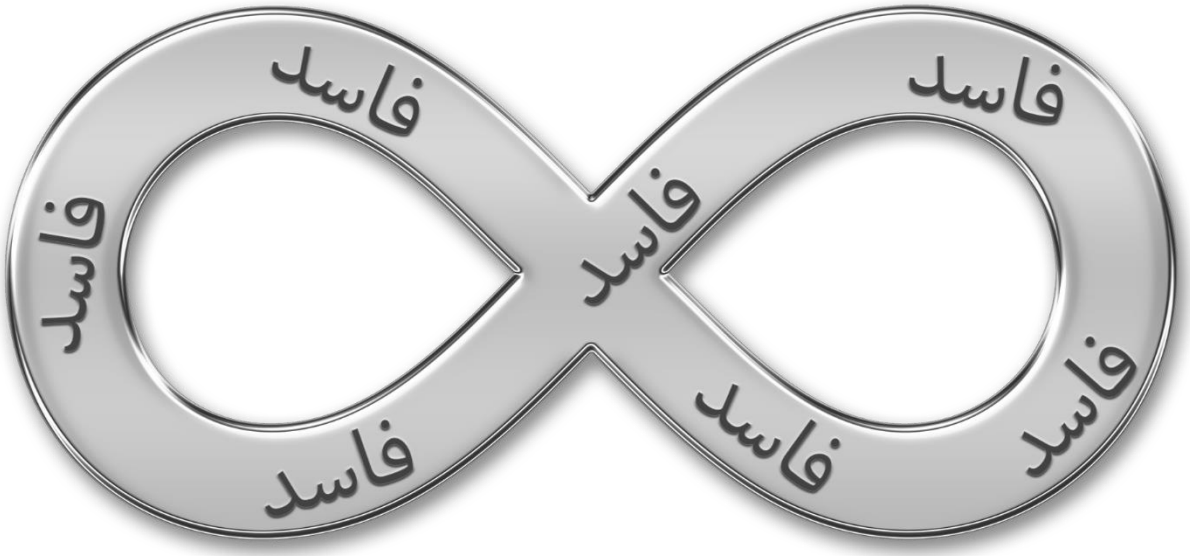
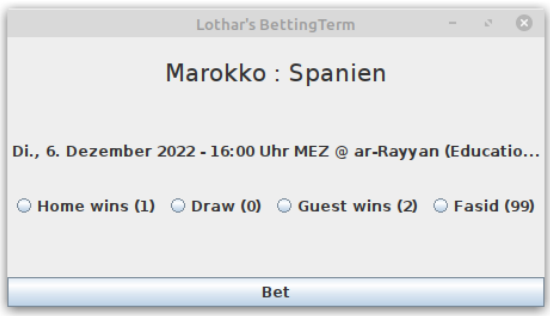
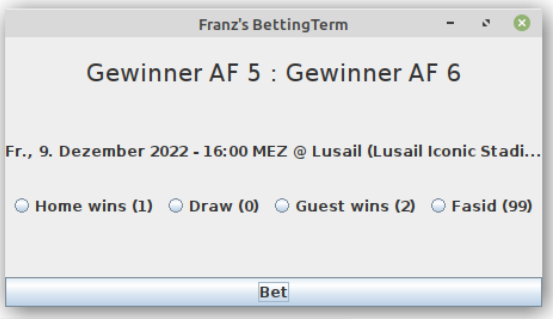
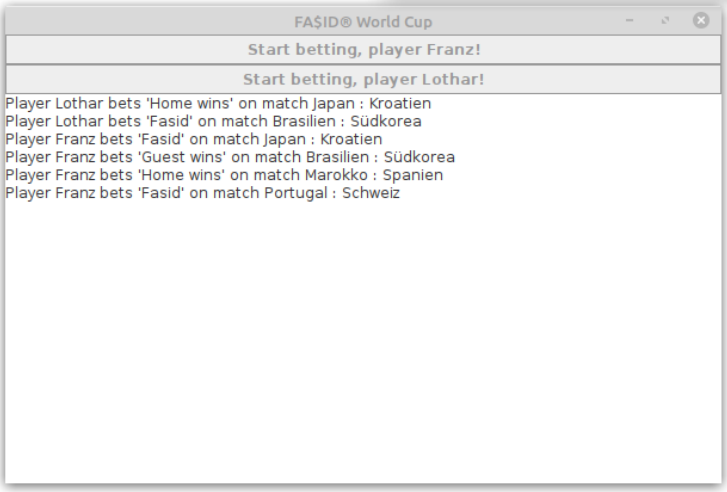


FA$ID® WORLD CUP







Hinweis zur Bewertung:

- 1/4 der Punkte (25%) wird nach Funktionstests Ihrer Lösung vergeben.
- 3/4 der Punkte (75%) werden entsprechend des in den Teilaufgaben angegebenen Schlüssels auf Basis des Quellcodes vergeben.

DHBW Karlsruhe, Vorlesung Programmieren I+II, Programmentwurf

09.01.2023

Bearbeitungszeit: 120 Minuten

Aufgabe

Nach der höchst transparenten und unstrittigen Entscheidung der FIFA, die Fußball-Weltmeisterschaft 2022 ins auch im Winter sonnige Katar zu vergeben, suchen Wettanbieter aktuell nach einer Lösung, mit der man beim Wetten auf Spielergebnisse - zusätzlich zu den bekannten Optionen {Sieg Mannschaft A, Unentschieden, Sieg Mannschaft B} - auch noch eine Einflussnahme vermeintlich unbeteiligter Organisationen (Namen werden hier ausdrücklich nicht genannt) bei der Wettannahme abbilden kann.
Schreiben Sie aus diesem Grund eine grafische Java-Anwendung **`Fasid`**, welche zusätzlich zu den genannten klassischen Funktionalitäten ebenfalls eine Option *Fasid* (arabisch, dt: „korrupt“) bereitstellt, mit der man bereits im Vorfeld darauf wetten kann, dass man die Integrität eines Spiels anzweifelt.

Teilaufgabe a) **[6%]**

Schreiben Sie zunächst einen *komplexen Aufzählungstypen* **`ResultType`**, mit dessen Hilfe die möglichen Ausgänge eines Spiels abgebildet werden. Neben der eigentlichen Konstante sind ein menschenlesbares Label (`label`) als Zeichenkette sowie ein Zahlenwert für die Auswahl (`toto`) vorzusehen.

Folgende Ergebnis-Werte sind zu implementieren:

ResultType	Label	Toto
`UNKNOWN`	`Unknown`	`-1`
`DRAW`	`Draw`	`0`
`HOME`	`Home`	`1`
`GUEST`	`Guest`	`2`
`FASID`	`Fasid`	`99`

Überschreiben Sie im komplexen Aufzählungstyp außerdem die Methode `toString` und geben Sie dort das Label und den Toto-Wert als Zeichenkette zurück (beliebig miteinander verkettet, bspw: „`Fasid (99)`“).

Teilaufgabe b) **[7%]**

Schreiben Sie eine Klasse **`Player`**, die einen Wettsteller repräsentiert. Ein Wettsteller besitzt lediglich einen Namen (`name`) als Attribut. Definieren Sie einen Konstruktor, der den Namen entgegennimmt!

Schreiben Sie außerdem eine Java-Schnittstelle mit dem Namen **`CorruptionPrevention`**, mit deren Hilfe unterschiedliche Clients realisiert werden können. Das Interface definiert folgende Methode für die spätere Realisierung:

- `void fasid(Player player)`: Soll verwendet werden, wenn in Realisierungen dieser Schnittstelle ein Betrug aufgedeckt wird. Wird in der Implementierung in *Teilaufgabe d)* einen Alarm-Text anzeigen.

Hinweis: Die Implementierung aus Teilaufgabe d) wird in Teilaufgabe h) nochmals erweitert!

Teilaufgabe c) **[10%]**

Schreiben Sie eine Klasse **`Match`**, mit der ein zu betippendes Spiel dargestellt wird. Ein Match besteht aus den folgenden Attributen:

Der Name des Heim-Teams (`homeTeam`), der Name des Gast-Teams (`guestTeam`), das Datum des Spiels (`date`, als Zeichenkette), die Uhrzeit des Spiels (`time`, als Zeichenkette), der Name des Stadions in dem das Spiel stattfindet (`stadium`), das Ergebnis des Spiels (`result`, als Zeichenkette) und die Art Ergebnisses (`resultType`, *vgl. Teilaufgabe a)*)

Die Klasse soll einen Konstruktor bereitstellen, welcher die oben genannten Attribute mit Ausnahme der Art des Ergebnisses (`resultType`) *in exakt dieser Reihenfolge* entgegennimmt (vgl. Nutzung des Konstruktors in Methode `parseMatch` in bereitgestellter Klasse `FasidWorldCup`).

Die Art des Ergebnisses ist initial immer auf `ResultType.UNKNOWN` zu setzen.

Weiterhin soll die Klasse die Methode `toString` überschreiben und dort eine Aneinanderreihung der für die Anzeige typische Schreibweise „`$homeTeam : $guestTeam`“ *(vgl. Bild auf Seite 1)* zurückliefern.

Darüber hinaus ist eine Methode `boolean hasResult()` umzusetzen, die genau dann `true` liefert, wenn die `result`-Instanz-Variable einen Wert **ungleich** `null` hat, sonst `false`**.**

Teilaufgabe d) **[23%]**

Für das Hauptfenster implementieren Sie eine Klasse **`FasidTerm`**, über welches die Spieler ihre Wetten starten können. Die Klasse ist eine Ausprägung der in *Teilaufgabe b)* angelegten Schnittstelle `CorruptionPrevention`.

Die Klasse stellt weiterhin einen Konstruktor bereit, welcher die beteiligten Spieler sowie eine Liste aller zu tippenden Matches entgegennimmt und diese jeweils in Instanzvariablen abspeichert (vgl. Benutzung des Konstruktors in bereitgestellter Klasse `FasidWorldCup`). Darüber hinaus soll der Konstruktor zu Beginn prüfen, ob mindestens ein Match bzw. Spieler vorhanden ist. Andernfalls soll eine selbst zu schreibende, sprechende **`FasidException`** geworfen werden (aus der Fehlernachricht muss hervorgehen, was fehlt!).

`FasidTerm` stellt eine grafische Oberfläche für die Anwendung bereit. Der Titel der Oberfläche soll mit dem Namen „FA$ID® World Cup“ (siehe Hinweise) initialisiert werden. Weiterhin besitzt sie am oberen Rand einen Button für jeden teilnehmenden Spieler (vgl. Screenshot).

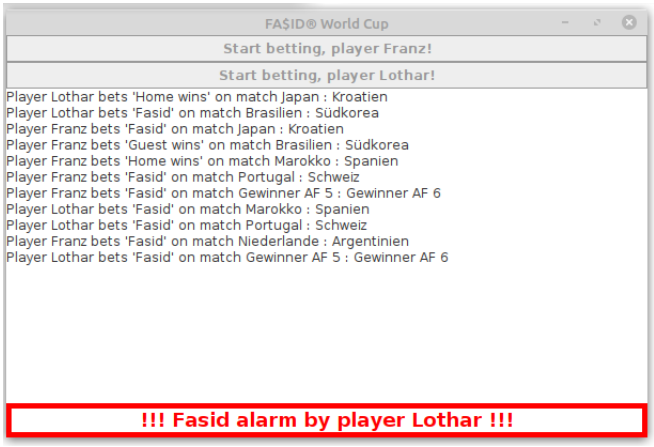
Beim Anklicken eines Buttons soll sich für den entsprechenden Spieler ein `BettingTerm` öffnen (vgl. *Teilaufgabe e)*), welchem der betreffende Spieler, die Liste der Matches sowie eine Referenz auf das `FasidTerm` übergeben werden müssen. Außerdem ist der Button eines Spielers nach dem Anklicken zu deaktivieren, um zu vermeiden, dass sich für den Spieler weitere Fenster öffnen.

Zusätzlich stellt die grafische Oberfläche einen Textbereich bereit, welcher die angenommenen Wetten logbuchartig aufschreibt und somit nachvollziehbar – Transparenz ist der FIFA ja sehr wichtig! – vermerkt (vgl. Screenshot).

Im Geiste der Transparenz soll im unteren Bereich auch ein `FasidAlarmLabel` (bereitgestellte Klasse mit grafischem Oberflächenelement, s. USB-Stick) vorgesehen werden. **Achtung**: dieses ist zunächst *nicht optisch wahrzunehmen* und nur bei einem Alarm (vgl. Screenshot) zu sehen!

Ebenso ist die folgende Methode zu realisieren: `void reportBet(Player player, Match match)`

Sie schreibt eine Textzeile in den Logbereich der grafischen Oberfläche, die Auskunft über den Tipp gibt, d.h. Spieler, Partie und Art des Tipps enthält (Beispiel siehe Screenshot). Der Text soll jeweils *in einer neuen Zeile* an den bereits vorhandenen Text *angehängt* werden.



Außerdem soll die Methode für den Fall, dass ein Spieler die Option „Fasid“ beim Wetten gewählt hat, prüfen, ob es sich tatsächlich um einen Betrug handelt. Dazu soll vereinfacht angenommen werden, dass in 30% aller Fälle ein Betrug vorliegt.

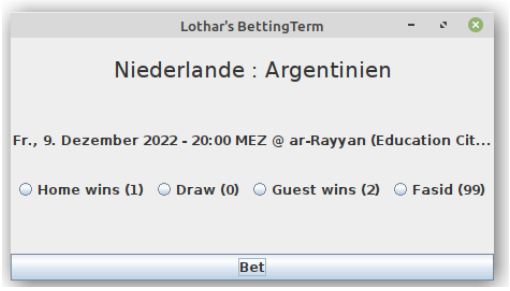
Im Betrugsfall ist die Methode `fasid`, welche durch die Schnittstelle vorgegeben ist, aufzurufen. Dort nutzen Sie die Methode `setFasidAlarm` des `FasidAlarmLabel`s mit dem Namen des Spielers, der getippt hat als Argument (Achtung: die UI fängt an zu blinken 😉). Sobald nach dem Bekanntwerden eines Betrugs die nächste Wette entgegengenommen wird, soll durch den Aufruf der Methode `hideFasidAlarm` des `FasidAlarmLabel`s der blinkende Alarm wieder deaktiviert werden und das Label ist wieder „unsichtbar“ (letzteres muss nicht in der `fasid`-Methode geschehen).

Hinweis 1: Die hier erstellte Funktionalität wird in Teilaufgabe g) und h) noch erweitert!
Hinweis 2: Das ®-Zeichen können Sie mit `\u00AE` in einer Zeichenkette einfügen (es darf aber auch weggelassen werden)

Teilaufgabe e) **[14%]**

Als weitere grafische Oberfläche zur Verwaltung der Wettentgegennahme soll eine Klasse **`BettingTerm`** *(vgl. Screenshot auf der Rückseite)* implementiert werden. Diese Klasse besitzt als Instanzvariablen mindestens eine Referenz auf den Spieler, auf den sich das Fenster bezieht `(player)`, eine Referenz auf das entsprechende `FasidTerm` sowie eine Referenz auf die Liste aller Spiele `(matches)`. Diese Variablen sind vom Konstruktor bei der Initialisierung entgegenzunehmen (vgl. *Teilaufgabe d)*).

Weiterhin soll der Titel des Fensters auf den Namen des zugehörigen Spielers gesetzt werden. Das Fenster zeigt einerseits an, auf welches Spiel aktuell gewettet wird (obere Zeile) sowie auch weitere Informationen zu diesem Match *(mindestens Datum, Uhrzeit und das Stadion, in dem das Spiel stattfindet, vgl. Screenshot)*. Außerdem soll das Fenster für die Selektierung der entsprechenden Option, auf die der Spieler wetten kann, eine Reihe von `JRadioButton`s, von denen nur *maximal eine Option gleichzeitig* selektierbar sein soll. Die unterschiedlichen Optionen werden dabei durch die entsprechenden `ResultType`s aus *Teilaufgabe a)* repräsentiert (alle Typen außer `ResultType.UNKNOWN`). Das erste Spiel, auf das gesetzt werden soll, ist das erste Spiel in der übergebenen Liste aller Spiele, für welches kein Resultat vorliegt (`hasResult` liefert `false`, vgl. *Teilaufgabe d)*)



Um Wetten abzugeben, soll das Fenster einen Button bereitstellen, der die selektierte Option übernimmt und die Wette für das angezeigte Spiel entsprechend weitergibt. Dazu soll zunächst geprüft werden, ob der Spieler eine Option selektiert hat. Ist dies nicht der Fall, ist dem Spieler mittels eines Dialogfensters mitzuteilen, dass er eine Option auswählen muss (vgl. Screenshot).

Anschließend ist der `ResultType` des aktuell angezeigten Matches auf den vom Spieler ausgewählten Typ zu setzen. Weiterhin ist im `FasidTerm` der entsprechende Eintrag in den Logbereich zu schreiben (Aufruf der `reportBet`-Methode, vgl. *Teilaufgabe d)*). Zuletzt muss das nächste Match aus der Liste angezeigt werden, auf das gesetzt werden kann. Sollten keine weiteren Matches zum Setzen mehr vorhanden sein, soll der „Bet"-Button *deaktiviert* werden.

Hinweis: Ob ein `JRadioButton` ausgewählt ist, können Sie mit der Methode `radioBtn.isSelected()` auslesen.

Teilaufgabe f) **[5%]**

Ändern Sie die Implementierung der `loadMatches`-Methode in der bereitgestellten Klasse `FasidWorldCup` (s. USB-Stick) so ab, dass die auf dem Stick bereitgestellte Datei `fasid-matches.csv` zeilenweise eingelesen wird. Fehler müssen dabei zwar abgefangen werden, müssen aber nicht weiter behandelt werden.

Dabei ist jede Zeile der Datei in ein `Match`-Objekt umzuwandeln und *anstatt der Beispieldaten* der Liste der Matches hinzuzufügen. Nutzen Sie dazu die bereits zur Verfügung gestellte Methode `parseMatch`, welche bereits für die Umwandlung der Beispieldaten genutzt wird.

Teilaufgabe g) **[5%]**

Erweitern Sie die `reportBet`-Methode von `FasidTerm` (siehe *Teilaufgabe d)*) so, dass die Textzeile aus der Methode `reportBet`, welche in den Logbereich von `FasidTerm` geschrieben wird, auch jeweils in eine Textdatei mit dem Namen `fasid.txt` geschrieben wird. Sollte die Textdatei bereits existieren, ist die Nachricht als neue Zeile an das Ende *anzuhängen*. Fehler müssen abgefangen, jedoch nicht weiter behandelt werden.

Teilaufgabe h) **[5%]**

Erweitern Sie die Klasse `FasidTerm` (vgl. *Teilaufgabe d)*) so, dass das `FasidAlarmLabel`, dessen Alarm bei jedem erneuten Wetten wieder versteckt wird, nach dem Auftreten eines Betrugs nach *fünf Sekunden* automatisch wieder versteckt wird, unabhängig davon, ob in der Zwischenzeit andere Wetten angenommen wurden.

Diese Aktion soll *nebenläufig* durchgeführt werden. Die nebenläufige Ausführung endet, nachdem die fünf Sekunden abgelaufen sind und der Alarm versteckt wurde.

Allgemeine Hinweise

Starten
Starten Sie die Anwendung mit der gegebenen Klasse `FasidWorldCup` (siehe USB-Stick).

Schließen eines Fensters
Beim Schließen eines Fensters soll die komplette Anwendung beendet werden.

Sichtbarkeit von Instanz-Attributen
Sämtliche Instanz-Attribute sind als privat zu definieren und von außerhalb der Klasse ggf. mittels Getter- und/oder Setter-Methoden zu verwenden.