Kurzbeschreibung der Gamepackage-Klassen:

Interface:
Deklariert alle moeglichen GUI-Variablen (Labels, Buttons, etc) und initialisiert diese mit Hilfe von Menues.InitComponents. Befeullt dann durch die Menues-Klassen, je nach Klick, das Panel neu.
Legt den GameMode fest: Multiplayer, KI, Offline
Implementiert einen Keylistener, und ueberschreibt zT Methoden.

Game:
Startet eine BomberDroid-Runde und je nach Einstellung den KI-Thread.

Field:
Alle Methoden zum Neuzeichnen des Spielfeldes. Wird durch den repaintThread alle 30ms aufgerufen.

Bomb:
Alle Methoden fuer die Entwicklung des Bombenthreads und GameOver-Abfragen.
Speichert das statische bombs[]-Array.

Carl:
Bombenthread. Bekommt eine Bombe uebergeben und handelt sie entsprechend ab.
Springt nach 3 Sekunden in Bomb.detonate(Bomb bomb), wartet dann noch einmal eine Sekunde und beendet die Detonation.

Init:
Hier befinden sich alle moeglichen Methoden zur Initialisierung aller vorhandenen, spieltechnisch wichtigen Variablen.
reset()-Methode setzt alle spieltechnisch wichtigen Variablen nach jeder Runde oder bei Klick auf Zurueck zum Hauptmenue wieder auf ihre Ausgangswerte.

Input:
Legt den Spielerobjekt-Parameter "ctrl" fest.

KI:
Alle moeglichen Methoden zur KI.
Sie entscheidet sich nach dem Zufallsprinzip fuer eine Richtung, reagiert auf Spieler und Kisten in ihrem Bombenradius und rennt bei Gefahr entsprechend weg.

Load:
Startet einen JFileChooser und wartet auf eine Textdatei aus dem Maps-Ordner.
Laedt danach den Content mit Hilfe der zwischenzeitlich initialisierten Informations-Arrays der Init-Klasse.

LockControl:
Entfernt die Kontrollmoeglichkeit eines Spielers fuer eine kurze Zeit, damit der Spieler - rein visuell - auf dem Spielfeld durch laengeres Druecken einer Taste nicht ueber frei begehbare Felder springt.

MenuInput:
Setzt die im Controls-Menue festgelegten Steuerungs-KeyIds.

Movement:
Bewegt einen Spieler, legt / startet eine Bombe und beinhaltet Kollisionsabfragen.

Paul:
KI-Thread. Ruft alle 750ms die KI-Methoden auf.

Player:
Initialisiert die Spieler-Objekte.

RepaintThread:
Zeichnet das Spielfeld alle 30ms neu.

Save:
Speichert das Spielfeld und wichtige Player- und Bombenvariablen nach einem bestimmten Schema in einer Textdatei.

Sound:
Schlichte Soundklasse. Wird aufgerufen und gibt danach einmalig Sound aus.

Das Bomberdroid Server-Klient Prinzip:

Der Server wird gestartet und initiiert die Felder fieldNumbers und powerUps. Er wartet darauf, dass sich 2 Klienten mit ihm verbinden. Sobald dies geschehen ist, versendet er fieldNumbers und powerUps an den Klient der sich als erstes mit ihm verbunden hatte und wartet darauf, dass dieser ihm ein bearbeitetes fieldNumbers, die Spielerinformation und die Bombeninformationen des Klienten übermittelt. Der Klient bearbeitet speichert das erhaltene fieldNumbers ab und setzt seinen Spieler darauf und verschickt die Daten auf die der Server wartet.
Nachdem der Server die erforderlichen Daten vom ersten Klienten erhalten hat, verschickt er die veränderte fieldNumbers und powerUps an den zweiten Klienten, welcher die fieldNumbers ebenfalls editiert indem er seinen Spieler drauf setzt. Hat der zweite Klient dies auch erledigt sendet er die gleichen Daten, wie auch Klient 1 an den Server.

Hat der Server auch diese erhalten, sendet der Server die nun zusammen getragenen Daten nochmal an beide Klienten. Die Klienten starten dann auf der Basis dieser Daten ein
neues Spiel. Ab diesem Zeitpunkt findet ein dauerhafter Austausch von Daten zwischen den Klienten und dem Server statt. Jeder Klient sendet hierbei, bei jeder Aktion immer nur seine Spielerinformationen und Bombeninformationen an den Server, welcher diese Daten dann verteilt.

Wichtigsten Variablen/Objekte/Programmstellen

Variablen/Objekte
Game.Field.fieldNumbers[][]
Game.Init.Player1
Game.Init.Player2
Game.Bomb.bombs[]

Programmstellen und beinhaltende Methoden
Game.Movement.getMovement(Player player);
Game.Carl: Bombenthread