Ausnahmenbehandlung

16.12.2019

Fabian Drach

Fehlerquellen

Ein Programm kann aus unterschiedlichen Gründen unerwünschtes Verhalten zeigen

- Logische Fehler beim Entwurf & Umsetzung (Programm macht nicht das, was es soll)
- Programm bricht wegen vermeidbaren Fehler ab (oder lässt sich nicht kompilieren)
- (z.B. Division durch 0, falscher Arrayzugriff, ...)
- Falscher Umgang mit unerwarteten Situationen
- (z.B. Falsche Benutzereingabe, ...)

Robuste Programme

Ziel ist es, **robuste Programme** zu schreiben

Ein Programm heißt **robust**, falls es mit Fehlern umgehen kann, und nicht sofort abbricht

Logische Fehler

Programm wurde geschrieben und kompiliert, jedoch wird bei der Ausführung ersichtlich, dass das Programm nicht seinen Zweck erfüllt

Mögliche Schritte zur Lösung:

- Code genauer studieren
- Code jemand anderes (auch Hund oder Gummiente) erklären
- Code in andere Hände geben
- **Debugger**

Debugger

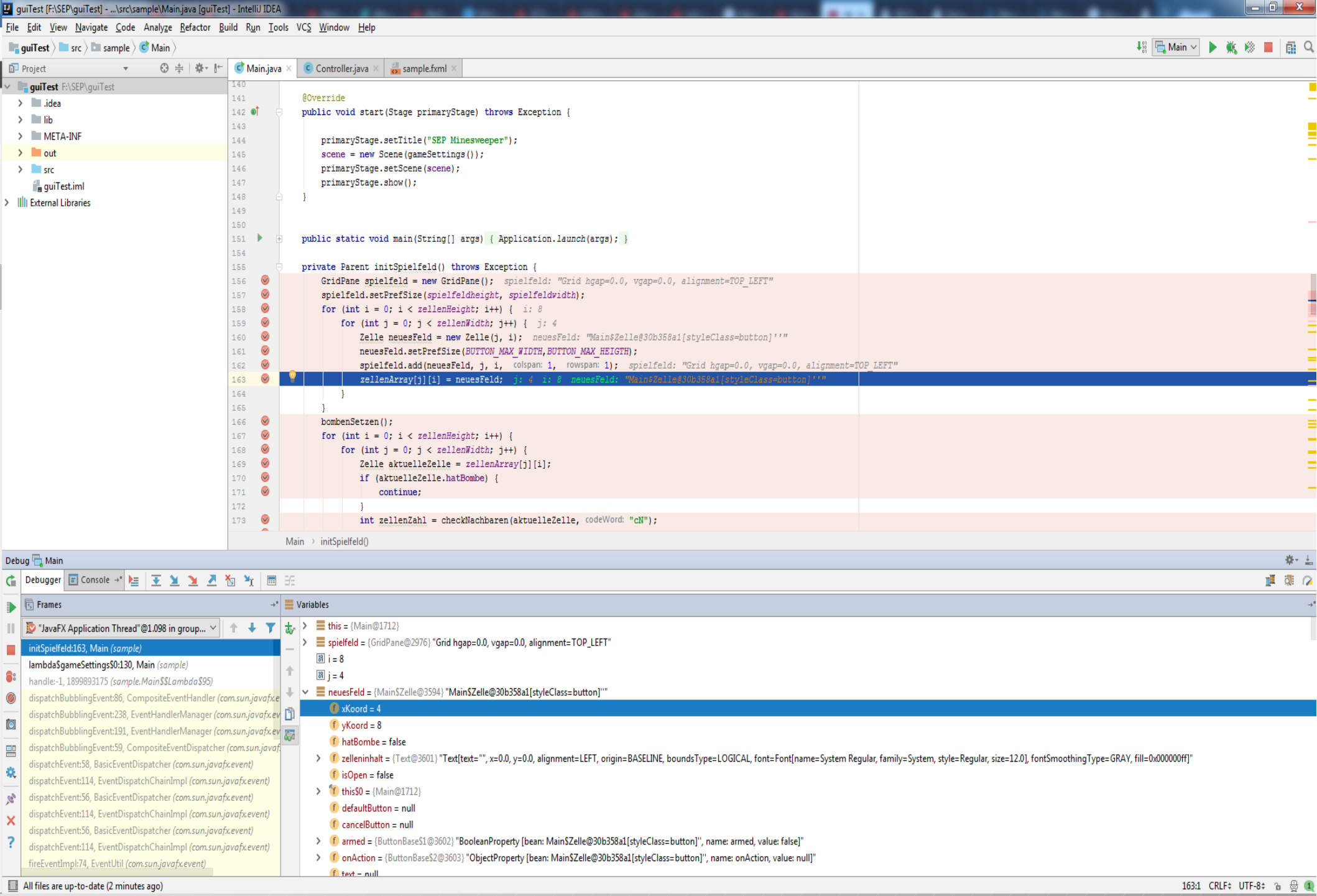
Debugger sind Funktionalitäten, die in jeder gängigen IDE vorhanden sind

(wird oft mit einen Käfer dargestellt)

Debugger ermöglichen es, **Breakpoints** zu setzen

Das Programm wird am Zeitpunkt des Breakpoints angehalten, und der Speicher kann untersucht werden

Zudem werden Eigenschaften der existierenden Objekte angezeigt


guiTest [F:\SEP\guiTest] - ...\src\sample\Main.java [guiTest] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
guiTest src sample Main
Main
Project
guiTest F:\SEP\guiTest
.idea
lib
META-INF
out
src
guiTest.iml
External Libraries
Main.java
Controller.java
sample.fxml
@Override
public void start(Stage primaryStage) throws Exception {
primaryStage.setTitle("SEP Minesweeper");
scene = new Scene(gameSettings());
primaryStage.setScene(scene);
primaryStage.show();
}
public static void main(String[] args) { Application.launch(args); }
private Parent initSpielfeld() throws Exception {
GridPane spielfeld = new GridPane(); spielfeld: "Grid hgap=0.0, vgap=0.0, alignment=TOP_LEFT"
spielfeld.setPrefSize(spielfeldheight, spielfeldwidth);
for (int i = 0; i < zellenHeight; i++) { i: 8
for (int j = 0; j < zellenWidth; j++) { j: 4
Zelle neuesFeld = new Zelle(j, i); neuesFeld: "Main$Zelle@30b358a1[styleClass=button]''"
neuesFeld.setPrefSize(BUTTON_MAX_WIDTH,BUTTON_MAX_HEIGTH);
spielfeld.add(neuesFeld, j, i, colspan: 1, rowspan: 1); spielfeld: "Grid hgap=0.0, vgap=0.0, alignment=TOP_LEFT"
zellenArray[j][i] = neuesFeld; j: 4 i: 8 neuesFeld: "Main$Zelle@30b358a1[styleClass=button]''"
}
}
bombenSetzen();
for (int i = 0; i < zellenHeight; i++) {
for (int j = 0; j < zellenWidth; j++) {
Zelle aktuelleZelle = zellenArray[j][i];
if (aktuelleZelle.hatBombe) {
continue;
}
int zellenZahl = checkNachbaren(aktuelleZelle, codeWord: "cN");
Main › initSpielfeld()
Debug Main
Debugger Console
Frames
"JavaFX Application Thread"@1.098 in group...
initSpielfeld:163, Main (sample)
lambda$gameSettings$0:130, Main (sample)
handle:-1, 1899893175 (sample.Main$$Lambda$95)
dispatchBubblingEvent:86, CompositeEventHandler (com.sun.javafx.e
dispatchBubblingEvent:238, EventHandlerManager (com.sun.javafx.ev
dispatchBubblingEvent:191, EventHandlerManager (com.sun.javafx.ev
dispatchBubblingEvent:59, CompositeEventDispatcher (com.sun.javaf
dispatchEvent:58, BasicEventDispatcher (com.sun.javafx.event)
dispatchEvent:114, EventDispatchChainImpl (com.sun.javafx.event)
dispatchEvent:56, BasicEventDispatcher (com.sun.javafx.event)
dispatchEvent:114, EventDispatchChainImpl (com.sun.javafx.event)
dispatchEvent:56, BasicEventDispatcher (com.sun.javafx.event)
dispatchEvent:114, EventDispatchChainImpl (com.sun.javafx.event)
fireEventImpl:74, EventUtil (com.sun.javafx.event)
Variables
this = {Main@1712}
spielfeld = {GridPane@2976} "Grid hgap=0.0, vgap=0.0, alignment=TOP_LEFT"
i = 8
j = 4
neuesFeld = {Main$Zelle@3594} "Main$Zelle@30b358a1[styleClass=button]''"
xKoord = 4
yKoord = 8
hatBombe = false
zelleninhalt = {Text@3601} "Text[text='', x=0.0, y=0.0, alignment=LEFT, origin=BASELINE, boundsType=LOGICAL, font=Font[name=System Regular, family=System, style=Regular, size=12.0], fontSmoothingType=GRAY, fill=0x000000ff]"
isOpen = false
this$0 = {Main@1712}
defaultButton = null
cancelButton = null
armed = {ButtonBase$1@3602} "BooleanProperty [bean: Main$Zelle@30b358a1[styleClass=button]'', name: armed, value: false]"
onAction = {ButtonBase$2@3603} "ObjectProperty [bean: Main$Zelle@30b358a1[styleClass=button]'', name: onAction, value: null]"
text = null
All files are up-to-date (2 minutes ago)
163:1 CRLF UTF-8

Fehler und Ausnahmen

Fehler sind schwerwiegende Probleme und sollten nie behandelt werden (Instanzen der Klasse Error)

Ausnahmen können abgefangen werden;

Man unterscheidet weiter zwischen:

- Unchecked Exceptions (Ausnahmen, die nicht zwingend behandelt werden müssen, Instanzen der Klasse RuntimeException)
- Checked Exceptions (Ausnahmen, die behandelt werden müssen, alle anderen Instanzen von Exception)

Umsetzung

In Java gibt es zwei Möglichkeiten, Ausnahmen zu definieren:

try … catch … finally Block

```
try {
    int i = scanner.nextInt();
}
catch (InputMismatchException ex) { //...
```

throw

```
if (i<10) { throw new
IllegalArgumentException("Falsche Zahl");}
```

Sämtliche Ausnahmen in Java sind wiederum Objekte ihrer Klasse

Dank Vererbung können Sie eigene Fehler und deren Behandlung integrieren

z.B. mit Oberklasse Exception

Fragen?

Anregungen?

Bemerkungen?