B) Kitten-Bot

Dein Exploding Kitten-Bot muss in der Lage sein, korrekt mit den Services zu kommunizieren. Ausserdem soll er seine Karten strategisch so ausspielen, dass er alle anderen Bots schlägt.

Aufbau

GitHub-Repository

Dein Bot besteht aus zwei Teilen:

Socket-Controller

Der Socket-Controller steuert die Kommunikation mit den Services. Er erzeugt auch ein Objekt deiner Bot-Klasse.

Beim Start registriert der Socket-Controller deinen Bot beim Clowder-Service. Als Response erhält er einen Port.

Request

- action: 'MEOW'
- ip: IP-Adresse des Bots
- name: Name des Bots
- type: 'bot'

Response

- Port: Eine zufällige Portnummer. Unter dieser Portnummer muss der Bot einen Socket öffnen und auf Nachrichten warten.

Danach öffnet er einen Socket mit diesem Port und hört dort auf die Nachrichten des Arena-Services.

Sobald ein Request eintrifft, wird dieser analysiert und die korrekte Methode in der Bot-Klasse aufgerufen.

Bot-Klasse

Die Bot-Klasse enthält die Logik zur Steuerung deines Bots. Im Vorlage-Repository findest du zwei Bots:

- TemplateBot: Eine Kopiervorlage für deine eigenen Bots
- RandomBot: Ein vollständiger Bot, der seine Entscheidungen anhand des Zufallsprinzips trifft.

Beide Beispiele enthalten eine Helfer-Klasse, welche die Methode `request(self, data)` implementiert. Diese Helfer-Klasse wird benötigt, um deine Bots mit der lokalen Version der Arena auszuführen. Sie ersetzen dabei den Socket-Controller.

Ablauf der Spielrunden

Immer wenn eine relevante Aktion im Spiel stattfindet, sendet der Arena-Service einen Request an deinen Bot. Er muss innerhalb einer Sekunde eine gültige Reponse an den Server senden. Andernfalls scheidet dein Bot aus.

Karten

Name	Beschreibung
NORMAL	Eine Karte ohne besondere Bedeutung.
DEFUSE	Diese Karte entschärft die Bombe. Sie wird automatisch gespielt, wenn du eine EXPLODE-Karte ziehst.
EXPLODING_KITTEN	Dein Bot explodiert, falls du keine DEFUSE-Karte spielst.
SEE_THE_FUTURE	Mit „See the future“ kannst du die nächsten 3 Karten auf dem Stapel sehen.
SHUFFLE	Mit dieser Karte kannst du den Kartenstapel neu mischen.

Spielstart

Dein Bot wird informiert, dass ein neues Spiel startet.

Request

- action: „START“
- bots: Anzahl der Bots in dieser Spielrunde
- card_counts: Array mit der Anzahl der verschiedenen Karten

Beispiel

```
{
  "card_counts": [
    {"name": "DEFUSE", "count": 2},
    {"name": "EXPLODING_KITTEN", "count": 1},
    {"name": "NORMAL", "count": 10},
    {"name": "SEE_THE_FUTURE", "count": 4},
    {"name": "SHUFFLE", "count": 3},
    {"name": "SKIP", "count": 8}
  ],
  "bots": [
    "cutekitty",
    "randombot"
  ],
```

```
  "action": "START"
}
```

Response

- ACK

Karte gezogen

Dein Bot hat eine Karte vom Stapel gezogen.

Request

- action: „DRAW“
- card: Name der gezogenen Karte

Beispiel

```
{„card“: „DEFUSE“, „action“: „DRAW“}
```

Response

- ACK

Dein Spielzug

Dein Bot ist am Zug und kann eine Karte ausspielen oder nicht. Dein Bot ist solange am Zug, bis er keine Karte mehr spielt. Er wird also mehrmals hintereinander mit dieser Aktion aufgerufen.

Die Ausnahme ist, wenn dein Bot die Karte „Skip“ ausspielt. Nach dieser Karte macht es keinen Sinn, dass er weitere Karten spielt.

Request

- action: „PLAY“
- bots: Anzahl der verbliebenen Bots in Spielrunde
- deck: Anzahl der Karten im Deck

Beispiel

```
{„action“: „PLAY“}
```

Response

- card: Name der auszuspielenden Karte **oder** „None“

Aktionen der anderen Bots

Ein Bot hat eine Aktion durchgeführt. Dein Bot wird über jede Aktion jedes Bots informiert, auch seine eigenen Aktionen.

Request

- action: „INFORM“
- event: „PLAY“ oder „DRAW“
- card: Name der Karte

Beispiele

Ein Bot hat …

- … eine Karte gespielt: `{„botname“: „cutekitty“, „event“: „PLAY“, „data“: „NORMAL“, „action“: „INFORM“}`
- … eine Karte gezogen: `{„botname“: „cutekitty“, „event“: „DRAW“, „data“: „null“, „action“: „INFORM“}`

Response

- ACK

Bombe entschärft

Dein Bot hat soeben eine „Exploding Kitten“ Karte gezogen und mit „Defuse“ entschärft. Du kannst nun die „Exploding Kitten“ Karte an einer beliebigen Stelle im Kartenstapel platzieren.

Request

- action: „PLACE“
- decksize: Anzahl der Karten im Stapel

Beispiel

```
{„decksize“: 3, „action“: „DEFUSE“}
```

Response

- position: Position innerhalb des Kartenstapels

Die Zukunft sehen

Dein Bot hat die Karte „See the future“ gespielt. Er sieht nun die nächsten 3 Karten im Stapel.

Request

- action: „FUTURE“
- cards: Liste mit den Namen der nächsten 3 Karten im Stapel.

Beispiel

```
{„cards“: [„EXPLODING_KITTEN“, „SEE_THE_FUTURE“, „NORMAL“], „action“: „FUTURE“}
```

Response

- ACK

Explosion

Dein Bot hat soeben eine „Exploding Kitten“ Karte gezogen und hatte keine „Defuse“ Karte. Damit ist dein Bot ausgeschieden.

Request

- action: „EXPLODE“

Beispiel

```
{„action“: „EXPLODE“}
```

Response

- ACK

Resultat einer Spielrunde

Am Ende einer Spielrunde erfährt dein Bot das Resultat.

Request

- action: „GAMEOVER“
- ranks: Die Rangliste der Bots vom ersten bis zum letzten Platz

Beispiel

```
{„ranks“: [„cutekitty“, „randombot“], „action“: „GAMEOVER“}
```

Response

- ACK