Programmieren 2 Labor - WiSe 2017/18
Fakultät Informatik - UIB
Prof. Dr. Frank Dopatka
Abgabe bis 27.10.2017

hochschule mannheim

Übungsblatt 1: Prozedural & Objektorientiert

25 Punkte möglich

Aufgabe 1: Aus dem Skript

[2 Punkte] Liefern Sie als Gruppe alle Quellcodes aus dem Skript, Kapitel 1 in einem Eclipse-Workspace als ZIP-Datei bei Prof. Dopatka ab. Der Eclipse-Workspace muss eine wohl strukturierte Package-Struktur aufweisen. Sind Quellcodes fehlerhaft oder unvollständig, so erhalten Sie keine Punkte.

Aufgabe 2: Live-Coding

Bei der Abnahme des Aufgabentyps „Live-Coding“ erstellen Sie an einem einzigen PC bei Prof. Dopatka Quellcode ohne weitere Hilfsmittel. Der Eclipse-Workspace muss geöffnet, aber leer sein. Weitere Dokumente, online oder in Papierform, sind untersagt. Prof. Dopatka prüft anhand dieses Tests, ob er Ihrem Kleinunternehmen[1] einen folgenden Großauftrag vergibt oder ob er Ihr Kleinunternehmen dazu nicht in der Lage sieht. Den geforderten Quellcode erstellen Sie zügig live während der Abgabe und erklären währenddessen Ihre Vorgehensweise. Prof. Dopatka kann alternative Vorgehens-weisen zur Lösung des Problems fordern.

2a) [3 Punkte] Erstellen Sie eine **main**-Methode, die eine **verarbeite**-Methode aufruft.
Die **verarbeite**-Methode erhält ein beliebiges zu verarbeitendes 2-dimensionales int-Array als Eingabe-Parameter.
Prof. Dopatka wählt aus, ob er

- das Minimum,
- das Maximum oder
- den arithmetischen Mittelwert

als Ergebnis der Verarbeitung in Form des Rückgabewertes der **verarbeite**-Methode von Ihnen codiert haben will. Dieser Rückgabewert ist dann über die **main**-Methode auf der System-Konsole auszugeben.

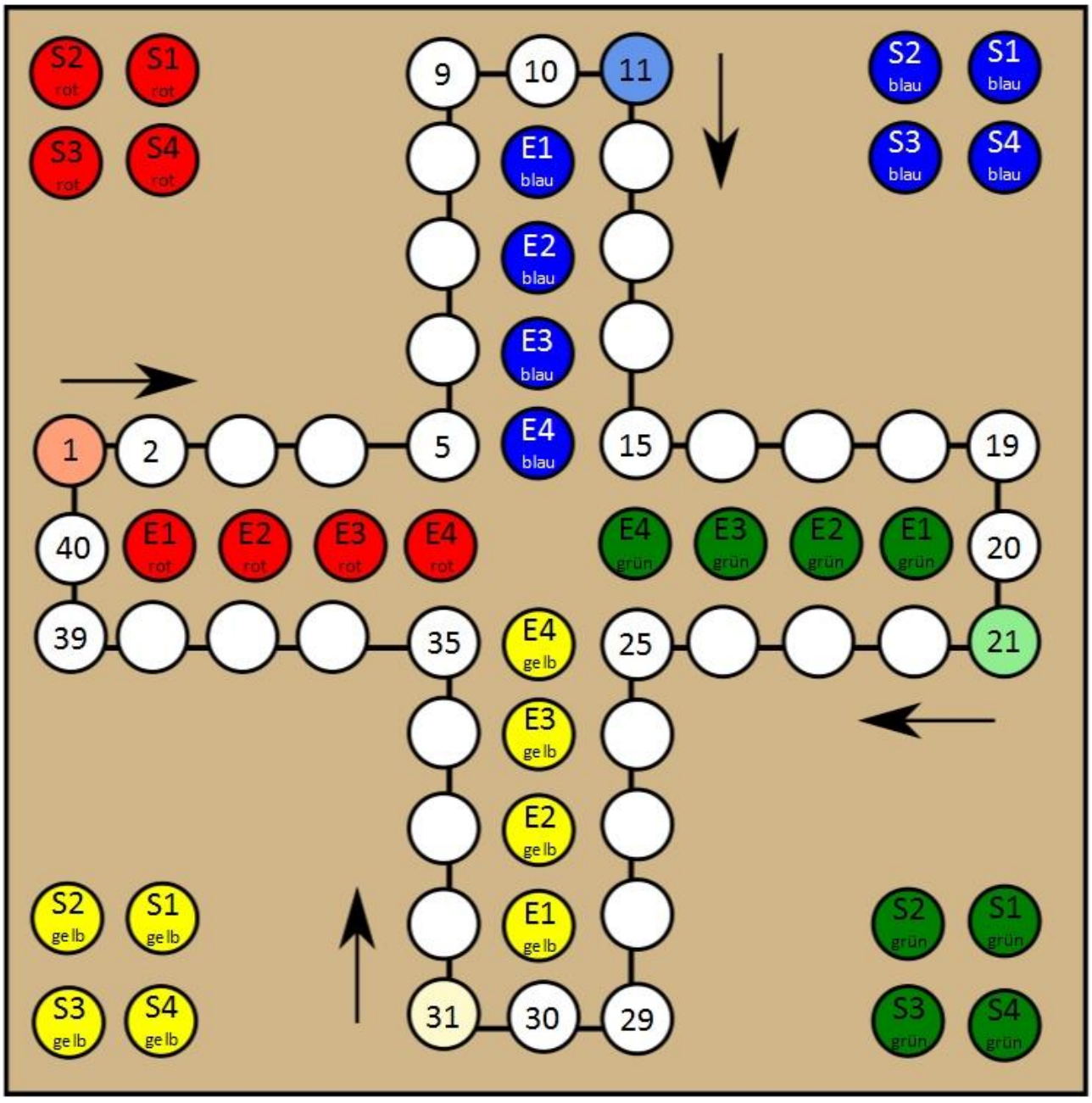
2b) [3 Punkte] Erstellen Sie die Funktion **getAnzGrößerAls**, die ein beliebiges 2-dimensionales Array von **int**-Zahlen sowie eine **int**-Zahl x als Eingaben in der **main**-Methode erhält und eine **int**-Zahl zurückgibt. Rufen Sie diese Funktion aus der **main**-Methode auf. Sie ermittelt alle Zahlen in dem Array, die größer sind als **x** und gibt diese Anzahl zurück. **NULL**-Werte sind einfach zu überspringen ohne eine Fehlerbehandlung. Existieren keine Zahlen im Array, so ist 0 zurückzugeben. Wird bei einer Eingabe eine Exception geworfen, so erhalten Sie keine Punkte.

[1] Ihr Kleinunternehmen ist Ihre Übungsgruppe. Sobald Sie etwas abgeben, ist Prof. Dopatka Ihr Großkunde. Treten Sie also entsprechend freundlich, team-fähig, organisiert und sehr gut vorbereitet gegenüber dem Großkunden auf, denn die Existenz Ihres Kleinunternehmens ist von diesem Großkunden abhängig. Sie sind Dienstleister des Großkunden!

hochschule mannheim

Aufgabe 3: Das Spiel

Ihre Aufgabe lautet, das Spiel „Mensch ärgere dich nicht“ strikt gemäß der folgenden Karte zu implementieren:



Dabei bedeutet

- 1 bis 40: reguläre Spielfelder für alle Spieler
- S1 bis S4: Startfelder des Spielers mit der entsprechenden Farbe
- E1 bis E4: Endfelder des Spielers mit der entsprechenden Farbe

Die strikt zu implementierenden Spielregeln lauten wie folgt:

1. Das Ziel des Spieles besteht darin, die vier eigenen Spielfiguren von den Startfeldern (S_i) auf die Endfelder (E_i) zu ziehen. Dazu müssen die Figuren das Spielbrett einmal in der vorgegebenen Richtung umrunden.
2. Über die Anzahl der zu ziehenden Felder eines Spielers pro Runde entscheidet ein Würfel.
3. Es wird reihum gewürfelt (Spieler 1,2,3,4,1 usw.) und gesetzt.
4. Wer eine Sechs würfelt, kann die eigene Spielfigur aus dem nächsten Startfeld (S_i) heraus auf sein Start-Spielfeld stellen oder eine vorhandene Figur auf einem Spielfeld vorrücken. Danach darf er nochmals würfeln.
5. Das Start-Spielfeld ist
    - 1 für den Spieler mit den roten Spielfiguren,
    - 11 für den Spieler mit den blauen Spielfiguren,
    - 21 für den Spieler mit den grünen Spielfiguren und
    - 31 für den Spieler mit den gelben Spielfiguren.
6. Kommt beim Umlauf eine Spielfigur auf ein Feld, das bereits von einer gegnerischen Spielfigur besetzt ist, gilt die gegnerische Figur als geschlagen und muss zurück auf ihr nächstes freies Startfeld S_i.
7. Eigene Figuren können nicht geschlagen werden: ist das Zielfeld bereits mit einer eigenen Figur besetzt, darf der Zug weder ausgeführt noch neu gewürfelt werden, sondern es ist der nächste Spieler an der Reihe.
8. Hat ein Spieler mehrere Spielfiguren im Umlauf, kann er frei entscheiden, mit welcher er ziehen möchte. Ein Würfelwurf darf nicht auf mehrere Figuren aufgeteilt werden.
9. Ein Überspringen im Bereich der farbigen Endfelder (E_i) ist nicht erlaubt. Es dürfen nur Startfelder und Endfelder der eigenen Farbe betreten werden.

3a) [10 Punkte] Richten Sie ein neues Projekt MADN (Mensch-Aergere-Dich-Nicht) in Eclipse ein. Für die Implementierung der folgenden Basisklassen erhalten Sie jeweils 2 Punkte, falls diese in jeweils einer **main**-Methode für den Kunden ausreichend getestet wurden:

- Implementieren Sie die Klasse **Würfel**. Ein Würfel soll beim Aufruf der Methode „werfen“ eine Ganzzahl zwischen 1 und 6 zurückgeben. Der Würfel soll bei Bedarf so manipulierbar sein, dass er eine beliebig lange, vordefinierte Folge von Zahlen würfelt.
- Implementieren Sie die Klasse **Spielfigur**. Jede Spielfigur hat eine Farbe, die aus einer FarbEnum (rot,blau,grün,gelb) zu wählen ist. Außerdem kann eine Spielfigur eine gültige Position auf dem Spielbrett haben.
- Implementieren Sie die Klasse **Spielbrett**. Das Spielbrett beinhaltet die im Folgenden beschriebenen Spielfelder.
- Implementieren Sie die Klasse **Spielfeld**. Jedes Spielfeld hat eine ID (siehe Bild). Außerdem kann ein Spielfeld keine oder genau eine Spielfigur kennen.
- Implementieren Sie die Klasse **Spieler**. Jeder Spieler hat einen Namen, eine Farbe aus der FarbEnum, 4 Spielfiguren seiner Farbe und einen Würfel.

3b) [1 Punkte] Implementieren Sie die Klasse **Spiel**. Jedes Spiel kennt ein Spielbrett, 1 bis 4 Spieler und den Spieler, der gerade am Zug ist.

3c) [6 Punkte] Implementieren Sie die Testklasse **SpielTest** mit einer **main**-Methode, die ausschließlich ein Spiel-Objekt kennt. Alle Aktionen laufen über die Spiel-Klasse, so dass sie über die Spiel-Klasse alle Aktionen des Spiels durchführen können. Es werden ausschließlich primitive Typen sowie String-Objekte von und an die Spiel-Klasse übergeben.
Weisen Sie unter Verwendung von realen Szenarien nach, dass Ihr Spiel gemäß der Regeln funktioniert. Verwenden Sie nach jedem Zug eine Ausgabe auf der System-Konsole. Verwenden Sie dazu auch die erste Seite dieser Aufgabenstellung als reales Spielbrett und bringen Sie entsprechende Spielfiguren mit, um die Züge nachzuvollziehen.

Programmieren 2 Labor - WiSe 2017/18

Fakultät Informatik - UIB
Prof. Dr. Frank Dopatka
Abgabe bis 27.10.2017

WICHTIG:

Sie können erst dann Punkte erhalten, wenn Sie sich im SVS registriert haben und einer Übungsgruppe zugeordnet sind.

Die Umsetzung der Spiel-Modellierung dient als Grundlage für die weiteren Übungsblätter! Implementieren Sie daher besonders sorgfältig, denn Sie müssen den Rest vom Semester mit Ihrem eigenen Code leben!

Die Verwendung von Quellcode von anderen Personen außerhalb Ihres Teams ist nicht gestattet und gilt als Täuschungsversuch! Es wird nochmals darauf hingewiesen, dass dass jedes Teammitglied die Implementierung vollständig verstanden haben muss! Das individuelle Testen dieser Kenntnis wird vorbehalten!

Jedes UML-Diagramm und jede Dokumentation ist digital in einem lesbaren Format (JPG, PNG, PDF, VSD, DOCX), beim Kunden abzugeben.

Wenn Sie in diesem Blatt nicht alle Anforderungen an das Spiel realisieren und eine Aufgabe im nächsten Übungsblatt baut darauf auf, so haben Sie diese Anforderungen bis zur Abgabe des nächsten Übungsblattes nachzuimplementieren! Fehlen dann die Funktionen dann immer noch, so gibt es erneut Punktabzug!

Es ist notwendig, das bereitgestellte Vorlesungsskript vorzuarbeiten, um in der Vorlesung rechtzeitig offene Fragen stellen zu können. Ansonsten wird die vollständige Fertigstellung der Aufgaben sehr schwierig! Arbeiten Sie ab jetzt ständig, eigenhändig und konzentriert!