.

Esempio:

```

Socket client = server.accept(); // inizializza un client
connesioneero = new connect(client,1,this); //istanza del client
connesioneero.start(); // faccio partire la connessione
users.add(connesioneero); //aggiungo alla lista le persone connesse
listasocket.add(client);
System.out.println("client " + client.getInetAddress() + " connesso");

```

### -Classe Connessione:

Questa classe rappresenta ciascun client connesso, è una classe che estende la classe Thread. In questo caso ci saranno più connessioni che lavorano contemporaneamente.

```
import java.io.BufferedReader;

class connect extends Thread {
    // dichiarazione delle variabili socket e dei buffer
```

Questo è il costruttore della classe Connect :

```
public connect(Socket client,int id,ServerScacchi s) {
    this.client = client;
    con=true;
    this.id=id;
    this.ServerScacchi=s;
}
```

Quando viene creata l'istanza, vengono dati come parametri i seguenti elementi:

**il socket** del client

**un numero id** ,che rappresenta bianco se è pari, nero se dipari ;

**un boolean con (private)** che viene inizializzata true, indicando che è in azione;

**la classe ServerScacchi** associata.

Questa classe **possiede inoltre il socket del suo nemico ,al quale manderà le mosse** .Siccome è un thread (extends) ha il metodo **run** : esso continua a leggere ed invia al suo nemico i parametri per fare la mossa come nell'esempio seguente:

```
while (con ) {

    pack = in.readLine();
    System.out.println(pack);
    if(this.ServerScacchi.GetIsSocket().size()==2){ // invia se ci sono
        inviaMossa(this.nemico,pack);
    }
}
```

Viene poi richiamato il metodo **inviaMossaAlNemico** che ha come parametri il socket del client nemico e il messaggio da inviare .

```

public void InviaMossaalNemico(Socket clientNemico, String line) {
    try{

        PrintWriter out = new PrintWriter( clientNemico.getOutputStream(), true );
        out.println("Messaggio da :"+clientNemico.getInetAddress()+" "+line);
        out.flush();

    } catch(Exception e)
    {
        System.out.println(e);
    }
}

```

Un esempio di messaggio mandato è la seguente: Torre;4;3;4;7. Il server gestisce esclusivamente il compito di trasporto , sarà il client ad interpretare il messaggio. Il protocollo di interpretazione viene spiegato nella sezione lato client.

### - Classe Main

La classe Main è molto semplice in cui viene istanziata una classe ServerScacchi .

```

public static void main(String[] args) {
    ServerScacchi s = new ServerScacchi();
}

```

## Implementazione lato client

Per poter interagire con il server, il client bisogna avere queste caratteristiche :

- 1) Una classe Player che fa da preambolo di connessione, se questa classe viene inizializzata con successo, ovvero esiste un server che lo ascolta , allora vengono inizializzate tutto resto(campo, graphic, ecc..)

Come prima cosa il server , manda al client una stringa ("B" o "N") che rappresenta il suo genere.

```

GetidFromSever=in.readLine();// è la prima riga che legge dal server!
ApplicationContext.diz.put("getIdfromServer", this.GetidFromSever);

```

Questa verrà poi messa nell'ApplicationContext ,in quanto mi servirà dopo per stabilire a quale segno appartiene il client e solo lui può muovere i pezzi di quel segno.

- 2) Modifiche nella classe eventi:

**premessa:** la classe eventi è una classe che implementa ActionCommand , ovvero un ascoltatore degli eventi dei Jbutton del programma (matrice 8x8 =64 buttoni)

- Viene aggiunto un booleano MioTurno : posso fare le mosse solo se questo è true

- C'è un selezionatore (integer) che identifica il primo click dal secondo click. È un numero che continua a incrementare: dispari → primo click ; pari → secondo click.

Scopo: per ottenere tramite mouse le coordinate (x, y) di partenza e arrivo

```
if(this.selezionatoreClick%2!=0){ //primo click

    this.xpartenza=Integer.parseInt(ev.getActionCommand())/8;
    this.ypartenza=Integer.parseInt(ev.getActionCommand())/8;

    if(this.selezionatoreClick%2==0){ //secondo click
        this.xarrivo=Integer.parseInt(ev.getActionCommand())/8;
        this.yarrivo=Integer.parseInt(ev.getActionCommand())/8;
    }
}
```

Ciascun JButton viene identificato da un numero(da 1 a 64) ; ottengo X facendo la divisione ( / ) e Y facendo il modulo ( % ). Dopo ciò, viene messo il booleano MioTurno false, in quanto ho già fatto la mossa e attendo con pazienza la mossa del mio avversario.

#### *Quando viene mandato il messaggio(MOSSA)?*

Viene mandata la mossa al server **solo ed esclusivamente** se è valida , sul client locale viene spostato il pezzo e vengono inviati questi parametri al server . [Vedi codice]:

- Id pezzo,
- X partenza,
- Y partenza,
- X arrivo,
- Y arrivo

```
boolean ris=Pezzoselezionato.checkMossavalida(this.xtmp, this.ytmp, this.xarrivo, this.yarrivo);
String id = (String) ApplicationContext.diz.get("getIdfromServer"); // Lo prende dal Applicationc, ricevuto id dal server

if( ris==true && Pezzoselezionato.getIdgruppo().equals(id)){ //Condizioni mossa valida : se è valida +se la pedina appartiene al gruppo

    c.spostapezzo(Pezzoselezionato, xtmp, ytmp, xarrivo, yarrivo);

    //invio la mossa al server
    PrintWriter out=(PrintWriter) ApplicationContext.diz.get("out");
    String MossaFatta = Pezzoselezionato.getId()+" "+
        this.xtmp+" "+
        this.ytmp+" "+
        this.xarrivo+" "+
        this.yarrivo+" ";
    out.println(MossaFatta);
}
```

### 3) Viene creata la Classe **Riceve**:

Questa è una classe Thread **che continua a leggere** dal server i messaggi ricevuti. Viene istanziata e messa a run fin dalla connessione, in quanto il client deve ricevere i dati

```
//istanza della parte thread riceve : continua a leggere
Thread t = new Thread(new Riceve());
t.start();
```



Per ogni messaggio ricevuto (che rappresenta una mossa), esso modifica la scacchiera secondo la mossa nemica.

Esempio messaggio: Torre;4;3;4;7;

**interpretazione del messaggio** : Sposta il pezzo torre dal JButton che ha X = 4 Y = 3 al JButton che ha X = 4 Y = 7

Quando questa classe ottiene una mossa dal server, lei entra in azione spostando il pezzo secondo le coordinate (viene sempre richiamato il metodo dal campo "spostaPezzo")

Questo è il codice della classe Ricevi :

```
public void run() {  
    while(true){ // continua a leggere  
        BufferedReader in = (BufferedReader) ApplicationContext.diz.get("in");  
        Campo c=(Campo) ApplicationContext.diz.get("campo");  
        Controller cl= new Controller();  
        try{  
            System.out.println("• Sto Leggendo...");  
            String received = in.readLine(); // è formato in questo modo: "pedone8;6;6;5;6"  
            System.out.println("• Received: "+received);  
  
            //ottengo i vari campi singoli  
            String campi[]=received.split(";");  
            String idpezzo=campi[0];  
            String xstart=campi[1];  
            String ystart=campi[2];  
            String xend=campi[3];  
            String yend=campi[4];  
  
            //trasformazione in numeri  
            int XStartNumero=Integer.parseInt(xstart);  
            int YStartNumero=Integer.parseInt(ystart);  
            int XendNumero= Integer.parseInt(xend);  
            int YendNumero= Integer.parseInt(yend);  
  
            Pezzo p = c.getMatrice()[XStartNumero][YStartNumero].getpezzo();
```

È necessario fare la **procedura dello split** per avere i parametri separati e **in seguito trasformarli in Integer**

```
c.spostaPezzo(p, XStartNumero, YStartNumero, XendNumero, YendNumero);  
//cambio turno  
Eventi ev=(Eventi) ApplicationContext.diz.get("eventi");  
ev.setMioTurnoTrue();  
System.out.print("Il nemico ha fatto la mossa, è il tuo turno!");  
cl.cambiaStato();
```

Infine viene identificato il pezzo ; viene spostato e viene **settato la variabile boolean MioTurno True**, in quanto il nemico ha fatto la mossa.

# Matematica

## Probabilità

La probabilità di un evento è *il rapporto tra il numero dei casi favorevoli all'evento e il numero dei casi possibili, purché questi ultimi siano tutti equiprobabili.*

Indicando con  $\Omega$  l'insieme di casi possibili e con  $|\Omega|=n$  la sua cardinalità, con  $A$  un evento e con  $n_A$  il numero dei casi favorevoli ad  $A$  (ad esempio, nel lancio di un dado  $\Omega=\{1,2,3,4,5,6\}$ ,  $n=6$ ,  $A$  = "numero pari",  $n_A=3$ ), la probabilità di  $A$ , indicata con  $P(A)$ , è pari a:

$$P(A) = \frac{n_A}{n} = \frac{3}{6} = \frac{1}{2}$$

Negli scacchi, è possibile determinare dei casi probabilistici, partendo dal più semplice, per esempio la probabilità che lo scacco sia fatto da una torre.

Quindi :

$\Omega$ ={tutti pezzi disponibili del giocatore},  $n=12$ ,  $A$  = "scacco fatto da una torre",  $n_A=2$ , la probabilità di  $A$ , indicata con  $P(A)$ , è pari a:  $P(A)=2/12 \rightarrow 1/6 \rightarrow 16.6\%$

Un altro esempio più complicato è la seguente:

**Qual è la probabilità che due regine poste a caso sulla scacchiera siano "indipendenti" (non si attaccino) ?**

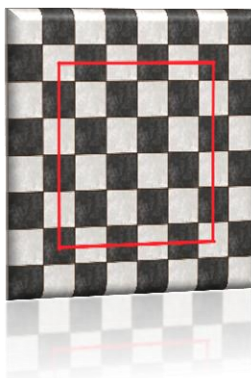
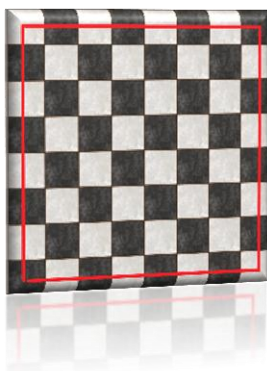
**Soluzione:** Usiamo la formula della Probabilità Totale:

$$P(I) = \sum_{i=1}^4 P(I|A_i)P(A_i)$$

dove  $A_i$  = la regina  $A$  si trova sull'  $i$ -esimo bordo.

Nel primo bordo ci sono **28** celle  $\rightarrow$  quindi  $P(A) = 28/64$

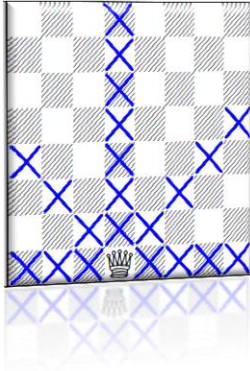
Nel secondo bordo ci sono **20** celle : quindi  $P(B) = 20/64$  e così via.



$P(I|A_1) \rightarrow$  è la probabilità di indipendenza nel bordo A1 (cioè nel primo), che è uguale a **42/63**.

Questo risultato è ottenuto nel modo seguente:

quando la regina A si trova nel bordo A1, essa può andare in 21 celle diverse nella scacchiera



Quindi la regina B, per essere indipendente dalla regina A (cioè che non si attacchino), non deve esserci in queste 21 posizioni e anche nella cella in cui si trova la regina A ( **$64 - 21 - 1 = 42$** ).

Per ogni bordo viene calcolata la probabilità di indipendenza condizionata a quel bordo e infine viene usato la formula della probabilità totale.

- $P(A_1) = \frac{28}{64}, P(I|A_1) = \frac{42}{63};$
- $P(A_2) = \frac{20}{64}, P(I|A_2) = \frac{40}{63};$
- $P(A_3) = \frac{12}{64}, P(I|A_3) = \frac{38}{63};$
- $P(A_4) = \frac{4}{64}, P(I|A_4) = \frac{36}{63};$

Quindi:

$$P(I) = \frac{28}{64} \frac{42}{63} + \frac{20}{64} \frac{40}{63} + \frac{12}{64} \frac{38}{63} + \frac{4}{64} \frac{36}{63} = \frac{2576}{4032} \approx 0.639$$

# Italiano & Storia

---

## 1. Le origini del gioco

Gli scacchi sono un gioco da tavolo di strategia, che vede opposti due avversari, detti Bianco e Nero. Viene giocato su una tavola quadrata, detta scacchiera, composta da 64 caselle di due colori alternanti e contrastanti, sulla quale all'inizio si trovano trentadue pezzi, sedici per ciascun colore: un re, una regina, due alfieri, due cavalli, due torri e otto pedoni. Lo scopo del gioco consiste di fare lo scacco matto, cioè attaccare il re avversario senza che quest'ultimo abbia la possibilità di sfuggire. L'etimologia del gioco deriva da "Shah", una parola persiana che significa re. Il gioco è di origine indiana nata nel VI secolo e giunto poi in Europa attorno all'anno 1000.

Gli scacchi vengono considerati anche uno sport riconosciuto dal Comitato Olimpico Internazionale e le competizioni ufficiali sono organizzate dalla FIDE (Fédération Internationale des Échecs, it. Federazione Internazionale degli Scacchi), fondata nel 1924.

### Il valore dei pezzi

A ogni pezzo degli scacchi viene attribuito un numero indicativo che rappresenta il peso strategico durante una partita. Alla regina viene attribuita un valore numerico di 10, alla torre 5, all'alfiere e al cavallo 3, e al pedone 1.

Pezzo	Donna	Torre	Alfiere	Cavallo	Pedone
					
Valore	9/10	5	3	3	1

## 2. Eugenio Montale e “Le nuove stanze”

Eugenio Montale, uno dei massimi poeti italiani, nacque a Genova il 12 ottobre del 1896. Le sue raccolte di poesie più famose sono: Ossi di seppia, Le Occasioni, La bufera e altro, Satura. Fu l'alfiere della corrente poetica dell'ermetismo, a cui appartengono anche Quasimodo e Ungaretti, con i quali forma una triade di spicco nella Poesia italiana del '900.

Il tema degli scacchi viene citato in una delle sue poesie, più precisamente in “Le nuove stanze”.

### “Le nuove stanze”

Poi che gli ultimi fili di tabacco  
al tuo gesto si spengono nel piatto  
di cristallo, al soffitto lenta sale  
la spirale del fumo  
che **gli alfieri e i cavalli degli scacchi**  
guardano stupefatti; e nuovi anelli

la seguono, più mobili di quelli  
delle tue dita.

La morgana che in cielo liberava  
torri e ponti è sparita  
al primo soffio; s'apre la finestra  
non vista e il fumo s'agita. Là in fondo,  
altro stormo si muove: una tregenda  
d'uomini che non sa questo tuo incenso,  
nella **scacchiera** di cui puoi tu sola  
comporre il senso.

Il mio dubbio d'un tempo era se forse  
tu stessa ignori **il giuoco che si svolge**  
**sul quadrato** e ora è nembo alle tue porte:  
follia di morte non si placa a poco  
prezzo, se poco è il lampo del tuo sguardo  
ma domanda altri fuochi, oltre le fitte  
cortine che per te fomenta il dio  
del caso, quando assiste.

Oggi so ciò che vuoi; batte il suo fioco  
tocco la Martinella ed impaura  
le sagome d'avorio in una luce  
spettrale di nevaio. Ma resiste  
e vince il premio della solitaria  
veglia chi può con te allo specchio ustorio  
che accieca le pedine opporre i tuoi  
occhi d'acciaio.

“Stanze” è il titolo di una precedente composizione della raccolta *Le Occasioni*, “Nuove stanze” è una delle poesie conclusive della raccolta, una delle più pregnanti in essa contenute. Il componimento apparve nel maggio del 1939, quando l’ombra adunca del secondo conflitto mondiale cominciava già a proiettarsi minacciosa sul mondo.

Il poeta e Clizia giocano a scacchi in un interno e nella prima strofa l’attenzione si concentra in particolare su due aspetti della donna: il gesto di spegnere nel portacenere la sigaretta e la presenza di numerosi anelli alle dita. Gli attributi di Clizia si rivelano già in questa strofa come l’ espressione di potere: gli anelli alle mani evocano una ricca simbologia di incantesimi; e non a caso la figura dell’anello si trasmette dalla mano di Clizia, che compie il gesto di spegnere la sigaretta, alle spire di fumo che se ne sprigionano, con un parallelismo sottolineato esplicitamente dal poeta. Queste figure di fumo costituiscono una vera e propria magia operata da Clizia, diventano la rappresentazione della realtà esterna costruendo nella stanza una città ideale: si allude all’apparente controllo che la cittadella della cultura può esercitare sulla vera città degli uomini. Ma la realtà esterna incalza: la finestra si apre e il vento della storia cancella quel miraggio scompigliando il fumo sul quale esso era costruito. La realtà esterna è la percezione della guerra, alla quale partecipano gli uomini ignari di Clizia e perciò ignari del significato della propria condizione. Appare troppo forte la differenza di forze tra la violenza degli eserciti e lo sguardo di Clizia; cioè la bellezza della donna inerme rispetto all’incalzare della guerra e delle barbarie, per frenare le quali sarebbero necessarie altre forze.

L’ultima strofa contiene però una risposta positiva a questi dubbi, infatti, all’avvicinarsi del pericolo, segnalato dal suono della campana, i pezzi degli scacchi, cioè gli uomini comuni coinvolti nei processi della storia ma ignari del loro significato, si spaventano e vengono travolti; invece chi è unito a Clizia e può contare sullo sguardo di lei (come il poeta) è in grado di resistere e sopravvivere, intellettualmente, alla

catastrofe, conservando la possibilità di vedere il significato delle cose senza essere accecato dall'apparente insensatezza e dalla brutalità della storia.

Tutto il componimento è basato su emblemi allegorici: è allegorico il gioco degli scacchi che assume un doppio valore: da un lato rappresenta una guerra simulata riproducendo simbolicamente la scacchiera dei campi di battaglia, dall'altro è il gioco dell'intelligenza e della cultura e dunque si adatta bene al personaggio di Clizia; è allegorico il contrasto interno\ esterno che fa coincidere al primo il valore, la condizione privilegiata di pochi eletti guidati dallo sguardo freddo e implacabile di Clizia, e al secondo il disvalore, le ignare "pedine" travolte sulla scacchiera della storia; è allegorico anche il personaggio della donna-angelo messaggera dei valori, ma il contenuto del messaggio non è di tipo religioso: la religione di Clizia è quella della cultura e dell'umanesimo. Per questo si può parlare di un allegorismo umanistico.



(fonte <http://treviglioscacchi.altervista.org/> )

### 3. Il match del secolo

Il campionato del mondo di scacchi del 1972, passato alla storia con l'appellativo di incontro del secolo, venne disputato tra il detentore del titolo Boris Spasskij (russo) e lo sfidante Bobby Fischer(americano). Giocato a Reykjavík, in Islanda, tra l'11 luglio e il 3 settembre, all'apice della guerra fredda, la sfida è probabilmente la più famosa della storia delle competizioni ufficiali di scacchi, nonché una delle più movimentate e drammatiche di tutti i tempi.

Per lungo tempo il sistema scacchistico sovietico aveva avuto il monopolio nel gioco ad alto livello, in particolare nei campionati mondiali (tutti i match per il campionato del mondo successivi alla seconda guerra mondiale erano stati giocati tra due sovietici), nel 1970, tuttavia, il campionato fu vinto dall'americano Bobby Fisher concludendo 12,5 - 8,5 .

Riepilogo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Totale
Boris Spassky (  URSS)	1	1	0	½	0	0	½	0	½	0	1	½	0	½	½	½	½	½	½	½	0	-	-	-	8,5
Bobby Fischer (  Stati Uniti)	0	-	1	½	1	1	½	1	½	1	0	½	1	½	½	½	½	½	½	½	1	-	-	-	12,5

La ventunesima partita fu l'ultima, Spasskij annunciò il proprio abbandono per telefono al giudice dell'incontro.



(Fisher a sinistra e Spasskij a destra)