Dokumentation

Inhaltsverzeichnis

1 Hinweise zu AGB - Benutzung Verboten

Laut den Allgemeine Geschäftsbedingungen der Spielwerk GmbH[1] ist die **Benutzung** dieses Programms **verboten**. Benutzt man es trotzdem, kann man **vom Spiel ausgeschlossen** werden.

Es gibt mittlerweile dazu auch ein Statement[2] von einem Administrator:

> Hallo,
>
> ich wünsche den Tüftlern der Bots weiterhin viel Spaß und an die Nutzer nur folgendes: Bitte wundert euch nicht, wenn eure Spielaccounts über kurz oder lang Lifetime gesperrt werden. Ihr verstoßt mit der Nutzung gegen AGB und verursacht Zusatzarbeitszeit, die wir eigentlich lieber in die Weiterentwicklung des Spieles investieren würden. Wie dem auch sein Botusing ist kein Kavaliersdelikt, kann sehr wohl von uns nachvollzogen werden und wird definitiv mit einer Komplettsperrung auf Lebenszeit geahndet.
>
> MfG,
>
> Daybreaker
>
> Spieladministrator Bundeskampf

2 Runterladen benötigter Dateien

Auf der Webseite[3] kann man verschiedene Dateien runterladen. Java[4] wird für die Ausführung benötigt. Für einen einfachen Betrieb braucht man dann nur zwei Dateien:

- BKampfBot.jar
- config.json

Die `.jar`-Datei heißt auf der Webseite, wie die aktuelle Version. Zur Zeit der Dokumentation lautet der Name `"1.2.7.jar"`. Man kann diesen Namen umbenennen. Beide Dateien müssen im gleichen Verzeichnis liegen.

[1] http://spielwerk.eu/index.php?option=com_content&view=article&id=90&yt_color=mint&Itemid=157
[2] http://www.georf.de/bundeskampf-bot-version-1/#comment-305
[3] http://bundeskampf.georf.de
[4] http://www.java.com/de/download/

Über das Trac[5] kann man sich den Quellcode angucken und runterladen. Diesen braucht man nur, falls man das Programm verändern möchte. Dazu sollte man beachten, dass der Code unter der Lizenz GPLv2[6] steht.

3 Schnellstart Windows

Da schon viele Windowsnutzer beim ersten Starten scheitern, habe ich eine Zip-Datei mit allen nötigen Dateien erstellt (`schnellstartWindows.zip`). Dort ist auch eine Stapelverarbeitungsdatei (`start.bat`) enthalten. Nach dem Entpacken einfach einmal diese Datei starten. Es sollte sich ein Fenster öffnen, welches erst nach Bestätigung wieder schließt. Nach dem ersten Starten sollte die Ausgabe ungefähr so aussehen:

```
00:41 Login (meinName)
Get an error at initiation
Reason 1: Login failed
Reason 2: Something on server side changed.

If you want to report a bug, please post this:

Expected a ':' after a key at character 24
json.JSONTokener.syntaxError(JSONTokener.java:419)
json.JSONObject.<init>(JSONObject.java:212)
[...]
```

Der Login kann natürlich nicht stattfinden. Es fehlen die Logindaten in der Konfiguration. Außerdem ist der Hostname des Spiels nicht gesetzt. Diese Datei (`config.json`) ist zu öffnen zu die entsprechenden Einträge zu bearbeiten. Bei Hostname könnte man `http://www.bundeskampf.com/` eintragen, wenn man sich dessen bewusst ist, was man dort tut.

Sollte nun ein Login möglich sein, erklärt der Rest der Dokumentation, wie man den Bot einstellt. Sollte es nicht funktionieren, bitte auch **zuerst** dieses PDF lesen und dann beschweren.

4 Allgemeine Hinweise zum Programm

Das Programm arbeitet über die Kommandozeile. Öffnen Sie also zuerst eine Konsole und navigieren Sie in das Verzeichnis, wohin Sie die Datei geladen haben. [7][8]

Der allgemeine Aufruf dann sieht wie folgt aus:

```
eingabe# java -jar BKampfBot.jar
```

Ruft man das Programm nur mit der Hilfe auf, erschein folgendes:

[5] https://warnow.mgvmedia.com/trac/BKampfBot/
[6] http://www.gnu.org/licenses/gpl.html
[7] Windows: http://www.carpelibrum.de/tutorials/windows_konsole.pdf
[8] Linux: http://wiki.ubuntuusers.de/Terminal

```
eingabe# java -jar BKampfBot.jar --help
 ____    _  __                        __
|  _  \ | |/ /                       / _|
| | | | | ' / __ _ _ __ ___  _ __ | |_
| | | | |  < / _` | '_ ` _ \| '_ \|  _|
| |_| | | . \ (_| | | | | | | |_) | |
|     / |_|\_\__,_|_| |_| |_| .__/|_|
|   -<   ____          _     | |
|  _  \ |  _ \      | |     |_|
| | | | | |_) | ___ | |_
| | | | |  _ < / _ \| __|
| |_| | | |_) | (_) | |_
|_____/ |____/ \___/ \__|


BKampfBot - Version: 1.1.alpha4

Aufruf: BKampfBot.jar [Optionen]
        BKampfBot.jar [Optionen] Tag-Modi [Tag-Modi]
        BKampfBot.jar [Optionen] lotto
        BKampfBot.jar [Optionen] pins=PINARRAY
                BKampfBot.jar [Optionen] testproxy

Optionen:
  --help  -h                    Zeigt diese Hilfe an
  --config=FILE                 Nimmt FILE als Konfiguration
  --output=INT                  Setzt INT als Output-Level
  --fightAgain=INT              Setzt INT als D-Mark für erneuten Angriff

Tag-Modi:
  quiz            Tagesquiz
  los             Rubbellos
  spiel           Tagesspiel
  wein            Weinkeller
  glueck          Glücksrad

Weitere Hinweise auf http://bundeskampf.georf.de
```

5 Ausgaben

Das Programm erzeugt Konsolenausgaben. Dabei schreibt es in `stderr` (Fehlerausgabe) und `stdout` (Normale Ausgabe). Es gibt nun Möglichkeiten diese Ausgaben getrennt, bzw. zusammen in eine Datei umzuleiten. Dadurch kann man mehrere Stunden zurückblicken. Die Konsole speichert nur eine bestimmte Anzahl von Zeilen, die man zurück scrollen kann. Die Ausgabeumleitung ist in Windows[9] und Linux[10] möglich.

6 Konfigurationsdatei

Normalerweise liegt die Datei `config.json` im gleichen Verzeichnis, wie das Programm. Es müssen bestimmte Angaben gemacht werden, ohne die das Programm beim Start einen Fehler ausgibt. Dazu gehören: "Benutzername", "Passwort" und "Host". Name und Passwort werden in ‘"’ eingeschlossen und bezeichnen die Zugangsdaten. Der Host ist das Ziel jeder Anfrage. Man könnte hier zum Beispiel `http://www.bundeskampf.com/` setzen, wenn man sich des AGB-Verstoßes bewusst ist.

"Ausgabe" bezeichnet die Menge der Ausgabe. 0 → nur Fehler; 1 → Hinweise auf aktuelle Funktionen; 2 → alle Ausgaben. Diesen Wert kann man auch mittels

[9] http://support.microsoft.com/kb/110930/de
[10] http://www.fibel.org/linux/lfo-0.6.0/node111.html

Option –output gesetzt werden. Die beiden weiteren Angaben werden als Listen bezeichnet und beinhalten die Funktionen, die im Abschnitt 7 erklärt werden.

Alle Zeilen, die mit der Raute (#) anfangen, werden nicht benutzt. Sie können also als Kommentare oder Ausklammeroperator genutzt werden.

Beispiele für Konfigurationsdateien finden Sie im Abschnitt 10.

Hier folgt eine minimale Konfigurationsdatei.

```
"Benutzername":"name",
"Passwort":"geheim"
```

In der folgenden Tabelle sind alle Konfigurationsparameter aufgelistet.

Name	**Beschreibung**	**Datentyp**
"Benutzername"	Pflichtangabe	`String`
"Passwort"	Pflichtangabe	`String`
"Ausgabe"	Wert für die Menge der Ausgabe (0,1 oder 2)	`Int`
"Plan0"	Plan, wenn der Außendienst verfügbar ist	`Array`
"Plan1"	Plan, wenn der Außendienst nicht verfügbar ist	`Array`
"Angriff nochmal"	Wenn der Gewinn beim Angriff überschritten wird, wird der Gegner nochmal angegriffen	`Int`
"Info Pfad"	Siehe Erweitere Ausgabe und Befehle	`String`
"Mehr Aussendienste"	Angabe, wie viele Außendienste zusätzlich mit Zwergeneinsatz gemacht werden sollen	`Int`
"Zeitfaktor"	Siehe Zeitfaktor	`Double`
"Useragent"	Siehe Useragent	`String`
"Vorausschauen"	Sollen die Pläne vorausschauend arbeiten (Siehe Vorausschauende Pläne)	`Bool`
"Proxy"	Siehe Proxy	`Object`

7 Programm-Ablaufplan

Die eigentliche Aufgabe des Programms beinhaltet das Abarbeiten eines Plans nach bestimmten Kriterien. Dieser Plan wird in der Konfigurationsdatei (`config.json`) definiert. Es gibt zwei unterschiedliche Pläne, in denen einzelne Funktionen aufgerufen werden.

Der Plan 0 (in der Version 1.0 als "Außendienst verfügbar" bezeichnet) wird ausgeführt, falls man am aktuellen Tag noch einen Außendienst machen darf. Nach jedem Aufruf einer Funktion wird erneut geprüft ob noch Außendienste gemacht werden dürfen. Ist das nicht der Fall, wird in den Plan 1 übergegangen.

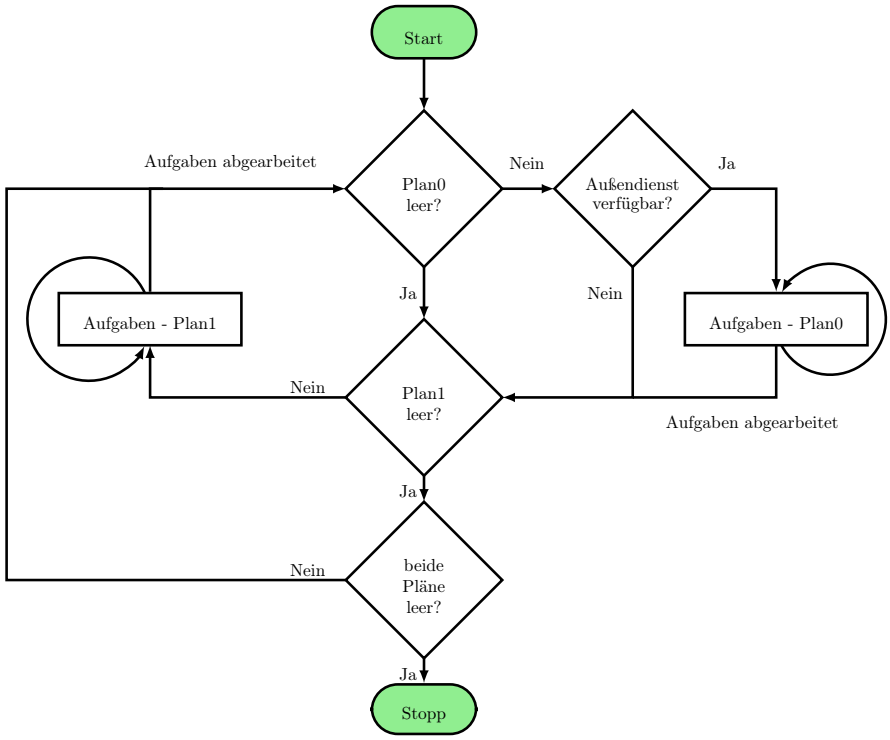
Der Plan 1 (in der Version 1.0 als "Kein Außendienst" bezeichnet) wird in zwei Fällen ausgeführt. Zum einen, wenn im Plan 0 keine Funktionen definiert sind, der Plan also leer ist. Zum anderen, wenn es für den aktuellen Plan keine Außendienste mehr ausgeführt werden können.

Die Pläne werden in der Config in einer Aufzählung beschrieben. Zwischen ihnen müssen also immer Kommata stehen. Hier ein Beispiel:

```
"Plan0":[
        {"Aussendienst":0},
        {"Aussendienst":1},
        {"Bank":0},
        {"Bank":103},
        {"Boese":0}
],
```

Wie man erkennen kann, werden die Funktionen in eckigen Klammern eingeschlossen. Nach der letzten Funktion kommt kein Kommata.

Es folgt ein Diagramm zur Erleuterung des Ablaufs der Pläne:



Nachfolgend werde alle implementierten Funktionen beschrieben. Sie können in beiden Plänen benutzt werden.

7.1 Angriff

Der Befehl "Angriff" bringt den Bot dazu sich einen geeigneten Gegner zu suchen und ihn anzugreifen. Dabei kann man zwei Parameter einstellen. "Stufe" bezeichnet die Anzahl der Level, die der Gegner Unterschied haben soll. Ist man selber im Level 8 und will jemanden aus Level 5 angreifen trägt man also -3 ein. Der zweite Parameter sagt was über die Vereinszugehörigkeit aus: -1 = Kein Verein; 0 = Egal; 1 = Im Verein.

```
"Plan0":[
# Angriff mit eigene Stufe+1 und im Verein
        {"Angriff":{"Stufe":1,"Verein":1}},
# Angriff mit eigene Stufe-5 und nicht im Verein
        {"Angriff":{"Stufe":-5,"Verein":-1}},
# Angriff mit eigene Stufe und egal ob im Verein
        {"Angriff":{"Stufe":0,"Verein":0}}
],
```

Ab Version *1.0.beta8* gibt es zusätzliche Optionen, die aber auch weggelassen oder kombiniert werden können:

```
"Plan0":[
# Kein Angriff gegen Freunde
        {"Angriff":{"Stufe":0,"Verein":0,"Freunde":true}},
# Angriff nur gegen gegen bestimmten 10000 Respekt
        {"Angriff":{
                "Stufe":0,
                "Verein":0,
                "Respekt":
                        {"Min":1000,"Max":10000}
                }
        }
],
```

Die Respekteinteilung ist ab *1.1.beta8* neu. Wenn man dort -1 wählt, wird diese Abgrenzung nicht beachtet, man kann also somit auch nur in eine Richtung begrenzen.

Freunde sind Personen auf der Freundesliste (*Zu mir* ⇒ *Freundes Liste*)
Generell werden nur Feinde aus anderen Bundesländern angegriffen.

Es kann auch die Medizin-Option gesetzt werden.

Es kann auch die Zwerge geben-Option gesetzt werden.

```
"Plan0":[
        {"Angriff":{"Stufe":0,"Verein":0,"Medizin":40,"Zwerg":true}}
],
```

7.2 Arbeiten

Der Befehl "Arbeiten" nimmt eine Zahl als Parameter. Diese stellt die Anzahl der Stunden ein.

```
"Plan0":[
        {"Arbeiten":1}
],
```

Mit der zusätzlichen Option "Stopp" wird der Bot nach dem Beginn der Arbeit beendet, um Systemressoucen zu sparen.

```
"Plan0":[
        {"Arbeiten":{"Stunden":10,"Stopp":true}}
],
```

7.3 Außendienst

Der Befehl "Aussendienst" ist nur in Plan 0 sinnvoll, kann also auch nur dort eingesetzt werden. Als Parameter kann 0 für leichten und 1 für schweren Außendienst gesetzt werden. (**Achtung**: In Version 1.0 hieß diese Funktion "Außendienst")

```
"Plan0":[
        {"Aussendienst":0},
        {"Aussendienst":1}
],
```

Es kann auch die "Medizin"-Option gesetzt werden. Dann lautet die die Einstufung in Schwierigkeiten `"Stufe"`. Dies macht natürlich nur beim schweren Außendienst Sinn.

```
"Plan0":[
        {"Aussendienst":{"Stufe":1, "Medizin":50}}
],
```

7.4 Beschreibung

Diese Funktion ermöglicht es, den Bot zu steuern ohne ihn auszuschalten oder am richtigen Rechner zu sitzen. Man trägt die Funktion einfach in den Plan ein.

```
"Plan0":[
        {"Beschreibung":true}
],
```

Immer wenn der Plan zu der Funktion "Beschreibung" kommt, sucht das Programm auf der eigenen Profilseite in der Beschreibung nach einem Befehl. Dieser

muss folgende Form haben:

```
{"Befehl":[
]}
```

Zwischen den eckigen Klammern können beliebige Befehle stehen, wie in den anderen Plänen auch. Wichtig, dass das erste Zeichen die geschweifte Klammer ist. Der Bot schreibt dann Hinweise in diese Beschreibung, wenn er fertig ist oder die Eingabe nicht richtig verstehen konnte.

Seit der Version `1.0.beta5` gibt es eine Erweiterung hierzu. Der Platz in der Beschreibung ist begrenzt, also kann man Angriffe in einer einfachen Liste ausführen. Der folgende Befehl so direkt in die Beschreibung eingegeben, würde zwei Angriffe auf "_wolf_" ausführen, da die ID "CHEGGF", die von "_wolf_" ist.

```
["CHEGGF","_wolf_"]
```

Ab Version `1.1.alpha3` ist es zusätzlich möglich, eine persönlich angepasste Beschreibung in den Plan anzugeben. Diese wird nach Abarbeitung der Befehle in die Beschreibung geschrieben. Wird diese nicht angegeben, schreibt das Programm immer *Ich war bisher zu faul eine Beschreibung einzugeben, hole ich aber sicher bald nach!* in die Beschreibung.

```
"Plan0":[
        {"Beschreibung":"Ich␣bin␣fertig␣mit␣den␣Aufgaben␣:)"}
],
```

7.5 Bank

Der Befehl "Bank" bringt Geld auf die Sparkasse. Mit dem Parameter 0 bringt man so viel wie möglich auf die Bank. Mit allen anderen positiven Zahlen wird genauso viel Geld eingezahlt, welches von der Zahl repräsentiert wird.

```
"Plan0":[
# Bringt alles auf die Bank
        {"Bank":0},
# Bringt nur 205 DM auf die Bank
        {"Bank":205}
],
```

7.6 Befehl

Der Befehl "Befehl" prüft ob ein Befehl vorliegt. Wird einer oder mehrere gefunden, wird immer der erste Abgearbeitet und danach wieder die Datei geöffnet um den nächsten zu suchen. Erst wenn alle Befehle abgearbeitet sind, wird die normale Ausführung fortgesetzt. Weitere Hinweise unter Erweitere Ausgabe und Befehle.

```
"Plan0":[
# Prüft ob ein neuer Befehl vorliegt
        {"Befehl":true}
],
```

7.7 Booster

Mit der Vereinsvitrine kam auch das Spiel "Booster" hinzu. Es kann mit folgenden Befehl ausgeführt werden. Funktioniert natürlich nur, wenn man am aktuellen Tag noch Booster ausführen darf.

```
"Plan0":[
# Booster mit Ergebnis 5 ausführen
        {"Booster":5}
],
```

7.8 Böse Spieler

Der Befehl "Boese" greift eine Person aus der "Böse Spieler"-Liste an. Alle Personen in der Liste werden gleichmäßig angegriffen. Pro Befehl wird ein Angriff ausgeführt. (**Achtung**: In Version 1.0 hieß diese Funktion "Böse", bis Version 1.2.3 hieß die Funktion dann "Boese")

Seitdem die Böseliste in drei Teile (Beute, Krieg, Respekt) aufgeteilt wurde, gibt es die Möglichkeit auch nur eine Liste davon anzugreifen:

```
"Plan0":[
# Beute-Liste-Person angreifen
        {"BoeseBeute":{}},
# Krieg-Liste-Person angreifen
        {"BoeseKrieg":{}},
# Respekt-Liste-Person angreifen
        {"BoeseRespekt":{}}
],
```

Es kann auch die Medizin-Option gesetzt werden.

Es kann auch die Zwerge geben-Option gesetzt werden.

```
"Plan0":[
        {"BoeseRespekt":{"Medizin":40,"Zwerg":true}}
],
```

Seit der Version `1.2.5` gibt es die Möglichkeit einen Beutereichen Gegner erneut anzugreifen. Dies geschieht ähnlich dem Prinzip beim normalen Angriff. Man gibt mit der Option `nochmal` einen D-Mark-Betrag an. Falls der bei einem Angriff erreicht wurde, wird bei dem nächsten Aufruf des gleichen Planelements genau dieser Gegner nochmal angegriffen.

```
"Plan0":[
# nochmal angreifen ab 40 D-Mark Beute
        {"BoeseRespekt":{"nochmal":40}}
],
```

7.9 Goldene Zwerge

Der Befehl "Golden" greift den nächsten Goldenen Zwerg an. Dabei ist diese Option nur Sinnvoll, wenn die Konfiguration mit der erweiterten Ausgabe eingestellt ist. Außerdem sollte man es nur in der Beschreibung benutzen.

```
{"Befehl":[
        {"Golden":true}
]}
```

Es kann auch die Medizin-Option gesetzt werden.

```
"Plan0":[
        {"Golden":{"Medizin":40}}
],
```

7.10 Kampf

Der Befehl "Kampf" erzeugt im Gegensatz zu "Boese" und "Angriff" eine Keilerei mit einem bestimmten Gegner. Als Parameter wird entweder die ID oder

der Name des Gegners übergeben. Die ID ist das Ende der URL, wenn man auf dem Profil ist. Bei der URL des Profils von "_wolf_" (1. Platz) lautet:
`http://www.bundeskampf.com/characters/profile/CHEGGF`
Die ID lautet also: `CHEGGF` Man könnte auch den Namen angeben, allerdings wird der bei zu kurzen Namen nicht immer gefunden. Hier beide Beispiele:

```
"Plan0":[
        {"Kampf":"CHEGGF"},
        {"Kampf":"_wolf_"}
],
```

Es kann auch die Medizin-Option gesetzt werden. Das feindliche Profil wird mit `"Gegner"` angegeben.

Es kann auch die Zwerge geben-Option gesetzt werden.

```
"Plan0":[
        {"Kampf":{"Gegner":"_wolf_","Medizin":40,"Zwerg":true}}
],
```

7.11 Minuten

Der Befehl "Minuten" bewirkt ein Schlafen des Bots. Als Parameter nimmt er die Anzahl der Minuten, die er Schlafen soll. Dieser Befehl muss nicht für die anderen Funktionen benutzt werden, sondern ist dafür da, dass man selber etwas machen kann.

```
"Plan0":[
# Warte 3 Minuten
        {"Minuten":3},
# Warte 2 Stunden
        {"Minuten":120}
],
```

7.12 Neustart

Will man an einer Stelle den internen Speicher säubern oder ist durch irgendeinen Bug genervt, kann man den Bot kontrolliert neu starten.

```
"Plan0":[
        {"Neustart":true}
],
```

7.13 Skill

Dieses Element ermöglicht es das erworbene Geld direkt in einer Kategorie zu skillen. Dafür wird es so benutzt:

```
"Plan1":[
        {"Skill":{"Kategorie":"Mukkies"}},
        {"Skill":{"Kategorie":"Schleuderkraft"}},
        {"Skill":{"Kategorie":"Fitness"}},
        {"Skill":{"Kategorie":"Wahrnehmung"}},
        {"Skill":{"Kategorie":"Glueck"}}
],
```

Bei jeden Aufruf wird so oft wie möglich geskillt. Will man das nicht, muss man die maximale Anzahl angeben:

```
"Plan1":[
        {"Skill":{"Kategorie":"Mukkies", "Anzahl":1}},
        {"Skill":{"Kategorie":"Schleuderkraft", "Anzahl":2}},
        {"Skill":{"Kategorie":"Fitness", "Anzahl":3}},
        {"Skill":{"Kategorie":"Wahrnehmung", "Anzahl":4}},
        {"Skill":{"Kategorie":"Glueck", "Anzahl":5}}
],
```

7.14 Stopp

Normalerweise arbeitet der Bot die Listen nacheinander ab und fängt wieder von vorne an. Um dies zu Unterbrechen gibt es den Befehl "Stopp".

```
"Plan0":[
        {"Stopp":true}
],
```

8 Programm-Varianten

Das Programm implementiert verschiedene Hilfsmittel. Hier werden Sie aufgeführt und beschrieben.

8.1 Tägliche Möglichkeiten

Folgende tägliche Möglichkeiten können kombiniert werden. Maximal zu:

```
eingabe# java -jar BKampfBot.jar los quiz glueck wein spiel
```

Sie werden in der jeweiligen Reihenfolge mit nur einem Login ausgeführt.

8.1.1 Tagesquiz

Mit Hilfe des Programms kann das Tagesquiz ziemlich gut gelöst werden. Und es geht viel schneller, als wenn man es selber macht. Der Aufruf sieht wie folgt

aus:

```
eingabe# java -jar BKampfBot.jar quiz
```

Die Fragen werden direkt (richtig) beantwortet. Sollen **nicht** alle Fragen richtig beantwortet werden[11], kann man auch angeben, wie viele Fragen **falsch** sein sollen:

```
eingabe# java -jar BKampfBot.jar quiz=2
```

8.1.2 Rubbelos

Wer schon mal selber versucht hat, die Rubbellose von ihrer Beschichtung zu befreien, der lässt es sehr schnell sein, denn es dauert zu lange. Will man es schneller erledigen, nimmt man folgenden Befehl:

```
eingabe# java -jar BKampfBot.jar los
```

8.1.3 Glücksrad

Diese Option funktioniert nur in einem Verein. Sie dreht das Rad auf die Gewinnposition.

```
eingabe# java -jar BKampfBot.jar glueck
```

8.1.4 Weinkeller

Diese Option funktioniert nur in einem Verein. Sie spielt das Weinkellerspiel und gewinnt es.

```
eingabe# java -jar BKampfBot.jar wein
```

8.1.5 Tagesspiel

Das Tagesspiel wird mit voller Punktzahl ausgeführt.

```
eingabe# java -jar BKampfBot.jar spiel
```

[11] https://warnow.mgvmedia.com/trac/BKampfBot/ticket/36

8.2 Lottoschein

Genau wie im Fernsehen könnt ihr es euch Sonntags Abend gemütlich machen und darauf hoffen, dass die Glücksfee euch diese Woche hold ist. Und ihr - zumindest in Bundeskampf - nicht mehr zu arbeiten braucht.

Das Ausfüllen eines Scheins dauert auch ein wenig. Und welche Zahlen soll man nehmen? Mit dem folgenden Aufruf erledigt es das Programm:

```
eingabe# java -jar BKampfBot.jar lotto
```

Beispielausgabe (die Zahlen werden zufällig gewählt):

```
Login
visit: http://www.bundeskampf.com/lotto/history
Added numbers: 2,5,10,38,40
Logout
visit: http://www.bundeskampf.com/signups/logout
```

Ab Version `1.1.alpha6` kann man die Nummern auch direkt angeben:

```
eingabe# java -jar BKampfBot.jar lotto=12,34,23,45,2
```

8.3 Testen des Proxy

Mit der Option `testproxy` wird eine Webseite aufgerufen, die nur die IP zurückgibt. Diese kann man dann mit der originalen IP vergleichen.

```
eingabe# java -jar BKampfBot.jar testproxy
Deine IP: 1.2.3.4
```

9 Weitere Hinweise

9.1 Medizin

Man kann bei verschiedenen Optionen die Wartezeit verkürzen, indem man Medizin kauft. Im Spiel wird diese "Direkt Fit" genannt. Folgende Pläne benutzen die Aktion: Angriff, Außendienst, Böse Spieler, ~~Goldene Zwerge~~ und Kampf.

In der Konfiguration gibt man eine ganzstellige Zahl an, die als Prozent der restlichen Lebensenergie angesehen wird, ab der Medizin gekauft wird. Die Rechnung lautet:

$$\frac{Lebenspunkte}{MaximaleLebenspunkte} = Prozent$$

Am besten ein Beispiel zur Anschauung. Jemand hat folgenden Plan eingetragen:

```
"Plan0":[
        {"Angriff":{"Stufe":0,"Verein":0,"Medizin":40}}
],
```

Es soll also Medizin gekauft werden, wenn nach dem Kampf die Lebensenergie unter 40% gefallen ist. Nun kämpft er und erhält folgendes Ergebnis:

$$\frac{345LP}{1000LP} = 35\%$$

Die Medizin wird also gekauft.

Trägt man 0 ein, wird gekauft, wenn man den Kampf verloren hat und die Lebenspunkte komplett runter sind. Wird 100 eingetragen, wird die Medizin immer gekauft.

9.2 Zwerge geben

Man kann bei den Kämpfen die Wartezeit auflösen, indem man einen Zwerg einsetzt. Deswegen gibt es ab Version `1.2.7` die Möglichkeit dies anzugeben. Dafür gibt man einfach die Option mit an:

```
"Plan0":[
        {"Angriff":{"Stufe":0,"Verein":0,"Medizin":40,"Zwerg":true}}
],
```

Wenn nun zu dem Zeitpunkt ein Zwerg eingelöst werden kann, wird die Wartezeit verkürzt und keine Medizin gekauft. Sollten keine Zwerge mehr da sein, wird in diesem Fall die Medizin gekauft.

9.3 Vorausschauende Pläne

Ab Version `1.1.alpha6` ermöglichen ein neuer Planmanager das vorausschauen von Aktionen. Innerhalb der Kampfwartezeit kann somit zum Beispiel schon ein Außendienst gestartet werden, falls er im Plan an nächster Stelle folgt.

Folgende Pläne werden früher gestartet:

- Arbeiten
- Außendienst
- Bank
- Booster

Folgende Pläne ermöglichen ein vorzeitiges Abarbeiten in der Wartezeit:

- Angriff
- Böse Spieler
- Goldene Zwerge
- Kampf

Um diese Funktionalität zu deaktivieren benötigt man folgenden Befehl in der Konfiguration:

```
"Benutzername":"meinName",
"Passwort":"geheim",
"Vorausschauen":false,
[...]
```

9.4 Useragent

Ein Programm sendet im Internet meistens einen Hinweis auf die eigene Version an die Webserver. Browser machen das auch. Deshalb sendet der Bot auch einen Useragent, der genauso aussieht, wie der eines Firefoxes oder Internet-Explorers. Mit der Option `Useragent` kann man seinen eigenen Useragent setzen um einen anderen Browser vorzutäuschen.

```
[...]
"Useragent":"Mozilla/5.0␣(Windows;␣U;␣Windows␣NT␣5.1;␣de;␣rv:1.9.0.10)␣Gecko/2009042316␣Firefox/3.0.10",
[...]
```

9.5 Zeitfaktor

Mittels der Option `Zeitfaktor` kann man die zeitliche Abarbeitung des Bots beeinflussen. Im Programm sind an einigen wiederkehrenden Stellen Halteposition eingebaut. An diesesen Stellen ‘schläft‘ das Programm für eine bestimmte Zeit, bevor es die Abarbeitung fortsetzt. Mit dem Zeitfaktor kann man diese Zeit um den angegeben Faktor verlängern.

Der Faktor wird an die vorgegebene Zeit angepasst. Schläft der Bot länger als eine Minute, wird der Faktor ignoriert.

```
Hole http://example.com/1.html
Schlafe (10 Millisekunden * Faktor)
Hole http://example.com/2.html
Schlafe (10 Millisekunden * Faktor)
```

Den Faktor kann man nun auf zwei Arten einstellen, entweder mit einem festen Wert oder mit einem Intervall, indem bei jedem Anwenden zufällig ein Wert daraus genommen wird.

```
[...]
# fest
"Zeitfaktor":20,

# Intervall
"Zeitfaktor":{"min":20, "max": 100},
[...]
```

Durch diese Option kann der Bot sehr individuell gestaltet werden. Ein höherer Wert lässt den Bot langsamer arbeiten, wodurch aber ein geringeres Entdeckungsrisiko gegeben ist.

9.6 Proxy

Ab der Version 1.2.3 ist es möglich einen Proxy zu benutzen. Dafür ist zwingend die Angabe von Hostname und Port erforderlich. Falls eine Anmeldung bei dem Proxy erforderlich ist, kann der Benutzername und das Passwort angegeben werden.

```
[...]
# Proxy ohne Authentifizierung
"Proxy":{
        "Host":"IP␣oder␣Domain",
        "Port":8080
},
[...]
```

```
[...]
# Proxy mit Authentifizierung
"Proxy":{
        "Host":"IP␣oder␣Domain",
        "Port":8080,
        "Username": "username",
        "Password": "password"
},
[...]
```

Um das ganze zu Testen gibt es die Option Testen des Proxy.

9.7 Erweitere Ausgabe und Befehle

Durch die Angabe der Config-Option `"Info Pfad"`, kann man ein Verzeichnis bestimmen, in dem eine erweiterte Ausgabe abgespeichert wird. Für jeden Kampf wird dort eine HTML-Datei mit Einzelheiten zum Verlauf gespeichert.

Legt man ein in diesem angegebenen Verzeichnis ein weiteres Verzeichnis mit dem Namen `befehl` an, hat man mit dem Plan Befehl weitere Möglichkeiten. Sobald der Bot zum diesem Planelement kommt, wird die Datei `befehl.json` abgearbeitet. In dieser Datei ist ein JSON-Objekt mit weiteren Planelementen

enthalten. Sobald ein Element abgearbeitet ist, wird es aus der Datei gelöscht. Der Inhalt von `befehl.json` sieht zum Beispiel so aus:

```
{
        "Befehl":[
                {"Kampf":"CHEGGF"},
                {"Kampf":"CHEGGF"},
                {"Kampf":"CHEGGF"}
        ]
}
```

9.8 Unerwartete Fehler

Sollte ein unerwarteter Fehler auftreten, welches zu einem Absturz des Systems führt, wird der Fehler mit mehreren Hinweisen ausgegeben. Danach startet sich das Programm neu. Das Programm versucht dann erneut seine Aufgaben zu erledigen. Passiert zu oft ein Absturz beendet sich das Programm.

Sollten Sie sich den Fehler durch die Ausgaben des Bots und durch selbstständigen Prüfen auf der Webseite nicht erklären lassen, wird darum gebeten ein Ticket auf der Entwicklerseite[12] zu erstellen. Dort können auch Wünsche für neue Funktionen geäußert werden.

9.8.1 Bekannte Fehler

Tritt ein Fehler auf, in dem von einem *Internal Server Error - 500* die Rede ist, liegt es an dem Bundeskampf-Server und nicht am Bot. Die genaue Fehlermeldung ist auf der Webseite zu finden.

10 Beispielkonfigurationen

Die folgenden Konfigurationen sind auch auf der Webseite zum Download verfügbar.

Eine Konfiguration für jemanden, der nur die Außendienste (schwer) machen will und danach das Programm beenden will, sieht dann so aus:

```
"Benutzername":"meinName",
"Passwort":"geheim",
"Plan0":[
        {"Aussendienst":1}
],
"Plan1":[
        {"Stopp":true}
]
```

[12] https://warnow.mgvmedia.com/trac/BKampfBot/newticket

Konfiguration: Nur Außendienste leicht, wenn fertig, regelmäßig prüfen, ob es schon ein neuer Tag ist und dann wieder Außendienste machen:

```
"Benutzername":"meinName",
"Passwort":"geheim",
"Plan0":[
        {"Aussendienst":0}
],
"Plan1":[
        {"Minuten":30}
]
```

Konfiguration: Keine Außendienste; Nur Angreifen, gleiche Stufe, im Verein.

```
"Benutzername":"meinName",
"Passwort":"geheim",
"Plan1":[
        {"Angriff":{"Stufe":0,"Verein":1}}
]
```

Konfiguration: Keine Außendienste; Nur Arbeiten, dann alles auf Bank bringen.

```
"Benutzername":"meinName",
"Passwort":"geheim",
"Plan1":[
        {"Arbeiten":1},
        {"Bank":0}
]
```

Konfiguration: Keine Außendienste; Angreifen, dann einen bösen Angreifen und dann eine Pause.

```
"Benutzername":"meinName",
"Passwort":"geheim",
"Plan1":[
        {"Angriff":{"Stufe":0,"Verein":1}},
        {"Boese":1},
        {"Minuten":10}
]
```