

ESCAPE FROM MARS: VARIABLEN

Mission 01: Start the Count

PEETIE nähern sich bewaffnete Drohnen! Um **PEETIE** effektiv steuern zu können, benötigen wir mehr Informationen über seinen Status (z.B. Munition, Lebensenergie, etc.).

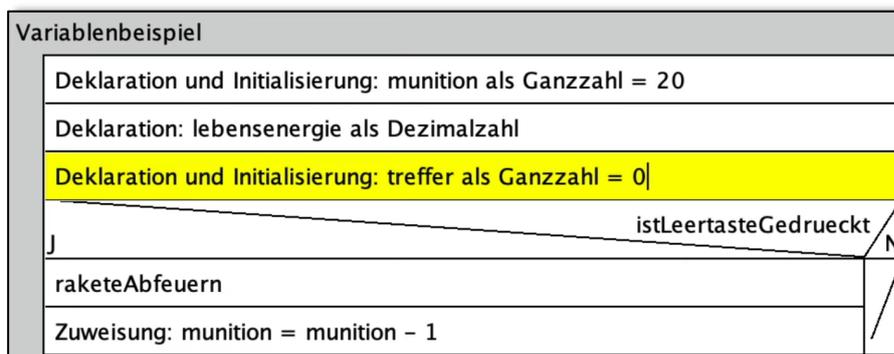


Informationen speichern wir in Variablen!

Weil die Art der Daten aber sehr unterschiedlich ist (z.B. Zahlen oder Texte), gibt es verschiedene Arten von Variablen:

Bezeichnung	in Greenfoot/Java	Beispiele
Ganzzahl	int	3 5 31921 -991
Dezimalzahl	double	3.141 -3152.00 317172.2
Wahrheitswert	boolean	true false
Zeichenkette	String	"Dr. Andy White" "u" "جيد يوم."

Beispiel: Struktogramm zum Speichern der Munition



Aufgabe

Job für Dr. White

Dein Job: Deklariere Variablen (Ganzzahl = int) für **munition**, **lebensenergie**, **peetiegetroffen** und **zerstoerteDrohnen** an. Gib allen Variablen einen Initialwert.

Zähle die entsprechenden Werte in der run-Funktion und gib alles (ebenfalls dort) aus mit textAusgeben.

Neue Befehle:

- drohneWurdeZerstoert

- peetieWurdeGetroffen

- textAusgeben("Munition: " + munition, 100, 100)

---- In den Anführungszeichen steht, was immer du willst. Nach dem Plus kommt die Variable, deren Wert du ausgeben willst.

---- 100, 100 - das ist die x- und y-Position

Mission 2: Treffer zählen

Dr. White muss wissen, wie viele Drohnen Peetie bereits abgeschossen hat. Das kann nicht so schwer zu programmieren sein. Dummerweise hat Dragan gleich drei Struktogramme geschickt, und wir müssen herausfinden, welches das korrekte Struktogramm ist.

Aufgabe

EFM

Job für Dr. White
 Dragan hat drei Struktogramme geschickt, mit denen die Anzahl der zerstörten Drohnen ermittelt werden soll.

Analysiere diese Struktogramme. Wo sind Fehler? Welches ist am besten gelungen?

Erstelle ein Struktogramm, das funktioniert. Teste es, indem du es implementierst.

E4M02 - Drohnenzerstörungen zählen VERSION 1

Deklaration und Initialisierung: zerstoerteDrohnen als Ganzzahl = 0	
J	drohneWurdeZerstoert? / N
Zuweisung: zerstoerteDrohnen + 1	
textAusgeben("Zerstörte Drohnen: " + zerstoerteDrohnen, 750, 50)	

E4M02 - Drohnenzerstörungen zählen VERSION 2

Deklaration und Initialisierung: zerstoerteDrohnen als Ganzzahl = 0	
J	drohneWurdeZerstoert? / N
textAusgeben("Zerstörte Drohnen: ", 750, 50)	

E4M02 - Drohnenzerstörungen zählen VERSION 3

Deklaration und Initialisierung: zerstoerteDrohnen als Ganzzahl = 0	
drohneWurdeZerstoert?	
Zuweisung: zerstoerteDrohnen = zerstoerteDrohnen + 1	
textAusgeben("Zerstörte Drohnen: " + zerstoerteDrohnen, 750, 50)	

Mission 03: Kompliziertes Programm

Dragan hat ein Struktogramm geschickt, das aber einigermaßen kompliziert zu sein scheint. Egal - spielen wir es **PEETIE** auf!

Aufgabe

EFM
 escape-from-mars.de

Job für Dr. White
 Programmiere das Struktogramm in Peeties Speicher und teste, ob es funktioniert!

Tipp: Wahrheitswerte heißen in Java "boolean".

E4M03 - Munitionsstand kritisch

Deklaration und Initialisierung: munition als Ganzzahl = 30	
Deklaration: munitionsstandIstKritisch als Wahrheitswert	
J	Leertaste gedrückt? / N
raketeAbfeuern	
Zuweisung: munition = munition - 1	
J	munition > 7 / N
Zuweisung: munitionsstandIstKritisch = false	
Zuweisung: munitionsstandIstKritisch = true	
munitionstandIstKritisch	
J	munition == 30 / N
Ausgabe: "Munition kritisch!"	
Ausgabe: "Genügend Munition"	
J	munition == 15 / N
Ausgabe: "Genau die Hälfte verbraucht!"	

Mission 04: Verschachtelt

Das letzte Programm hätte viel einfacher funktioniert, wenn man es mit verschachtelten Verzweigungen gebaut hätte! Zum Glück hat Dragan schon eine Lösung parat.

Aufgabe

Job für Dr. White

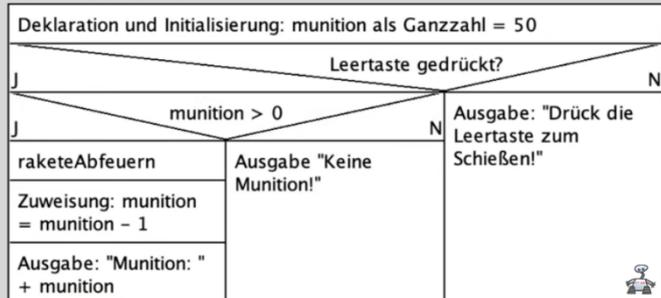
Implementiere das Struktogramm bei Peetie und prüfe seine Funktionsfähigkeit.

Du musst dazu eine Verzweigung IN den Zweig einer Verzweigung bauen, so etwa: `if (leertasteGedueckt())`

```
if (munition > 0)
```

```
  tue etwas
```

E4M04 Munitionsprüfung verschachtelt



Mission 05: &&

Dragan hat Dr. White das falsche Struktogramm geschickt! In Wahrheit soll das Programm mit einer durch UND (&&) verknüpften Bedingung funktionieren!

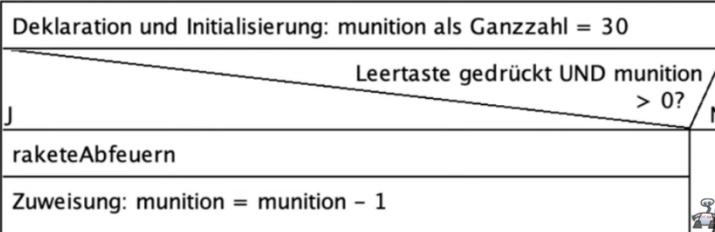
Aufgabe

Job für Dr. White

Implementiere das Struktogramm bei Peetie und teste, ob es funktioniert.

Die Verknüpfung zweier Bedingungen mit UND machst du in Java/Greenfoot mit `&&`

E4M05 Munitionsprüfung UND



Mission 06: Maustaste und -Feuer frei!

Neben der Leertaste soll PEETIE jetzt auch mit der Maustaste schießen können, um noch effizienter gegen die SyberDrone-Killer vorgehen zu können.

Aufgabe

EFM
escape-from-mars.de

Job für Dr. White

Programmiere bei Peetie korrekte Raketenabschüsse:

1. Wenn Leertaste gedrückt ODER Maustaste gedrückt, soll gefeuert werden (**neuer Befehl: maustasteGedruickt**)
2. Es kann nur gefeuert werden, wenn die Munition größer 0 ist.

Tipp 1: Verwende UND (&&) bzw. ODER (||). Das ODER-Zeichen sind zwei "Pipes" (macOS: alt/opt + 7, Windows: Alt Gr + >).

Tipp 2: Du hast zwei Möglichkeiten, das Problem zu lösen. Entweder verwendest du zusätzlich zum ODER eine verschachtelte Verzweigung, oder du machst alles auf einmal (also etwas wie:

```
if((leertaste || maustaste) && munition > 0) ...
```

Mission 07: Felsen, Schritte, Batterien

PEETIE hat alle Drohnen erledigt! Aber er zeigt heftige Abnutzungserscheinungen, man muss unbedingt sein Wartungsprogramm laufen lassen. Das geht aber erst, wenn die notwendigen Statistiken vorliegen.

Aufgabe

Job für Dr. White

Entwickle ein Struktogramm, damit Peetie mitzählen kann:

- Wie viele Schritte gemacht?
- Wie viele Batterien eingesammelt?
- Wie viele Felsen umrundet?

In Peetie ist schon ein Teil des Programms angefangen, diesen Programmcode baust du ebenfalls ins Struktogramm ein.

Tipp: Die Welt ist 18 Felder breit.

Neue Befehle: batterieWirdBeruehrt, batterieAufnehmen

Wie viele Schritte?

Wie viele Batterien?

Wie viele Felsumrundungen?