Wiederholung: Datentypen, Arrays, Kontrollstrukturen, Methoden(1-35)

Neue Themen: Structs & Enums (36-41)

Console Application

Created by http://twitch.tv/mostwanted38/ - https://discord.gg/WMFucAD - 21.07.2020

Schiffe versenken

Implementieren Sie den Klassiker „Schiffe versenken“. Dabei wird unterstellt, dass die Anzahl der Spiel 2 ist. Beide haben jeweils ein eigenes Spielfeld der Größe 10x10.

Aufgaben (1-41):

1) Überlegen Sie welchen Datentyp Sie für dieses Projekt am besten verwenden sollten. Bedenken Sie dabei, dass Sie geeignete Werte finden müssen für die einzelnen Felder der Spielfelder. Folgende Repräsentationen auf den jeweiligen Feldern [X,Y] müssen sie abbilden können für dieses Projekt:
   - Das Feld ist Wasser.
   - Auf diesem Feld sitzt ein Schiff von Spieler 1
   - Auf diesem Feld sitzt ein Schiff von Spieler 2
   - Auf dieses Feld hat Spieler 1 angegriffen & Spieler 2 getroffen!
   - Auf dieses Feld hat Spieler 2 angegriffen & Spieler 1 getroffen!
   - Auf dieses Feld hat Spieler 1 angegriffen & nichts getroffen!
   - Auf dieses Feld hat Spieler 1 angegriffen & nichts getroffen!

   Legen Sie für alle 7 gegebenen Spielfeld-States eine **const** Variable ihres gewählten Datentyps an und weisen sie diesen direkt einen Wert zu beim Anlegen!

2) Legen Sie das Spielfeld für Spieler 1 in einer Variable namens „spielfeld1“ an.
   Legen Sie das Spielfeld für Spieler 2 in einer Variable namens „spielfeld2“ an.
   Diese sind ein 2D-Array vom gewählten Datentyp in Aufgabe 1.
   Schreiben sie weiterhin **2 globale const Variablen** vom Typ **int**:
   public const int spielfeldbreite = 10;
   public const int spielfeldhöhe = 10;
   Schreiben sie eine Methode „**SpielfelderErstellen**“.
   Diese soll die alle Felder auf allen Spielfeldern komplett mit dem Wert Wasser versehen.

3) Implementieren Sie 2 Methoden um die Spielfelder auszugeben!
   - Schreiben Sie eine Methode namens „**SpielfeldAusgeben**“ welche ein referenziertes Spielfeld annimmt und auf der Console ausgibt.
   - Schreiben sie zusätzlich eine Methode „**AlleSpielfelderAusgeben**“, die die Spielfelder aller Spieler ausgibt!

Überlegen Sie wie diese Methoden in Kombination verwendet werden.

4) Schreiben Sie eine Methode namens „**AnzahlSchiffeBestimmen**“ die eine angegebene Anzahl der zu spawnenden Schiffe im Programm setzt.
Legen sie dazu 2 globale Variablen an mit den Namen „AnzahlDerSchiffeVonSpieler1“ und „AnzahlDerSchiffeVonSpieler2“ und setzen Sie diese über den einen Methoden-Parameter.

5) Schreiben sie dazu noch eine Methode „**AnzahlSchiffeEinlesen**“, die den Benutzer eine einzige Zahl einlesen lässt um die Anzahl der zu spawnenden Schiffe *beider Spieler gleichzeitig* zu bestimmen. Diese Methode soll bei Spielstart aufgerufen werden.

   Überlegen Sie ob sie „**int.Parse(string)**“ oder „**int.TryParse(string, out int)**“ verwenden sollten und welche Vorteile das eine gegenüber dem anderen hat.
   Schauen Sie sich an welche Rückgabewerte diese Methoden haben und wie sie diese weiterführen einbinden können in Anweisungen oder Schleifen.
   Beschäftigen Sie sich mit den Unterschieden zwischen den **ref**, **out** und **in** Schlüsselwörtern.
   Zum Erlernen der Unterschiede probieren Sie folgenden Code in Ihrem Projekt aus.
   Schauen Sie welche Stellen klappen & Kommentieren Sie Nicht funktionerende aus.

```
static void testen_callbyreference()
{
        int zahl1 = 1; int zahl2 = 5; int zahl3 = 8;
        test_ref(ref zahl1);
        test_out(out zahl2);
        test_in(in zahl3);
}
static void test_ref(ref int x)
{
        var y = x; //Klappt das?
        x = 42; //Klappt das?
}
static void test_out(out int x)
{
        var y = x; //Klappt das?
        x = 42; //Klappt das?
}
static void test_in(in int x)
{
        var y = x; //Klappt das?
        x = 42; //Klappt das?
}
```

6) Schreiben Sie nun zusätzlich eine Methode namens „**AnzahlSchiffeFürSpielerBestimmen**“ die den Benutzer einen Wert eingeben lässt um eine referenzierte Anzahl von Schiffen (siehe Aufgabe 4) zu überschreiben mit der eingelesenen Zahl. Überlegen Sie was mit referenziert gemeint ist und fangen Sie ungültige Eingaben des Benutzers ab! (zB „-1“, „-5“, „-10“, usw..).

7) Schreiben sie eine Methode „**SpielfeldAufraeumen**“, welche alle Felder eines referenzierten Spielfelds auf den Wert für „Wasser“ setzt. Können Sie diese nutzen bei der zuvor geschriebenen Methode „SpielfelderErstellen“?

8) Schreiben sie eine Methode „**PositionIstInnerhalbDesSpielfelds**“ die überprüft ob eine angegebene Position (X, Y) sich innerhalb des Spielfelds befindet oder nicht. Lassen Sie die Methode „Ja“ oder „Nein“ zurückgeben. (Spielfeldgrenzen = Aufgabe2)

9) Schreiben sie eine Methode namens „**PositionIstAußerhalbDesSpielfeldes**. Diese soll die umgekehrte Logik der vorherigen Aufgabe haben. Wenn nicht im Feld, dann „Ja“, ansonsten „Nein“. Überlegen Sie wie Sie diese Logik implementieren müssen. Können Sie die Methode der vorherigen Aufgabe nutzen?

10) Schreiben Sie eine Methode „**DasAngegbeneFeldHabeIchBereitsBesetzt**“ welche überprüft ob das angegebene Feld(X, Y) eines referenzierten Spielfelds von dem referenzierten Spieler bereits besetzt wurde.
Überlegen Sie Welche der Methoden aus Aufgabe 8 oder 9 Ihnen hier helfen!
Je nachdem ob besetzt oder nicht, soll sie „Ja“ oder „Nein“ zurückgeben.

11) Schreiben Sie eine Methode „**DasAngegebeneFeldIstBereitsBesetzt**“ welche überprüft ob das Feld an einer angegebenen Position(X, Y) auf einem referenzierten Spielfeld bereits durch irgendeinen beliebigen Spieler besetzt wurde.
Fragen Sie dabei auch gleichzeitig nach ungültige Werten für X und Y die als Parameter mitgegebene werden könnten.
Erkennen Sie eine Gemeinsamkeit zu Aufgabe 10?
Welche Hilfsmethoden (vorherige Aufgaben) können Sie hier verwenden?

12) Schreiben Sie eine Methode „**DasAngegebeneFeldIstLeer**“ welche überprüft ob das Feld(X, Y) auf einem referenzierten Spielfeld leer ist.
Fangen Sie ungültige X und Y Werte entsprechend ab und Überlegen Sie mit welchem definierten Wert aus Aufgabe 1 sie sagen, dass ein Feld leer ist (kein Schiff, kein Angriffsergebnis)

13) Schreiben Sie eine Methode „**DasGesamteSpielfeldIstLeer**“ welche überprüft ob ein referenziertes Spielfeld von egal welchem Spieler kein einziges Schiff enthält.
Lassen sie die Methode auf die Beantwortung der Frage hin geeignete Werte zurückgeben.
Hinweis: Überlegen Sie welche Werte der Aufgabe 1 Ihnen sagen, dass dort kein Schiff **jetzt mehr steht** und dass dort auch ein Schiff **vorher nie stand.**

Schreiben Sie hierfür weiterhin für diese Aufgabe eine Hilfs-Methode „**ZaehleAnzahlLeererFelderAufDemSpielfeld**“ welche ein referenziertes Spielfeld nimmt & am Ende zurückgibt wie viele freie Felder (siehe Hinweis) es hat.

Beantworten Sie Anschließend die Frage ob „Das Gesamte Spielfeld leer ist“ mittels eines Vergleichs in der Methode „DasGesamteSpielfeldIstLeer“, welche die Hilfsmethode aufruft & mit einem Wert vergleicht.
Wie viele Leere Felder muss es auf dem Spielfeld geben damit ALLE leer sind?

14) Implementieren Sie Methoden um Schiffe diverser Schiffsgrößen (1, 2 und 3) in diversen Richtungen (Horizontal, Vertikal) auf dem einem referenzierten Spielfeld für einen angegebenen Spieler zu spawnen.
Hinweis:
Y = Horizontal, X = Vertikal.
Da wir kein wirkliches Koordinaten System haben orientieren wir uns an den Indexen eines 2D Arrays. Nach unten erhöht sich der X Index, nach rechts vergrößert sich der Y Index.
So ist ein Schiff was Vertikal gespawnt wird innerhalb einer „Spalte“ des Arrays & ein Schiff was Horizontal gespawnt werden soll innerhalb einer „Zeile“ des Arrays
   a) Schreiben Sie eine Methode namens „**SpawneKleinesSchiff**“ welche an angegebener Position (X, Y) für den angegebenen Spieler auf dem gewählten Spielfeld ein kleines Schiff spawnt.
   b) Schreiben Sie eine Methode namens „**SpawneMittleresSchiff_Vertikal**“ welche an angegebener Position (X, Y) sowie (X+1, Y) für den angegebenen Spieler auf dem gewählten Spielfeld ein mittleres Schiff spawnt.
   c) Schreiben Sie eine Methode namens „**SpawneGroßesSchiff_Vertikal**“, welches an angegebener Position (X, Y) sowie (X+1, Y) und (X-1, Y) für den angegebenen Spieler auf dem gewählten Spielfeld ein großes Schiff spawnt.
   d) Schreiben Sie eine Methode namens „**SpawneMittleresSchiff_Horizontal**“ welche an angegebener Position (X, Y) sowie (X, Y+1) für den angegebenen Spieler auf dem gewählten Spielfeld ein Mittleres Schiff spawnt.
   e) Schreiben Sie eine Methode namens „**SpawneGroßesSchiff_Horizontal**“, welches an angegebener Position (X, Y) sowie (X, Y+1) und (X, Y+2) für den angegebenen Spieler auf dem gewählten Spielfeld ein Großes Schiff spawnt.
   f) <Wird in einer späteren Aufgabe erstellt>
   g) <Wird in einer späteren Aufgabe erstellt>

(siehe nächste Seite)

Schreiben sie 2 globale Variablen mit dem Namen „anzahlVerfuegbarerSchiffe“ und „anzahlGespawnterSchiffe“, die sich jeweils 2 Werte merken. (2x 2Werte)

In allen 5 Methoden (a, b, c, d, e) soll immer dann, wenn 1 Feld besetzt wird, die „anzahlVerfuegbarerSchiffe“ am jeweiligen Index des angegebenen Spielers um 1 reduziert werden und „anzahlGespawnterSchiffe“ am Index des angegebenen Spielers um 1 nach oben gezählt werden.

Bei einem Mittleren Schiff senken wir die erstgenannte Variable um 2 und erhöhen die 2. Variable um 2. (da ein Mittleres Schiff 2 Felder groß ist)
Bei einem Großen Schiff senken wir die erstgenannte Variable um 3 und erhöhen die 2. Variable um 3 (da ein Großes Schiff 3 Felder groß ist)

Fangen Sie bei allen 5 Methoden ungültige Situationen & Werte ab! Überlegen Sie welche Methoden aus den vorherigen Aufgaben Ihnen hierbei helfen kann!

15) Generieren Sie Zufällige X und Y Positionen! (erst komplett lesen, dann implementieren)
    - Schreiben Sie eine Methode „**ZufaelligeXPositionGenerieren**“ welche eine Zufällige X Position auf Basis der Höhe des Spielfelds generiert und zurückgibt.
    - Schreiben Sie eine Methode „**ZufaelligeYPositionGenerieren**“ welche eine Zufällige Y Position auf Basis der Breite des Spielfelds generiert und zurückgibt.
    - Schreiben Sie eine Methode „**ZufaelligeSpielfeldPositionGenerieren**“ welche die beiden Methoden für das Generieren von X und Y Werten einbindet & ein Array mit 2 Generierten Werten, also X und Y, zurückgibt

    Als Zufallsgenerator wird die **class** **Random** aus dem Namespace **System** verwendet.
    using System;
    Random random = **new** **Random**();
    int zufallszahl = random.Next(zahl1, zahl2);
    //Generiert eine Zahl zwischen zahl1 und zahl2 - 1 und gibt einen int zurück

16) Schreiben Sie weiterhin eine Methode namens **GeneriereZufallszahlen** welche ein Array von **int**-Werten zurückgibt für ein angegebenes Minium, ein angegebenes Maximum und eine angegebene Anzahl von Werten.
    Beispiel:
    - Sei das Minimum 2, das Maximum 12 & die Anzahl 15.
      So wird ein Array mit 15 verschiedenen zufallsgenerierten Zahlen von 2 bis 12 zurückgegeben.
    - Sei das Minimum 7, das Maximum 42 & die Anzahl 31
      So wird ein Array mit 31 verschiedenen zufallsgenerierten Zahlen von 7 bis 42 zurückgegeben.

    Bedenken Sie dabei welche Parameter die Methode „Next“ der Klasse Random hat und in welchem Bereich die Zufallsgenerieren Zahlen liegen.

17) (Umfangreiche Aufgabe! Erst einmal komplett lesen, dann nachdenken)

Schreiben Sie eine Methode namens „**ZufaelligSchiffeSpawnen**“, die für eine angegebene Schiffsgröße (1, 2 oder 3), eine angegebene Anzahl zu spawnender Schiffe auf einem referenzierten Spielfeld für einen referenzierten Spieler entsprechend viele Schiffe spawnt.

Überlegen Sie während sie den nachfolgenden Text lesen welche Lokalen Variablen sie für diese Aufgabe benötigen. (Das ist keine Fangfrage! Sie benötigen welche!)

(1) Zuerst soll eine Zufalls-Koordinate[X,Y] generiert werden.

(2) Im Anschluss soll, für *ALLE BENÖTIGTEN* Koordinaten[X,Y] beim Spawnen einer Schiffsgröße überprüft werden ob Feld[X,Y] sich innerhalb des Spielfelds befindet & ob das Feld[X,Y] auch noch leer ist.
   (a) Wenn nicht soll eine weitere Position generiert werden & dieser Vorgang wiederholt werden.
   (b) Wenn ja dann soll ein Schiff des referenzierten Spielers auf dem Zufalls-Feld[X,Y] & allen benötigten Nachbar-Feldern[X,Y] gespawnt werden. Durch Aufgabe 14 wird beim Spawnen „anzahlVerfuegbarerSchiffe“ und „anzahlGespawnterSchiffe“ entsprechend geupdated.

Dieser Vorgang geht solange wie der Spieler noch Schiffe spawnen kann!
Hinweis: anzahlVerfuegbarerSchiffe und anzahlGespawnterSchiffe

Passt die Schiffsgröße nicht mehr für die anzahlVerfuegbarerSchiffe**, so wird der Vorgang oben wiederholt, jedoch werden jetzt nur noch kleine Schiffe gespawnt.**

Damit die Aufgabe einfach bleibt, kann diese Methode Schiffe *NUR HORIZONTAL* spawnen (Siehe Spawn-Methoden in Aufgabe 14)

Idee für Kreative: Zufallsgenerator nutzen & zufällige Richtung bei jedem Spawnversuch generieren & entsprechende Methoden der Aufgabe 14 aufrufen

Denken Sie zusätzlich daran, dass wenn [0,0] zum Beispiel als Position gewählt wird ein Schiff über die Stellen [0,-1] [0,0] [0,1] generiert werden würde wenn Schiffsgröße 3 ist. Müssen sie ggf. Spawn-Methoden aus Aufgabe 14 hier anpassen um dieses Problem zu vermeiden?

Verfolgen Sie auch folgende Spawn-Logik:

- Sei die Anzahl von Schiffen die gespawnt werden müssen 10. Die angegebene Schiffsgröße ist 3. Das bedeutet, dass 3 Große Schiffe (3x 3 Felder) und 1 kleines Schiff (1x 1 Feld) auf dem Spielfeld platziert werden müssen.
  (mehr Infos – siehe nächste Seite)

- Sei die Anzahl von Schiffen die gespawnt werden müssen 9. Die angegebene Schiffsgröße ist 2. Das bedeutet, dass 4 Mittlere Schiffe (4x 2 Felder) und 1 kleines Schiff (1x 1 Feld) auf dem Spielfeld platziert werden müssen
- Sei die Anzahl von Schiffen die gespawnt werden müssen 8. Die angegebene Schiffsgröße ist 1. Das bedeutet es werden 8 Kleine Schiffe (8x 1 Feld) zufällig auf dem Spielfeld platziert
- Sie die Anzahl von Schiffen die gespawnt werden müssen 7. Die angegebene Schiffsgröße ist 2. Das bedeutet es werden 3 Mittlere Schiffe (3x 2Felder) und 1 kleines Schiff (1x 1 Feld) auf dem Spielfeld platziert.
- Sie die Anzahl von Schiffen die gespawnt werden müssen 6. Die angegebene Schiffsgröße ist 3. Das bedeutet es werden 2 Große Schiffe (2x 3Felder) auf dem Spielfeld platziert.
- Sie die Anzahl von Schiffen die gespawnt werden müssen 5. Die angegebene Schiffsgröße ist 2. Das bedeutet es werden 2 Mittlere Schiffe (2x 2Felder) und 1 Kleines (1x 1Feld) auf dem Spielfeld platziert.

Führen Sie das Gedankenspiel durch zum besseren Verständnis der Spawn-Logik

Gedankenspiel:

Sei die Anzahl der zu spawnenden Schiffe A.
A könnte folgende Werte haben: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 oder 10.
Sei die Schiffsgröße B.
Die Schiffsgröße kann wie beschrieben die Werte 1, 2 oder 3 annehmen.

Frage: Wie viele Schiffe der jeweiligen Schiffsgröße B werden gespawnt für eine angegebene Anzahl A & wie viele Kleine Schiffe werden gespawnt?

Versuchen sie eine Gemeinsamkeit zu erkennen & bilden sie eine Mathematische Funktion **f(a, b)** = ….. welche Ihnen sagt wie viele Schiffe gespawnt werden.

Verfolgen Sie die 7 oben angegebenen Beispiele und ihre Erkenntnisse aus dem Gedankenspiel versuchen Sie anhand dieser…

a) eine einfache Berechnung für die Anzahl der zu spawnenden Schiffe mit jeweiliger Schiffsgröße zu erstellen
b) eine einfache Berechnung für die Anzahl der Kleinen Schiffe die gespawnt werden zu erstellen. Hinweisfrage: Diese werden nur gespawnt, wenn …?

Überlegen Sie weiterhin welche Ungültigen Werte es bei den Parametern „Schiffsgröße“ und „Anzahl zu Spawnender Schiffe“ geben kann.

Fangen Sie alle ungültigen Situation und Möglichkeiten ab indem Sie über ein
If(ungültige Situation)
  return;
die Methode sofort enden lassen!

Überlegen Sie zusätzlich auch welche unerwünschten Ergebnisse beim Spawnen passieren können und wie die Logik der Methode dann agieren soll.

18) Implementieren sie eine Methode namens „**HierWurdeBereitsAngegriffen**“ welche testet ob auf die angegebenen Ziel-Position des angegebenen Ziel-Spielfeld bereits eine Kanonenkugel vom referenzierten Spieler geflogen ist oder nicht.
In Aufgabe 1 sollten sie Werte für die diversen Fälle überlegen.
Wenn auf dem Feld[X,Y] bereits eine Kugel vom referenzierten gelandet ist (egal ob Treffer oder kein Treffer), soll die Methode „Ja“ zurückgeben. Ansonsten „Nein“!

19) Implementieren Sie eine Methode namens „**HierHabeIchEinSchiffZerstört**“ die prüft ob an angegebener Position (X, Y) auf einem referenzierten Ziel-Spielfeld die Kanonenkugel eines referenzierten Angreifenden-Spielers ein Schiff zerstört (getroffen) hat. Überlegen Sie welcher der 7 Werte aus Aufgabe 1 hier benötigt wird.

20) Implementieren Sie eine Methode namens „**HierHatIrgendjemandEinSchiffZerstört**“ („Hier war mal ein Schiff“) welche prüft ob auf einem referenzierten Ziel-Spielfeld an der angegebenen Position[X,Y] mal ein Schiff eines beliebigen Spielers war.
Überlegen Sie dabei welche in Aufgabe 1 definierten Werte ihnen Bescheid geben, dass dort ein Schiff war

21) Implementieren Sie eine Methode namens
„**ZaehleAnzahlVerbleibenderSchiffeAufDemSpielfeld**“ welche für ein angegebenes Spielfeld die Anzahl an noch existierenden (nicht zerstörten) Schiffen zählt. Dabei ist egal von welchem Spieler das Schiff ist.

22) Implementieren Sie eine Methode namens
„**AnzahlSchiffeAufZielSpielfeldVonSpielerX**“ welche die noch verbleibenden (nicht zerstörten) Schiffe eines angegebenen Spielers auf dem angegebenen Ziel-Spielfeld zählt und zurückgibt.

23) Schreiben Sie eine Methode namens „**IstDiesEinSchiffVonSpielerX**“ welche überpüft ob auf einem angegebenen Ziel-Spielfeld an einer angegebenen Position[X,Y] ein Schiff steht & ob dieses vom angegebenen Spieler ist. Lassen Sie die Methode „Ja“ oder „Nein“ zurückgeben.

24) Implementieren sie eine Methode „**Angreifen**“, die einen Angreifenden Spieler, einen Ziel-Spieler, ein Ziel-Spielfeld referenziert und eine Position[X,Y] als Parameter annimmt. Zusätzlich bekommt die Methode noch einen Parameter namens Diplomatie welcher per Standardwert auf „Aus“ gesetzt ist

(mehr Infos – siehe nächste Seite)

a) Finden Sie heraus wie sie Methoden-Standardwerte setzen!

b) Testen Sie dabei die Methoden Parameter auf ungültige Werte und lassen Sie die Methode sofort enden falls ungültige Werte angegeben wurden. Welche Werte können ungültig sein?

c) Stellen Sie sich vor Diplomatie wäre „An“ und es gäbe mehr als 2 Spieler.
Sie als Spieler 1 greifen eine Position[X,Y] auf einem Spielfeld an & erwarten, dass dort ein Schiff von Spieler 2 steht. Jetzt steht dort aber ein Schiff von Spieler 3, welcher Ihr Verbündeter ist. Wie muss die Methode agieren bei Diplomatie „An“ und „Aus“ wenn dieses Schiff nicht vom gewünschten Ziel-Spieler ist?

d) Greifen Sie die gewählte Position[X,Y] an.
Sollte dort kein Schiff stehen wird der Wert des Feldes[X,Y] auf „Kein Treffer“ gesetzt. Sollte dort ein Schiff stehen wird der Wer auf „Treffer“ gesetzt.
Beachten sie dabei die in Aufgabe 1 definierten Werte. Unterscheiden Sie wer der angreifende Spieler ist und setzen sie den entsprechenden Wert auf Basis des Ziel-Spielers der angegriffen wurde!

Entscheiden Sie sich hierbei für geeignete Datentypen in den Parametern & Rückgabewerten!

25) Implementieren sie eine Methode namens „**ZielAuswaehlenUndAngreifen**“, welche den Benutzer eine X und Y Koordinate, einen angreifenden Spieler & einen Ziel-Spieler eingeben lässt. Nach Eingabe dieser 4 Werte soll ein Angriff auf die Stelle [X,Y] des Spielfeldes des eingelesen Ziel-Spielers erfolgen. Alle 4 eingelesen Werte sollen als lokale Variablen vom Typ **int** gespeichert werden!
Schreiben Sie weiterhin eine Methode namens „**WaehleSpielfeld**“ welche für einen angegebenen Spieler (Spieler1 oder Spieler2) dessen Spielfeld als Referenz zurückgibt!

26) Implementieren Sie eine Methode namens „**ZufaelligerAngriff**“ die eine Zufallsposition (X, Y) auf einem zufälligen Ziel-Spielfeld für einen zufälligen Angreifen-Spieler & zufälligen Ziel-Spieler generiert dort „Angreifen“ nutzt.
Zusätzlich soll diese auch selbstständig entscheiden ob Diplomatie an oder aus ist. Nutzen sie dafür eine Zufallszahl zwischen 0 und 100. Sollte die Zufallszahl > 50 sein, wird Diplomatie berücksichtigt, ansonsten nicht.
Überlegen Sie welche Methoden von vorherigen Aufgaben hierfür verwendet werden können. Wann wurde bereits auf Diplomatie geachtet?
Denken Sie sich passende Werte für den Zufallsgenerator (**class** Random / Methode Next(a,b)) bei allen benötigten Zufallswerten aus!

27) Implementieren Sie eine Methode namens „**AngriffAnZufaelligerPosition**“ welche für einen referenzierten Angriffs-Spieler & ein referenzierten Ziel-Spielfeld eine Zufallsposition[X,Y] angreift. Dabei soll der Angriff nur dann erfolgen wenn das Feld[X,Y] noch nie angegriffen wurde

28) Implementieren Sie eine Methode namens „**HatVerloren**“, die für einen angegebenen Spieler prüft ob alle seine Schiffe zerstört worden sind. Entsprechend soll die Methode „Ja“ oder „Nein“ zurückgeben. Überlegen sie welchen Datentyp sie hier am besten verwenden & ob Sie Methoden aus vorherigen Aufgaben nutzen können.

29) Implementieren Sie eine globale Variable namens „AktuellerSpieler“ sowie eine Methode namens „**SpielerWechseln**“.
Die Methode soll den aktuellen Spieler von „Spieler1“ nach „Spieler2“ wechseln und umgekehrt. Diese Methode wird am Ende eines Spielzugs verwendet um den Angreifenden Spieler zu wechseln.

30) Implementieren Sie eine Methode namens „**Spielablauf**“ in der Sie ihr gesamtes Spiel implementieren und dessen Logik testen! Verwenden Sie die Methode „Spielablauf“ innerhalb der Main-Methode als einzige Methode!
(siehe nächste Seite)
Die Main-Methode sieht am Ende wie folgt aus:

```
public static void Main(string[] args)
{
        Spielablauf ();
}
```

31) Erweitern Sie den Spielablauf so, dass das Spiel wiederholbar ist.
Schreiben sie dafür eine neue Methode namens „**ErweiterterSpielablauf**“.
Diese beinhaltet den Aufruf von „Spielablauf“ und soll nach Beendigung einer Runde den Nutzer fragen ob er noch eine Runde spielen möchte. Wenn dem so ist, soll der Spielablauf wiederholt werden, andernfalls nicht.
Diese Methode soll nun anstelle der „Spielablauf“ Methode in der Main-Methode aufgerufen werden.

```
public static void Main(string[] args)
{
        ErweiterterSpielablauf ();
}
```

32) Erweitern Sie das Projekt indem sie eine neue Methode namens „**SpielzugZaehlen**“ anlegen, die immer dann, wenn ein Spieler seinen Zug gemacht hat die globale Variable mit dem Namen „**anzahlSpielzuege**“ am Index des aktuellen Spielers um 1 nach oben zählt. Überlegen Sie was mit Index gemeint ist und was die Variable ist. Implementieren sie die Logik in ihren „Spielablauf“.
Zusätzlich soll immer dann, wenn eine Spielrunde..
   a) neugestartet wird, eine Methode namens „**SpielzuegeZuruecksetzen**“ die Anzahl der bisherigen Spielzüge an allen Indexen auf 0 gesetzt werden.
   b) vorbei ist, eine Methode namens „**SpielzuegeZaehlen**“ die benötigten Spielzüge aller Spieler zählen & ausgeben!

33) **Erstellen Sie eine Kopie ihrer Methode „Spielablauf“ und benennen Sie die Kopie um in „AutomatischerSpielablauf**“. Ändern sie innerhalb dieser alles so um, dass das Spiel komplett ohne Benutzereingaben abläuft & so lange simuliert bis einer von beiden Spielern verloren hat. Zählen Sie dabei die Anzahl an Spielzügen mit! (siehe Aufgabe 32). Anstelle von einer wirklichen KI verwenden wir hier weiter den Zufallsgenerator um Felder auszuwählen. Verwenden sie die entsprechenden Zufalls-basierten Methoden der vorherigen Aufgaben um diese Simulation zu ermöglichen!

34) Erweitern Sie Ihr Spiel **von einem 2 Spieler** (1vs1) System **zu einem 4 Spieler System!**
Spieler 1 und Spieler 2 sind im „Team1“.
Spieler 3 und Spieler 4 sind im „Team2“ (2vs2).
Jeder Spieler hat ein eigenes Spielfeld & kann einen beliebigen Spieler des anderen Teams angreifen!
Überlegen Sie wie sie die Spielfelder absofort gespeichert werden!
Entscheiden sie ob Sie **(A)** neue Variablen „spielfeld3“ und „spielfeld4“ für die einzelnen Spielfelder nutzen wollen oder **(B)** ein einziges 2D Array für alle 4 Spielfelder nutzen sollen (Größe 40x10).

   Welche Vorteile hat diese Variante (B)? Welche Vorteile hat die Variante (A)? Überlegen Sie mit Bedacht wie sie das Projekt fortführen!

   Falls sie sich für die 2D-Array Variante entscheiden…

   - Überlegen Sie von welchem Index [X,Y] bis zu welchem Index [X,Y] das Spielfeld von Spieler1, Spieler2, Spieler3 und Spieler4 jeweils in diesem 2D Array der Größe (40 Zeilen x 10 Spalten) geht.

   - Schreiben sie eine Methode namens „**SpielfeldGrenzenLesen**“ welche für einen angegebenen Spieler (1, 2, 3 oder 4) die obere Linke Ecke (Index [X,Y]) sowie untere Rechte Ecke (Index [X,Y]) seines Spielfeldes zurückgibt.
     Können Sie für dieses Problem eine simple Funktion f(x) formulieren, die Ihnen für x (einen angegebenen Spieler) die beiden Positionen zurückgibt?
     (mehr Infos – siehe nächste Seite)

- Wie müssen Sie die Methode „WaehleSpielfeld“ aus Aufgabe 25 abändern damit Sie jetzt aus dem 40x10 2DArray das passende 10x10 2DArray lesen.

- Wie müssen Sie, wenn Sie Werte in dem geladenen 10x10 Array modifiziert haben, dieses wieder komplett zurück in das 40x10 Array schreiben? Schreiben Sie hierfür eine Methode namens „**SpielfeldUpdaten**“ welche anhand von einem angegebenen 10x10 Spielfeld sowie einem angegebenen Spieler die passenden Stellen des 40x10 Arrays überschreibt & updated.

Gehen Sie unabhängig von (A) oder (B) jetzt **gedanklich** ebenfalls durch welche Möglichkeiten für Angriffe durch das bilden der Teams jetzt möglich sind. Wer kann wen angreifen?

Lesen sie sich anschließend die weiteren Informationen durch.
Die nachfolgende Beschreibung erweitert die Feld[X,Y]-Zustände aus Aufgabe 1

Definieren Sie zunächst neue Werte für Repräsentationen und legen Sie entsprechend wie in Aufgabe 1 const Variablen für folgende Spielfeld-Zustände an!

- Auf diesem Feld sitzt ein Schiff von Spieler 3
- Auf diesem Feld sitzt ein Schiff von Spieler 4

- Auf dieses Feld hat Spieler 1 angegriffen & Spieler 3 getroffen!
- Auf dieses Feld hat Spieler 1 angegriffen & Spieler 4 getroffen!

- Auf dieses Feld hat Spieler 2 angegriffen & Spieler 3 getroffen!
- Auf dieses Feld hat Spieler 2 angegriffen & Spieler 4 getroffen!

- Auf dieses Feld hat Spieler 3 angegriffen & Spieler 1 getroffen!
- Auf dieses Feld hat Spieler 3 angegriffen & Spieler 2 getroffen!
- Auf dieses Feld hat Spieler 3 angegriffen & Spieler 4 getroffen!

- Auf dieses Feld hat Spieler 4 angegriffen & Spieler 1 getroffen!
- Auf dieses Feld hat Spieler 4 angegriffen & Spieler 2 getroffen!
- Auf dieses Feld hat Spieler 4 angegriffen & Spieler 3 getroffen!

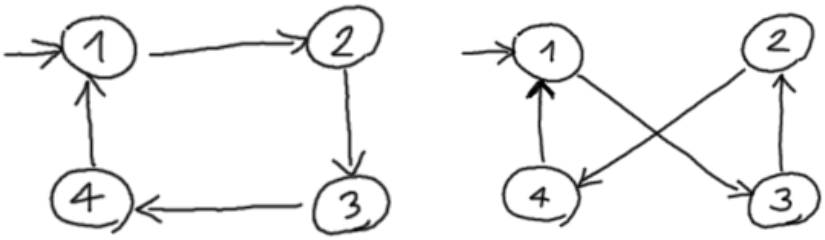
- Auf dieses Feld hat Spieler 3 geschossen, jedoch kein Treffer erzielt
- Auf dieses Feld hat Spieler 4 geschossen, jedoch kein Treffer erzielt

Somit existieren nun 7 + 14 Feld-Beschreibungen!

Überlegen Sie ob es sinnvoll ist weitere einzelne Variablen für die einzelnen Werte anzulegen oder ob man diese Infos nicht mittels eines Datentyps kapseln kann und somit exklusiv machen kann so dass Spielfelder gar keine anderen Werte mehr annehmen können außer den oben genannten & in Aufgabe 1 beschriebenen.

Die Logik der Methode „**SpielerWechseln“** aus Aufgabe 29 soll zusätzlich auf 4 Spieler angepasst werden. Entscheiden Sie sich für 1 der nachfolgenden Spielzug-Schemas bzgl. der Aktualisierung des aktuellen Spielers:

a) Spieler 1 → Spieler 2 → Spieler 3 → Spieler 4 → Spieler 1 …
b) Spieler 1 → Spieler 3 → Spieler 2 → Spieler 4 → Spieler 1 …



Überlegen Sie welchen Methoden der vorherigen Aufgaben Sie ändern müssen & ändern Sie diese Methoden entsprechend bei Bedarf ab!

Worauf hat die Erweiterung des Spieler-Systems Einfluss?

35) Erweitern Sie Aufgabe 34 indem jetzt Teams deaktiviert werden könnnen.

Nutzen Sie hierfür eine globale Variable „TeamsAktiviert“, welche entscheidet ob das beschriebene Team System der Aufgabe 34 aktiviert ist oder nicht.

Die globale Variable soll über eine Methode namens „**TeamsAktivieren**“ auf An und „**TeamsDeaktivieren**“ auf Aus gesetzt werden.
Alternativ kann auch eine Methode „**TeamsUmschalten**“ mit entsprechendem Parameter die Funktion der beiden Methoden übernehmen.

Bei Spielbeginn soll innerhalb der „Spielablauf“ Methode der Benutzer gefragt werden ob Teams aktiviert sein sollen oder nicht.

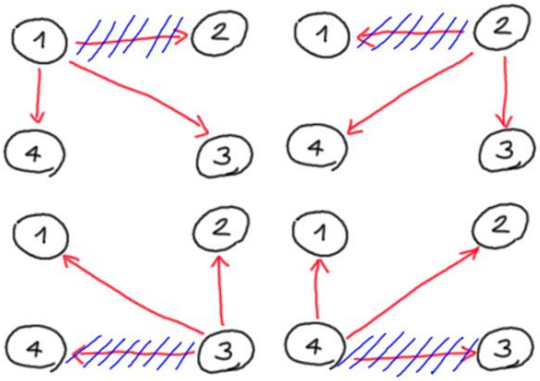
Schreiben Sie dafür eine „**MoechtenSieTeamsAktivieren**“ Hilfs-Methode, die den Benutzer „Ja“ oder „Nein“ eingeben lässt und entsprechend die Variable setzen kann. Fangen sie alle Ungültigen Benutzereingaben ab & wiederholen Sie solange die Eingabe bis einer der beiden Werte [„Ja“, „Nein“] eingelesen wurden.

Überlegen Sie weiterhin wo diese „TeamsAktiviert“ Einstellung eine Rolle spielt. Beim Aufruf welcher Methode können Sie diese Variable nun verwenden?

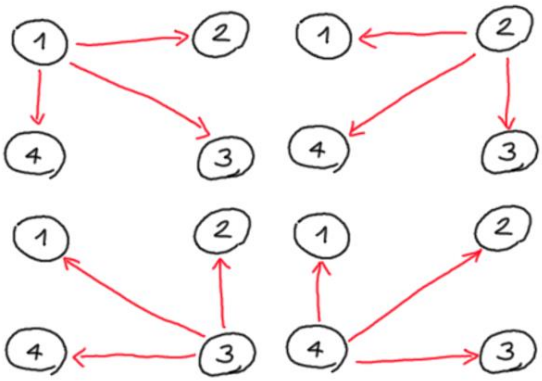
Überlegen Sie anhand der nachfolgenden Schaubilder auf der folgenden Seite welche der Werte aus der Aufgabe 34 bei Angriffen jetzt auftauchen können im Vergleich zum Teamfight.

Angriffs-Systeme:

a) Team Angriffs-System
„Team1“ vs „Team2“ (wer kann wen angreifen)



b) Jeder-Gegen-Jeden Angriffs-System
„Player1“ vs „Player2“ vs „Player3“ vs „Player4“ (wer kann wen angreifen)



Was ist dadurch jetzt möglich?

Müssen hierdurch Methode angepasst werden?
Welche Aufrufe von Methoden ändern sich jetzt?

<u>Ab der nächsten Seite werden die Themen Enumerationen & Structs eingeführt.</u>

EINFÜHRUNG VON ENUMS & STRUCTS

36) **(Erst komplett durchlesen!)**

Überladen Sie alle Methoden die als Parameter die Position X, Y nutzen indem diese Kopieren, Einfügen und dann vom Namen erweitern mit „**_vec2**“. Diese sollen anstelle der Position X, Y (2 Parameter) einen „**Vec2**“ als Parameter annehmen.
„**Vec2**“ bildet einen Vector2 (X, Y) aus der Mathematik ab.
Um einen Vec2 anzulegen benötigen wir „**struct**“s.

Diese dienen als Datenkapseln in der Programmierung und sind der Einstieg in die Welt der Objektorientieren-Programmierung.
Kopieren Sie folgenden Code in ihr Projekt oberhalb der class Program

```
public struct Vec2
{
        public int X;
        public int Y;
        public Vec2(int x, int y) //Konstruktor
        {
                X = x ;
                Y = y ;
        }
}
public class Program
{
        public static void Main(string[] args)
        {
                ErweiterterSpielablauf();
        }
}
```

Schreiben sie einen Methode namens „**Struct_Learning**“ die sich innerhalb der class Program befindet um das Verwenden von struct zu lernen.
Diese dient lediglich dem Experimentieren mit structs.

Kopieren den Code der nachfolgenden 3 Methoden & finden sie heraus was sie mit Structs machen können. Versuchen sie zuerst diesen Code zu verstehen bevor sie die Aufgabe lösen. Schreiben sie zusätzlich eigenen Code innerhalb der Struct_Learning um zu testen ob sie verstehen was sie machen können und rufen sie diese in der Main zum Testen auf.

```
public static void Struct_Learning()
{
        Vec2 vector = new Vec2(1,0); //erzeugen eines neuen Objekts vom Typ „Vec2“
        vector.X = 3;
        vector.Y = 5;
        Struct_CallbyValue(vector);
        //Frage: Welche Werte haben X und Y jetzt?
        Console.WriteLine(vector.X);
        Console.WriteLine(vector.Y);
        Struct_CallbyReference(ref vector);
        //Frage: Welche Werte haben X und Y jetzt?
        Console.WriteLine(vector.X);
        Console.WriteLine(vector.Y);
}
```

```
public static void Struct_CallbyValue(Vec2 vector)
{
        vector.X += 1;
        vector.Y += 1;
}
public static void Struct_CallbyReference(ref Vec2 vector)
{
        vector.X += 1;
        vector.Y += 1;
}
```

37) Überladen Sie weiterhin alle Methoden, welche die Schiffsgröße (1, 2 oder 3) als Parameter nutzen in dem sie nun diese Kopieren, Einfügen ansstelle dessen als Parameter eine Enumeration mit dem Namen „Schiffsgroesse“ angeben. Der Name der neuen Methode soll mit „**_enum**“ erweitert werden.
„Schiffsgroesse“ ist eine Enumeration die erstellt wird und (in C#) folgendermaßen aussieht

```
public enum Schiffsgroesse : int
{
        Klein = 1,
        Mittel = 2,
        Gross = 3
}
```

Zusätzlich soll es eine Methode namens „**Enum_Learning**“ geben, in welcher mit der Funktionalität der Enumeration herumgespielt werden soll

```
public static void Enum_Learning()
{
        //Eine Variable vom Typ „Schiffsgroesse“ anlegen und den Wert „Klein“ zuweisen
        Schiffsgroesse groesse = Schiffsgroesse.Klein;
        //Den Namen des Aktuellen Werts als String auslesen => „Klein“
        string name = groesse.ToString();
        //Den Wert hinter des aktuellen Werts auslesen => 1
        int aktuellerWert = (int) groesse;
        //Testen & Finden Sie heraus innerhalb dieser Methode wie Enums in if-elseif und
        switch-case Anweisungen funktionieren. Testen Sie dabei auch die
        Autovervollständigung in den jeweiligen Anweisungen!
}
```

Überlegen Sie hier welche Vorteile die Enumeration gegenüber einer normalen int Variable haben kann, welchen Vorteil diese Ihnen bei Anweisungen & Schleifen gibt und wie sie diese innerhalb ihres Projekts verwenden können.

Informieren sie sich weiterhin über diese in der Microsoft Dokumentation
https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/builtin-types/enum

38) Überlegen Sie weiterhin ob sie die Anzahl der Methoden in Aufgabe 14 reduzieren können mittels einer neuen Methode „**ErweiterteSchiffPlatzierung**“, die einen Vec2 (struct), eine Schiffsgroesse (enum) & eine Richtung (enum) annimt.
Richtung ist eine weitere Enumeration die für dieses Projekt wie folgt aussieht:

```
public enum Richtung : int
{
        Horizontal = 0,
        Vertikal = 1,
}
```

Passen Sie entsprechend Anweisungen/Abfragen an & Rufen sie die anderen Methoden aus der Aufgabe 14 bei Bedarf auf!

39) Erweitern Sie die Enumeration aus Aufgabe 38 mit dem Wert „Diagonal = 2“.
Das hat zur Folge, dass die Methoden in Aufgabe 14 wie folgt erweitert werden würden:

h) Schreiben Sie eine Methode namens „**SpawneGroßesSchiff_Diagonal**“, welches an angegebener Position (X, Y) sowie (X+1, Y+1) und (X-1, Y-1) für den angegebenen Spieler auf dessen Spielfeld ein Großes Schiff spawnt.
i) Schreiben Sie eine Methode namens „**SpawneMittleresSchiff_Diagonal**“, welches an angegebener Position (X, Y) sowie (X+1, Y+1) für den angegebenen Spieler auf dessen Spielfeld ein Mittleres Schiff spawnt.

Überlegen sie zusätzlich auch wie sie die „**ErweiterteSchiffPlatzierung**“ anpassen müssen!

40) Gehen Sie nochmal über die Aufgabenstellung der **Aufgabe 1 und Aufgabe 34** mit dem Hintergrundwissen wie Enumerationen funktionieren und bauen sie ein enum namens „FeldStatus“ für alle 21 Spielfeld-Beschreibungen aus Aufgabe 1 & 34.
Entferne alle const Variablen der beiden genannten Aufgaben aus dem Programm!
Überlegen Sie welche Methodenaufrufe jetzt nicht mehr funktionieren & wie sie diese anpassen müssen.
Hinweis: Parameter „ref <Datentyp> spieler“ – Werte aus Aufgabe 1 & 34

Welche möglichen Datentypen werden eliminiert um die Zustände eines Feldes[X,Y] darzustellen, wenn sie ab sofort eine Enumeration nutzen?

Denken Sie erneut über die nachfolgende Aussage aus der Aufgabe 34 nach:

„Überlegen Sie ob es sinnvoll ist weitere einzelne Variablen für die einzelnen Werte anzulegen oder ob man diese Infos nicht mittels eines Datentyps kapseln kann und somit exklusiv machen kann, so dass Spielfelder gar keine anderen Werte mehr annehmen können außer den oben genannten & in Aufgabe 1 beschriebenen.“

Jetzt sollten Sie wissen was in der Aussage gemeint war mit „ob man diese Infos mittels eines Datentyps kapseln kann“. (Das geht mittels Enumerationen)

41) **(FREIWILLIGE ZUSATZAUFGABE / ERWEITERUNGS-AUFGABE)**
Erweitern sie dieses Projekt mit eigenen Ideen!

Probieren Sie dabei das neu gelernte Wissen bezüglich Enumerations & Structs anzuwenden!

Informieren sie sich zusätzlich über alles was diese können und bauen sie ggf. weitere structs und Enums ins Spiel ein.
https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/builtin-types/struct
https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/builtin-types/enum

Hier einige Ideen
(Sie müssen keine davon übernehmen! Sie können auch eigenständig mit ihrem Code herumspielen):

a) Finden Sie Stellen wo Enumerationen & Struct eingesetzt werden könnten anstelle von Primitiven Datentypen
b) Erweitern sie die Spielfeld[X,Y] Information von einem int bzw. einem Enum zu einem Struct mit dem Namen „Feld“. Innerhalb von Feld soll eine Variable Schiff sein welche eine struct „Schiff“ referenziert. Überlegen Sie welche Infos in dem „Feld“ und „Schiff“ stehen.
c) Implementieren Sie eine Funktion „SchiffBewegen“ die es erlaubt ein Schiff eines angegebenen Spielers innerhalb einer Runde auf einem referenzierten Spielfeld in eine angegebene Himmelsrichtung (Norden, Süden, Osten, Westen, Nord-Ost, Süd-Ost, Süd-West, Nord-West) um 1 Feld zu bewegen. Denken Sie sich von einer Position (zB [3,3]) entsprechend die Ziel-Position aus anhand der genannten Himmelsrichtungen. Testen sie auch ob diese Ziel-Postion(en)[X,Y] im Spielfeld liegt.
d) Implementieren Sie ein Ereignis „Wind“ was zufällig während jeder Spielrunde mit einer Chance von 30% auftreten kann. Dieses soll 1 zufälliges spielfeld wählen & alle Schiffe (auch zerstörte Schiffe) wie in c) in eine Richtung bewegen. (Denken Sie an ungültige Werte).
e) Fügen Sie den Schiffen Haltbarkeit hinzu & Nutzen sie als Gegenstück entsprechend einen Schadenswert bei den Kanonenkugeln
(Vorraussetzung: Schiffe als struct → Spielfeld-Felder[X,Y] als struct darstellen)
f) Erlauben Sie eine variable (also änderbare) Anzahl von Kanonenkugeln die pro Zug abgefeuert werden können
g) Erweitern sie das Projekt mit einer variable „Trefferchance“ die eine Basis-Chance von 50% hat. Immer wenn eine Kanonenkugel abgefeuert wird zufällig ein Wert zwischen 0 und 50% auf diese hinzugerechnet. Nur wenn die Chance > 75% ist soll ein Schiff getroffen werden.

Seien Sie Kreativ!

Auf der nachfolgenden Frage finden Sie Wiederholungsfragen zu dieser Aufgabe & dem gesamten Umfang an Themen der mit diesem Aufgabenblatt verbunden ist.

Wiederholungsfragen (zum mündlich beantworten & eigenständigen recherchieren):

42) Welche Arten von Datentypen gibt es?
43) Welches ist der kleinste ganzzahlige Datentyp?
44) Welches ist der größte ganzzahlige Datentyp?
45) Wie viele Werte kann ein bool annehmen?
46) Woraus besteht ein string eigentlich?
47) Was ist eine globale Variable?
48) Was macht das **const** vor einer globalen Variable?

49) Was sind Arrays?
50) Was ist ein 1-Dimensionaler Array?
51) Was ist ein 2-Dimensionales Array?
52) Nennen Sie ein Beispiel aus der echten Welt um Arrays abzubilden.
53) Welche Datentypen können bei einem Array geneutzt werden?

54) Was sind Methoden?
55) Aus welchen Bestandteilen bestehen Methoden?
56) Erklären Sie am Beispiel „f(x) = 2x“ wie die Methode im Code (nur ganze Zahlen) aussehen könnte.
57) Was ist der Unterschied zwischen „Call by Value“ und „Call by Reference“?
58) Erklären Sie den Unterschied aus Frage 55) mittels eine Methode namens „int Addiere1(int x)“ und „void Addiere1(ref int x)“
59) Wann verwendet man das Schlüsselwort ref?
60) Wie schränken out und in das Schlüsselwort ref ein?
61) Bilden Sie eine mathematische Funktion „f(x,y,z) = ….“ welche 3 Zahlen miteinander addiert, wenn alle Zahlen größer 0 sind und genau dann das Ergebnis zurückgibt. Ansonsten soll die Methode -1 zurückgeben. Erklären sie wie diese im Code aussehen würde.
62) Wo kann man return verwenden? Nennen Sie die beiden Anwendungsfälle!

63) Welche Kontrollstrukturen gibt es?
64) Bilden sie die Logik des **switch-case** auf ein **If-elseif-else** ab!
65) Was sind Anweisungen und wie funktionieren diese ?
66) Was sind Schleifen und wie funktionieren diese?
67) Welche Arten von Schleifen gibt es?
68) Was ist der Unterschied zwischen einer Kopfgesteuerten & Fußgesteuerten Schleife?
69) Was ist eine Dauer-Schleife?
70) Was ist der Unterschied zwischen **return** und **break**;
71) Erklären Sie anhand einer while-Schleife wie die for-schleife funktioniert!
72) Was ist aufwendiger?
    Eine Methode mit 2 geschachtelten for-schleifen, die zusammen **100 Iterationen** (Durchgänge) machen um **1 Wert zu setzen**, **2x aufzurufen**
    ODER
    Eine Methode mit 2 geschachtelten for-Schleifen, die zusammen **100 Iterationen** (Durchgänge) machen um **2 Werte gleichzeitig zu setzen, nur 1x aufzurufen**?
    (Wie ist Gesamt-Anzahl der Iterationen?)

73) Welche Klasse wird verwendet um Zufallszahlen zu generieren?
74) In welchem Namespace befindet sich die Klasse zum generieren von Zufallszahlen?
75) Wie erstellt man eine **neue** Instanz der Zufallszahlen-Generator Klasse?
76) Welche Methode generiert ganzzahlige Zufallszahlen zwischen 2 Zahlen?
77) Ist der 2. Parameter der Methode zum generieren von Zufallszahlen inklusiv oder exklusiv? (Wird diese Zahl generiert oder nicht?)

78) Was ist ein **enum** / eine Enumeration?
79) Stellen Sie sich vor es gibt 3 Variablen x = 2, y = 3 und z = 4. Würden sie sich Variablen sparen wenn sie eine Enumeration verwenden und diese als Variable zahl anlegen?
80) Welchen Vorteil bietet eine Enumeration im Vergleich zu einer einfachen int-Variable?
81) Welche Datentypen können nicht bei einer Enumeration verwendet werden?
82) Wie sinnvoll sind enums in if-elseif bzw switch-case Anweisungen
83) Wenn Sie in Visual Studio ein switch-case für eine Variable eines Enums nutzen & auf die Autovervollständigung gehen (Glühbirne links bei der Zeilen-Nummer wo das switch(?) steht) & sagen „Fehlende Fälle hinzufügen“. Was passiert dann? Warum passiert das bei einer normalen int Variable nicht?.

84) Was sind **struct**?
85) Gehören **struct** genauo wie **int** zu den Value-Types oder Reference-Types?
    (Hinweisfrage: Werden diese wenn sie ohne **ref** in einer Methode angegebenen werden nur
    a) Als Kopie in die Methode gegeben
    b) Als Referenz in die Methode gegeben so dass es möglich ist Werte innerhalb des angegebenen structs zu verändern
86) Erklären sie wie ein **struct** namens „**Vec3**“ im Code aussehen könnte
    (Hinweis: Mathematik => Vector3 = Ein Vektor mit genau 3 Werten)
87) Haben Sie einen Vorteil durch „**Vec2**“ anstelle eines Arrays **int**[2] { X, Y }?

Haben Sie alle Fragen verstanden?

Wenn ja, dann Gratulation die Grundlagen der Themen Datentypen, Arrays, Methoden, Kontrollanweisungen, Struct & Enums verstanden.

Nutzen Sie die Fragen 42) bis 87) immer wieder um sich zu testen!

Nachfolgende Aufgaben nach diesem Aufgabenblatt werden diese Grundlagen einbinden!

Ziel im nachfolgenden Aufgabenblatt ist es in die Grundlagen der Objekt-Orientierten-Programmierung tiefer einzutauchen

Kommentar:

Ich hoffe das Lösen hat Spaß gemacht!
Es mag zwar ein einfaches Spielprinzip sein aber mit genug Ideen, Kreativität & Lust kann man selbst dieses kleine Projekt erweitern & in diversen Bereichen ganz individuell gestalten.
Die Beispiele die in Aufgabe 40 genannt werden für Erweiterungen sind dabei simple Ideen. Sie machen den Code zwar komplexer an manchen Stellen aber genau dies macht ein Spiel vielfältig. Das Spiel kann variieren und wird durch simple Sachen erweitert um den Spielspaß auf Dauer zu halten.

Man braucht keine Game Engine um ein Spiel zu implementieren.
Natürlich wäre

- ein Meer,
- 3D Modelle von Schiffen,
- Musik,
- Soundeffekte
- UI
- Grafiken
- & fliegende Kanonenkugeln (Physics)

toll. Aber ein Spiel ist modular.
Das heißt die Game-Logik funktioniert alleine für sich und kann durch Sachen wie eine UI und allem oben genannten erweitert werden. Das ist das schöne an Spieleentwicklung. Es mag solche Konzepte schon in 1000 anderen Spielen geben. Aber die eigene Kreativität, das eigene Auge & Design ist es am Ende was ein Spiel / Produkt von anderen unterscheidet.

Schiffe versenken – Created by http://twitch.tv/mostwanted38/ - https://discord.gg/WMFucAD - 21.07.2020