Corso di Programmazione e Algoritmi 2
Modulo di Laboratorio
A.A. 2020/2021

Regole per lo svolgimento del progetto:

- Il progetto può essere svolto **singolarmente** o **in gruppi di qualsiasi dimensione**.
  In ogni caso, ciascuno studente dovrà poi **singolarmente** discutere il progetto dimostrando di essere in grado di modificare e/o aggiungere alcune parti. In caso di esito sufficiente di tale discussione, **verrà attribuito il punteggio da 0 a 3 punti (fatta eccezione per il caso illustrato all'inizio di pagina 2)**
- Il progetto deve essere consegnato **almeno 7 giorni prima** della prova orale di *Programmazione e Algoritmi 2* relativa all'appello in cui si desidera discutere il progetto.
- Il progetto deve contenere le classi, i campi e i metodi richiesti **rispettandone esattamente il nome, il tipo e l'ordine dei parametri formali, ed il tipo di ritorno**.
- Si tenga presente che **non sono sempre presenti** tutti i modificatori (***public, private, protected, static, final, …***) che devono essere opportunamente aggiunti da voi nella consegna.
- I **metodi di accesso e modifica** dei vari parametri non sono tutti presenti nella specifica. Deve essere vostra cura aggiungerli in modo opportuno.
- Si è naturalmente liberi di sviluppare (e anzi siete incoraggiati a farlo) metodi aggiuntivi di supporto ai metodi richiesti, laddove lo si ritenga utile.
- Deve essere consegnato via email all'indirizzo **luca.moscardelli@unich.it** un file compresso contenente tutti i file sorgenti e avente come nome
  [java] progetto 2020_2021.zip
  l'email deve avere come oggetto
  [java] consegna progetto programmazione 2020/2021
  e deve contenere nel testo la lista (Cognome, Nome e Matricola) di tutti gli studenti che hanno svolto il progetto consegnato.
  In caso di consegna in gruppo, la consegna deve essere effettuata da uno dei componenti del gruppo, mettendo in CC nell'email anche l'indirizzo di tutti gli altri componenti.
- Tutte le classi (tranne eventualmente quelle astratte) devono sovrascrivere il metodo **`toString()`** in modo opportuno
- Non è necessario consegnare il metodo **`main`** (che voi potete chiaramente utilizzare per i vostri test)
- L'unica libreria del Java che potete usare è **`java.util.ArrayList`** (oltre naturalmente alle classi del package **`java.lang`**)

Testo del progetto:

Sviluppare in **Java** un package **cleii.scacchi** per la rappresentazione ed esecuzione di partite di scacchi.

Per le regole (semplificate) del gioco degli scacchi **si faccia riferimento alle ultime pagine di questo pdf**. Sole le regole presenti in tale file (e non sbarrate) devono essere tenute in conto nella stesura del presente progetto. **Non** è quindi richiesta la verifica della triplice ripetizione di posizione, e la verifica delle cosiddette 50 mosse (che sarebbero presenti nel regolamento completo del gioco); chi è interessato a implementarle entrambe, potrà raggiungere un bonus totale di **4 punti**, ma solo svolgendo il progetto singolarmente.

Il package deve contenere le seguenti classi:

- `Scacchiera`

  Classe pubblica che rappresenta una scacchiera di 64 caselle. Ogni casella deve essere capace di ospitare un pezzo. Possiede un costruttore senza parametri che crea una scacchiera con i pezzi nella configurazione iniziale.

  In tutti i metodi, se non diversamente specificato, in modo simile a quanto si usa fare nella cosiddetta notazione numerica (https://it.wikipedia.org/wiki/Notazione_numerica), le case della scacchiera sono indicate con un numero intero a due cifre come di seguito riportato:

18	28	38	48	58	68	78	88
17	27	37	47	57	67	77	87
16	26	36	46	56	66	76	86
15	25	35	45	55	65	75	85
14	24	34	44	54	64	74	84
13	23	33	43	53	63	73	83
12	22	32	42	52	62	72	82
11	21	31	41	51	61	71	81

  Le colonne, come le righe (traverse), sono numerate da 1 a 8. Nella notazione numerica tutte le case sono individuate da una coppia di cifre, la prima descrive la colonna (dalla sinistra alla destra del Bianco) e la seconda la riga (o traversa, dalla prima traversa in basso del Bianco all'ultima).

  Contiene i seguenti metodi:

  - `Pezzo get (int pos)`
    che restituisce il pezzo in posizione pos, null se tale posizione è libera.
  - `int getPos (Pezzo p)`
    che restituisce la posizione del pezzo p. Restituisce 0 se tale pezzo non è presente sulla scacchiera.
  - metodo `toString` che deve rappresentare la scacchiera nella seguente modalità testuale (nell'esempio è riportata la posizione iniziale: si noti che le maiuscole identificano i pezzi del bianco e le minuscole quelli del nero; le case vuote sono indicate da un punto):

```
r n b q k b n r
p p p p p p p p
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
P P P P P P P P
R N B Q K B N R
```

- `Stato`

  Classe pubblica che rappresenta lo stato del gioco. Lo stato del gioco è identificato da tutte le informazioni necessarie per codificare:
  - la scacchiera
  - il giocatore che deve fare la prossima mossa
  - le possibilità della mossa di arrocco per il bianco
  - le possibilità della mossa di arrocco per il nero
  - le possibilità della cattura "en passant" per il bianco
  - le possibilità della cattura "en passant" per il nero

  Contiene i seguenti metodi:
  - `boolean sottoAttacco (int pos, boolean white)`
    che restituisce `true` se e solo se la casa `pos` della scacchiera è sotto attacco da parte di un qualsiasi pezzo bianco (qualora il parametro `white` valga true) o nero (qualora il parametro `white` valgo false). Si noti che `pos` deve essere una casa vuota o contenente un pezzo di colore diverso dal colore che pone sotto attacco.
  - `boolean scacco ()`
    che restituisce `true` se e solo se il re del giocatore di turno (che deve fare la prossima mossa) si trova sotto scacco, cioè in una casa sotto attacco.
  - `boolean scaccoMatto ()`
    che restituisce `true` se e solo se il re del giocatore di turno (che deve fare la prossima mossa) si trova sotto scacco e il giocatore di turno non ha nessuna mossa valida a disposizione. *In altre parole, tutti gli spostamenti potenziali di un qualsiasi suo pezzo o le catture da esso realizzate non possono realizzare una mossa valida in quanto lascerebbero il proprio re sotto scacco.*
  - `boolean stallo ()`
    che restituisce `true` se e solo se il giocatore di turno (che deve fare la prossima mossa) non ha nessuna mossa valida a disposizione e non ha il proprio re attualmente sotto scacco. *In altre parole, tutti gli spostamenti potenziali di un qualsiasi suo pezzo o le catture da esso realizzate non possono realizzare una mossa valida in quanto metterebbero il proprio re sotto scacco.*
  - `Stato simulaSpostamentoOCattura (int from, int to, int promozione)`
    che restituisce un nuovo stato risultante dalla mossa del giocatore di turno dalla casa `from` alla casa `to`, se
    - nella casa from è presente un pezzo del giocatore individuato dal parametro `white`
    - tale mossa rientra tra gli *spostamenti potenziali* del pezzo in questione, oppure è diretta verso una casa, *sotto attacco* dal pezzo in questione, che contiene un pezzo del giocatore avversario.

    Altrimenti restituisce `null`.
    Il parametro `promozione` è usato unicamente nel caso in cui si tratti di uno spostamento del pedone che raggiunge la traversa più lontana dalla sua posizione inziale e indica il pezzo di promozione del pedone: (0=regina, 1=cavallo, 2=alfiere, 3=torre).
  - `Stato simulaSpostamentoOCattura (int from, int to)`
    che richiama e restituisce `simulaSpostamentoOCattura (from, to, 0)`

- `boolean mossaValida (int from, int to, int promozione)`
  che simula la mossa del giocatore di turno dalla casa `from` alla casa `to` e restituisce `true` se e solo se tale mossa è valida.
  Il parametro `promozione` è usato unicamente nel caso in cui la mossa corrisponde a uno spostamento del pedone che raggiunge la traversa più lontana dalla sua posizione inziale e indica il pezzo di promozione del pedone: (0=regina, 1=cavallo, 2=alfiere, 3=torre).
  Per essere valida, devono valere entrambe le seguenti condizioni:
  - `simulaSpostamentoOCattura (from, to, promozione)` non deve restituire `null`;
  - lo Stato restituito da `simulaSpostamentoOCattura (from, to, promozione)` non deve essere tale che il proprio re si trovi in una situazione di scacco (ovvero si trovi in una casa sotto attacco da parte di un pezzo del giocatore avversario).
- `boolean mossaValida (int from, int to)`
  che richiama e restituisce `mossaValida (from, to, 0)`
- `boolean eseguiMossa (int from, int to, int promozione)`
  che modifica lo stato in modo da eseguire la mossa del giocatore di turno dalla casa `from` alla casa `to`, e restituisce `true`, se tale mossa è valida; altrimenti lascia invariato lo stato e restituisce `false`.
  Il parametro `promozione` è usato unicamente nel caso in cui la mossa corrisponde a uno spostamento del pedone che raggiunge la traversa più lontana dalla sua posizione inziale e indica il pezzo di promozione del pedone: (0 = regina, 1= cavallo, 2=alfiere, 3=torre).
- `boolean eseguiMossa (int from, int to)`
  che richiama e restituisce `eseguiMossa (from, to, 0)`

- `Pezzo (classe astratta)`

Classe pubblica per rappresentare i pezzi che possono essere posizionati sulla scacchiera.
Contiene un parametro che specifica se il pezzo è bianco o nero, e i seguenti metodi **astratti**:

- `abstract boolean spostamentoPotenziale (Stato s, int target);`
  che restituisce `true` se e solo se il pezzo può muovere nello stato `s` dalla propria casa alla casa `target` (che deve essere libera in `s`). Si tenga presente che negli spostamenti del re bisogna considerare anche l'eventuale arrocco (che invece non va considerato tra gli spostamenti della torre). Tale metodo non tiene conto di eventuali situazioni di scacco (ai danni del proprio re causato dal movimento in questione), che rendono lo spostamento non realizzabile in una mossa valida.
- `ArrayList<Integer> listaSpostamentoPotenziale (Stato s)`
  che restituisce un arraylist contenente tutte e sole le posizioni della scacchiera verso le quali il pezzo può muovere a partire dallo stato `s`. Si noti che le posizioni restituite devono corrispondere a una casa libera in `s` e che negli spostamenti del re bisogna considerare anche l'eventuale arrocco (che invece non va considerato tra gli spostamenti della torre). Tale metodo non tiene conto di eventuali situazioni di scacco (ai danni del proprio re causato dal movimento in questione), che rendono lo spostamento non realizzabile in una mossa valida.
  Si noti infine che, sebbene tale metodo possa essere implementato direttamente nella

classe Pezzo, può avere senso sovrascriverlo nelle sottoclassi in modo da renderne più efficiente l'implementazione.

- `abstract boolean attacco (Stato s, int target);`
  che restituisce `true` se e solo se nello stato `s` il pezzo pone sotto attacco la casa `target`. Si noti che `target` deve essere libero oppure occupato da un pezzo avversario. A proposito di questo ultimo caso, si tenga presente che tale metodo non tiene conto di eventuali situazioni di scacco (ai danni del proprio re causato dalla cattura in questione), che rendono la cattura non realizzabile in una mossa valida.
- `ArrayList<Integer> listaAttacco (Stato s)`
  che restituisce un arraylist contenente tutte e sole le posizioni della scacchiera che sono sotto attacco da parte del pezzo nello stato `s`. Si noti che le posizioni restituite devono corrispondere a una casa libera oppure occupata da un pezzo avversario. Si noti infine che, sebbene tale metodo possa essere implementato direttamente nella classe Pezzo, può avere senso sovrascriverlo nelle sottoclassi in modo da renderne più efficiente l'implementazione.

Tale classe astratta ha le seguenti classi pubbliche che la estendono per rappresentare i vari pezzi del gioco:

- `Pedone`
- `Torre`
- `Alfiere`
- `Cavallo`
- `Regina`
- `Re`

- `Partita`

Classe **pubblica** per rappresentare una partita del gioco. Contiene i campi per memorizzare:

- lo stato corrente del gioco;
- le mosse della partita;
- l'eventuale esito della partita (in corso, vittoria del bianco, vittoria del nero, patta)

Contiene i seguenti metodi:

- Un costruttore senza parametri che crea una partita senza mosse.
- `public void eseguiMossa (int from, int to, int promozione) throws EccezioneMossa`
  che, qualora la partita sia in corso, modifica lo stato in modo da eseguire la mossa del giocatore di turno dalla casa `from` alla casa `to`, e memorizza la mossa nell'opportuna struttura dati, se tale mossa è valida; altrimenti **solleva un'eccezione** `EccezioneMossa` **(controllata) di mossa non valida**.
  Si tenga presente che per effettuare una mossa di arrocco i parametri `from` e `to` si riferiscono allo spostamento del re.
  Se la mossa eseguita conduce in uno stato di *stallo*, la partita raggiunge l'esito di patta.
  Se la mossa eseguita conduce in uno stato di *scacco matto*, la partita raggiunge l'esito di vittoria dell'opportuno giocatore.
  Il parametro `promozione` è usato unicamente nel caso in cui la mossa corrisponde a uno spostamento del pedone che raggiunge la traversa più lontana dalla sua posizione inziale e indica il pezzo di promozione del pedone: (0=regina, 1=cavallo, 2=alfiere, 3=torre).
- `public void eseguiMossa (int from, int to) throws EccezioneMossa`
  che richiama `eseguiMossa (from, to, 0)`

- `public void abbandona()`
  che, qualora la partita sia in corso, causa l'abbandono del giocatore di turno, assegnando quindi la vittoria al suo avversario.
- `public boolean inCorso()`
  che restituisce `true` se e solo se la partita è attualmente in corso
- `public boolean vittoriaBianco()`
  che restituisce `true` se e solo se la partita è terminata con la vittoria del bianco
- `public boolean vittoriaNero()`
  che restituisce `true` se e solo se la partita è terminata con la vittoria del nero
- `public boolean patta()`
  che restituisce `true` se e solo se la partita è terminata come patta.

- `EccezioneMossa`
  Classe pubblica per rappresentare l'opportuna eccezione in caso di mossa non valida.

REGOLE DI BASE DEL GIOCO DEGLI SCACCHI

**Queste sono le regole di base del gioco degli scacchi.
Per giocare nei tornei occorre seguire anche altre regole comportamentali ed agonistiche.
Il regolamento completo del gioco degli scacchi si può consultare a questo indirizzo:
http://www.federscacchi.it/str_reg.php?tipo=3**

LA NATURA E GLI OBIETTIVI DELLA PARTITA DI SCACCHI

La partita di scacchi si gioca tra due avversari che muovono i propri pezzi su una tavola quadrata detta 'scacchiera'.

Il giocatore che ha i pezzi di colore chiaro (Bianco) esegue la prima mossa; quindi i giocatori muovono alternativamente, con il giocatore che ha i pezzi di colore scuro (Nero) che esegue la mossa successiva.

L'obiettivo di ciascun giocatore è porre il Re avversario 'sotto attacco' in maniera tale che l'avversario non abbia più alcuna mossa legale.
Il giocatore che raggiunge questo risultato è detto aver dato 'scaccomatto' al Re avversario ed ha vinto la partita.

Non è consentito lasciare il proprio Re sotto attacco, esporre il proprio Re all'attacco e nemmeno 'catturare' il Re avversario.

L'avversario il cui Re sia stato posto in scaccomatto ha perso la partita.

Se la posizione è tale che nessuno dei due giocatori possa in alcun modo dare scaccomatto al Re avversario, la partita è patta.

LA POSIZIONE INIZIALE DEI PEZZI SULLA SCACCHIERA

La scacchiera è composta da una griglia 8 x 8 di 64 case uguali, alternativamente chiare (case 'bianche') e scure (case 'nere').

La scacchiera è collocata tra i giocatori in modo che la casa posta nell'angolo in basso alla destra del giocatore sia bianca.

All'inizio della partita il Bianco ha 16 pezzi di colore chiaro (i pezzi 'bianchi'); il Nero ha 16 pezzi di colore scuro (i pezzi 'neri').

I pezzi sono i seguenti:

un Re bianco	solitamente indicato con il simbolo	R
una Donna bianca	solitamente indicata con il simbolo	D
due Torri bianche	solitamente indicate con il simbolo	T
due Alfieri bianchi	solitamente indicati con il simbolo	A
due Cavalli bianchi	solitamente indicati con il simbolo	C
otto pedoni bianchi	solitamente indicati con il simbolo	
un Re nero	solitamente indicato con il simbolo	R
una Donna nera	solitamente indicata con il simbolo	D
due Torri nere	solitamente indicate con il simbolo	T
due Alfieri neri	solitamente indicati con il simbolo	A
due Cavalli neri	solitamente indicati con il simbolo	C
otto pedoni neri	solitamente indicati con il simbolo	

Pezzi Staunton

(p) D R A C T

La posizione iniziale dei pezzi sulla scacchiera è la seguente:

Le otto file verticali di case sono dette 'colonne'. Le otto righe orizzontali di case sono dette 'traverse'. Una linea retta di case dello stesso colore che vada da un lato della scacchiera al lato adiacente è detta 'diagonale'.

IL MOVIMENTO DEI PEZZI

Non è permesso muovere un pezzo su una casa occupata da un pezzo dello stesso colore.

La cattura di un pezzo avversario

Se un pezzo viene mosso su una casa occupata da un pezzo avversario, quest'ultimo viene catturato e rimosso dalla scacchiera come parte della stessa mossa.

Si dice che un pezzo attacca un pezzo avversario se il pezzo può effettuare una cattura su quella casa.

Si considera che un pezzo attacchi una casa anche qualora quel pezzo sia impossibilitato a muoversi verso quella casa perché così facendo lascerebbe o porrebbe sotto attacco il Re del proprio colore.

Alfiere

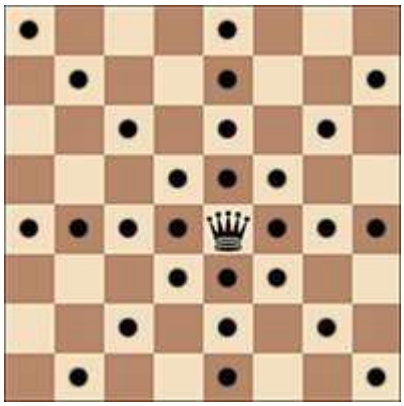
L'Alfiere si può muovere su una qualunque casa lungo una diagonale su cui si trova.

Torre

La Torre si può muovere su una qualunque casa lungo la colonna o la traversa sulle quali si trova.

Donna

La Donna si può muovere su una qualunque casa lungo la colonna, la traversa o una diagonale sulle quali si trova.



Nota:

Nell'effettuare queste mosse, l'Alfiere, la Torre e la Donna non possono scavalcare alcun pezzo interposto.

Cavallo

Il Cavallo si può muovere su ciascuna delle case più vicine a quella sulla quale si trova ma non poste sulla stessa colonna, traversa o diagonale.

Pedone

Il pedone si può muovere sulla casa ad esso immediatamente successiva sulla stessa colonna, a condizione che detta casa non sia occupata, ovvero …

alla sua prima mossa, il pedone si può muovere come appena detto oppure, in alternativa, può avanzare di due case lungo la stessa colonna, a condizione che dette case non siano occupate, ovvero …

Inoltre il pedone si può muovere su una casa posta diagonalmente di fronte ad esso su una colonna adiacente, occupata da un pezzo avversario, catturando il pezzo.

Presa en passant

Un pedone che occupi la casa nella stessa traversa e sulla colonna adiacente di un pedone avversario il quale sia appena stato avanzato di due case dalla sua casa d'origine, può catturare il pedone avversario come se quest'ultimo fosse stato avanzato di una sola casa. Questa cattura è legale solo nella mossa immediatamente successiva al suddetto avanzamento ed è detta cattura 'en passant ' ('al varco').

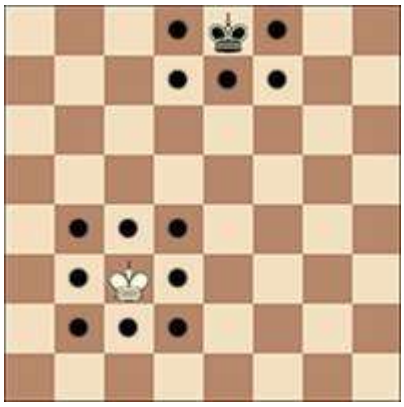
Promozione

Quando un giocatore avanza un pedone alla traversa più lontana dalla sua posizione iniziale, come parte integrante della stessa mossa deve scambiare quel pedone con una nuova Donna, Torre, Alfiere o Cavallo dello stesso colore, ponendolo sulla prevista casa di destinazione. Quest'ultima è detta 'casa di promozione'. La scelta del giocatore non è limitata ai pezzi catturati in precedenza. Lo scambio del pedone con un altro pezzo è detto promozione, e l'effetto del nuovo pezzo è immediato.

Re

Ci sono due diversi modi di muovere il Re:

muovendolo su una casa adiacente …



… oppure ***'arroccando'.*** Questa è una mossa del Re e di una Torre dello stesso colore lungo la prima traversa del giocatore, che conta come una singola mossa del Re ed è eseguita come segue: dalla sua casa di origine, il Re viene trasferito due case verso la Torre che si trova ancora nella sua casa d'origine; quindi la Torre stessa viene trasferita sulla casa che il Re ha appena attraversato.

Prima dell'arrocco corto del Bianco
Prima dell'arrocco lungo del Nero

Dopo l'arrocco corto del Bianco
Dopo l'arrocco lungo del Nero

Prima dell'arrocco lungo del Bianco
Prima dell'arrocco corto del Nero

Dopo l'arrocco lungo del Bianco
Dopo l'arrocco corto del Nero

Il diritto all'arrocco è perduto:

- se il Re è già stato mosso, ovvero …
- con una Torre che sia già stata mossa.

Il diritto all'arrocco è temporaneamente sospeso:

- se la casa su cui il Re si trova o la casa che deve attraversare o la casa che deve occupare sono attaccate da uno o più pezzi avversari, ovvero …
- se tra il Re e la Torre con la quale l'arrocco deve essere eseguito si trova un qualsiasi pezzo.

Per tutti questi motivi l'arrocco è una mossa che ogni giocatore può eseguire una sola volta in tutta la partita.

Il Re si dice essere 'in scacco' quando è attaccato da uno o più pezzi avversari, anche qualora questi pezzi fossero impossibilitati a muoversi dalla casa che occupano in quanto così facendo lascerebbero o porrebbero il proprio Re sotto scacco. Non si può muovere alcun pezzo che esponga ad uno scacco il Re del proprio colore, o che lasci il Re stesso sotto scacco.

Mosse legali e illegali

Una mossa è legale quando sono state ottemperate tutte le prescrizioni prima dette.

Una mossa è illegale quando non ottempera tutte le prescrizioni prima dette (per esempio mettendo il proprio Re sotto scacco, muovendo un alfiere in orizzontale, ecc.)

Naturalmente, non è possibile eseguire mosse illegali.

L’ESECUZIONE DELLA MOSSA

Se il giocatore che ha la mossa tocca sulla scacchiera, con l’intenzione di muovere o catturare:
a. uno o più dei propri pezzi, deve muovere il primo pezzo toccato che possa essere mosso
b. uno o più pezzi avversari, deve catturare il primo pezzo toccato che possa essere catturato
c. un pezzo di ciascun colore, deve catturare il pezzo avversario con il proprio pezzo oppure, qualora ciò sia illegale, muovere o catturare il primo pezzo toccato che possa essere mosso o catturato.

Se un giocatore che ha la mossa:
a. tocca il Re ed una Torre, deve arroccare da quel lato, sempre che ciò sia legale
b. tocca deliberatamente una Torre e quindi il proprio Re, non gli è consentito arroccare da quel lato in quella mossa, e la situazione sarà regolata da quanto detto più sopra.
c. volendo arroccare, tocca il Re e quindi una Torre, ma l’arrocco da quel lato è illegale, il giocatore deve fare un’altra mossa legale con il proprio Re (il che può includere l’arrocco con l’altra Torre). Se il Re non ha mosse legali, il giocatore è libero di eseguire una qualsiasi mossa legale.

Se nessuno dei pezzi toccati in base a quanto detto prima può essere mosso o catturato, il giocatore può fare una qualsiasi mossa legale.

LA CONCLUSIONE DELLA PARTITA

La partita è vinta da parte del giocatore che ha dato scaccomatto al Re avversario.

La partita è vinta dal giocatore il cui avversario dichiara di abbandonare.

La partita è patta quando il giocatore che deve muovere non ha alcuna mossa legale ed il suo Re non è sotto scacco. Si dice che la partita finisce per 'stallo'.

~~La partita è patta quando si raggiunge una posizione in cui nessuno dei due giocatori può dare scaccomatto all'avversario con una qualsiasi sequenza di mosse legali. Si dice allora che la partita finisce in 'posizione morta'.~~

~~La partita è patta per accordo tra i due giocatori.~~

Il testo qui sotto evidenziato si riferisce alle regole da considerare solo per il bonus da 4 punti in consegna singola

~~La partita può finire patta se un'identica posizione sta per apparire od è apparsa sulla scacchiera almeno tre volte (regola detta della 'triplice ripetizione di posizione').~~

~~La partita può finire patta se ciascun giocatore ha eseguito almeno le ultime 50 mosse senza alcuna mossa di pedone e senza alcuna cattura (regola detta della 'patta per 50 mosse').~~

Nota:
Per gli scopi di questo documento basti conoscere lo scaccomatto e lo stallo. Le altre tipologie per terminare la partita riguardano soprattutto gli scacchi agonistici e i tornei, e in alcuni casi non sono così semplici; si consiglia pertanto di leggere il regolamento completo per avere informazioni dettagliate a questo riguardo.