Debugging

Arten von Fehlern

Compiler Fehler

```
for (int i=1;i<=10;i++ ) {
   System.out.println("Compiler error")
```

; is missing

Laufzeit Fehler

```
for (int i=1;i>=10; i++ ) {
   System.out.println("Runtime error" +i/0);
```

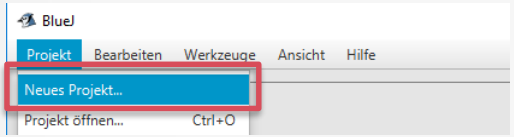
Exception in thread "main" java.lang.ArithmeticExpression / by zero at test.main(test.java:4)

Logische Fehler

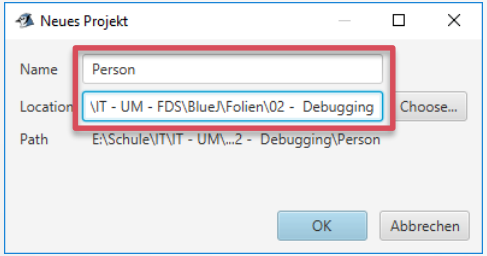
```
for (int i=1;i>=10; i++ ) {
   System.out.println("Logical error")
```

Erstellen eines neuen Projektes

1. Menüpunkt „Neues Projekt“ wählen

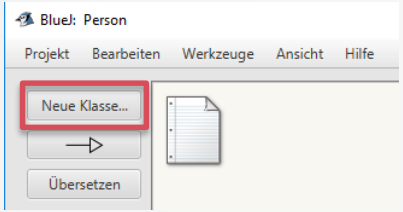


2. Name(ohne „“) und neuen Pfad(der darf NICHT schreibgeschützt sein) eingeben:

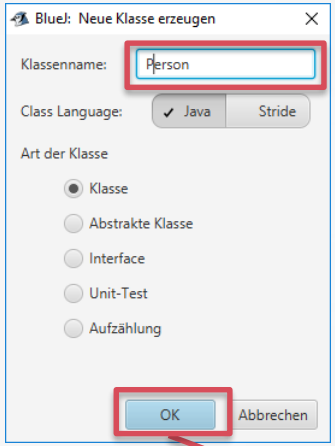


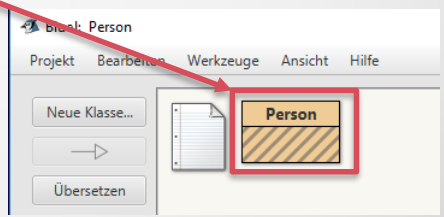
Erstellen einer neuen Klasse I

1.



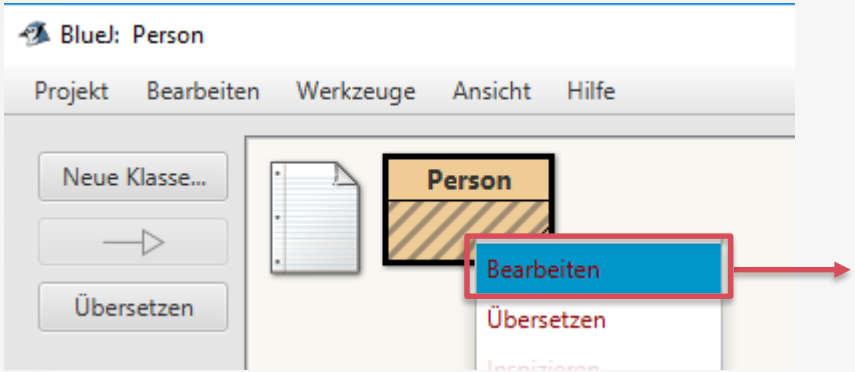
2. Namen der Klasse eingeben





Erstellen einer neuen Klasse II

3. Rechte Maustaste betätigen:



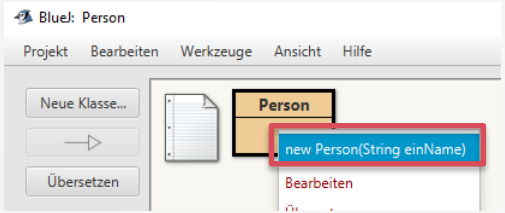
4. Quelltext rechts eingeben:

```
class Person {
    private String name;
    private int alter;
    public Person( String einName) {
       name = einName;
    }
    public void setAlter(int lebensjahre) {
       alter = lebensjahre;
    }
    public String getName() {
       return name;
    }
    public int getAlter() {
       return alter;
    }
}
```

Erstellen eines neuen Objektes

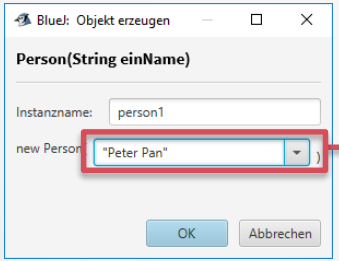
1.

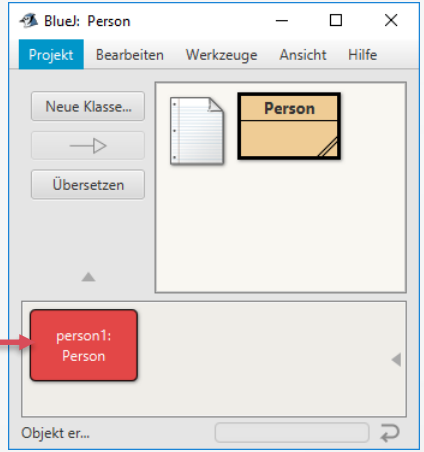
Rechte Maustaste betätigen:



2.

Namen der Person eingeben: Mit „“-Zeichen, da Datentyp String





Erläuterung der Klasse – Rahmen und private Variablen

Schlüsselwort für Klassendefinition

Name der Klasse(Großschreibung)

```
class Person {
    private String name;
    private int alter;
    ……….
}
```

Private Variablen(von außen unsichtbar)

Erläuterung der Klasse - Konstruktor

Konstruktor: Wird beim Erstellen eines Objektes aufgerufen

Beim Erstellen eines Objektes wird 1 Parameter vom Typ String übergeben

```
class Person {
    ……….
    public Person( String einName) {
      name = einName;
    }
     ……….
}
```

Speichern des eingegebenen Wertes in der privaten Variable

Erläuterung der Klasse - Konstruktor

Zu jeder privaten Variable gehört ein Getter und Setter

Beim Erstellen eines Objektes wird 1 Parameter vom Typ String übergeben

```
.....
private int alter;
.....       }
public void setAlter(int lebensjahre) {
   alter = lebensjahre;
}

public int getAlter() {
  return alter;
}
}
```

Kein Rückgabewert!
Aufruf:
peter.setAlter(3);

Immer EIN Parameter; muss vom gleichen Datentype wie alter sein!

Rückgabewert „int“:
Der gleiche Datentyp wie die Variable „alter“

Gibt „alter“ zurück:
Aufruf:
x=peter.getAlter();

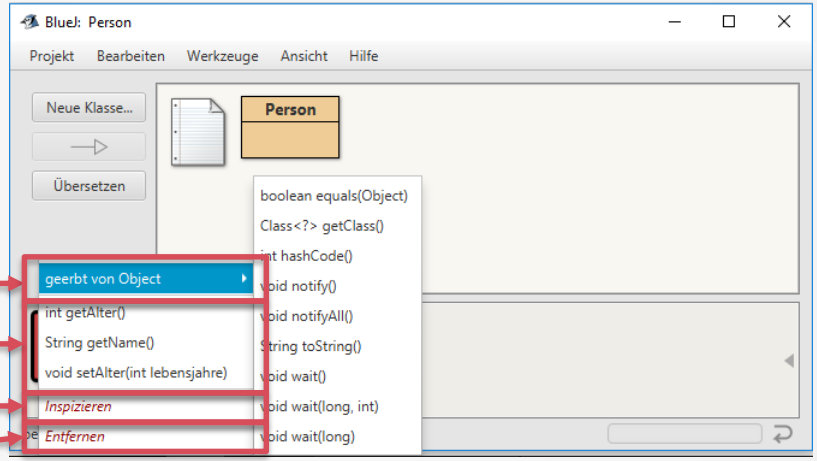
Kontextmenü (rechte Maustaste)

Methoden der Elternklasse Object

Methoden der Klasse

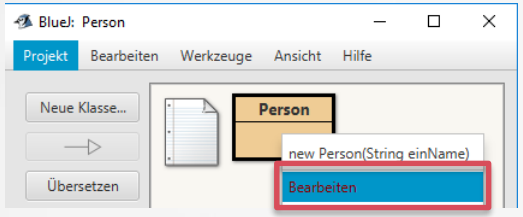
Eigenschaften der Klasse

Löschen der Klasse



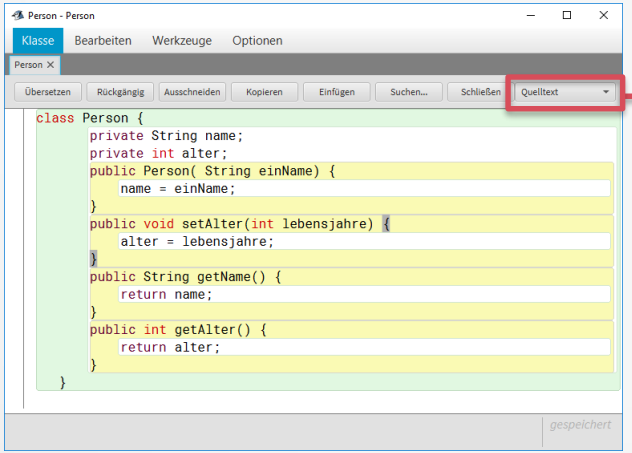
Dokumentation

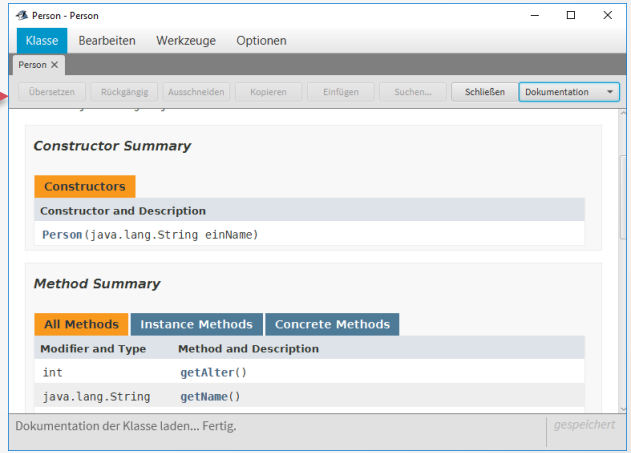
1.



Im Kombinationsfeld
Dokumentation wählen

2.

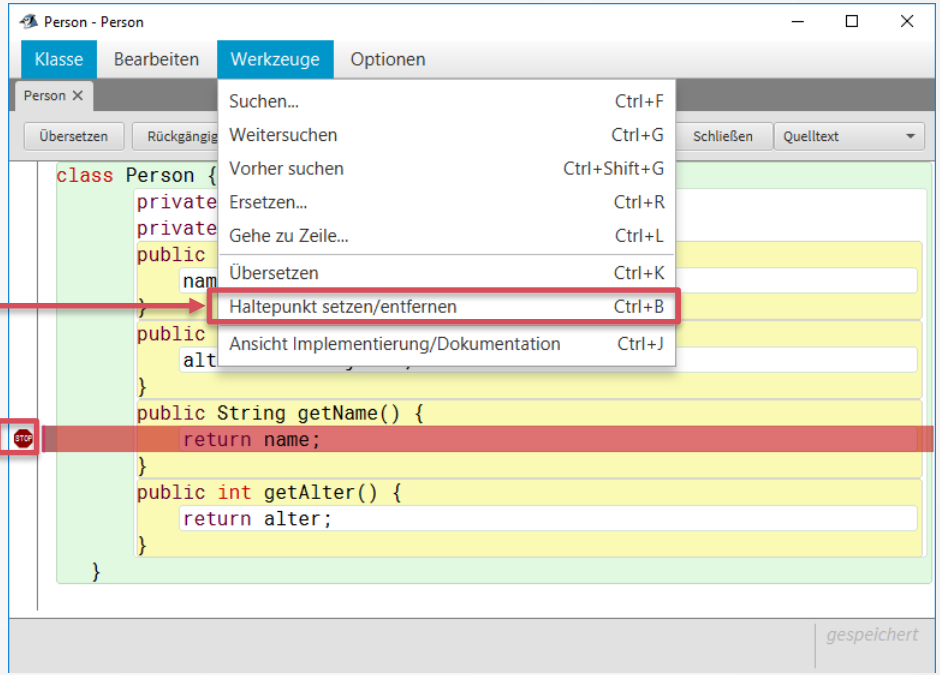




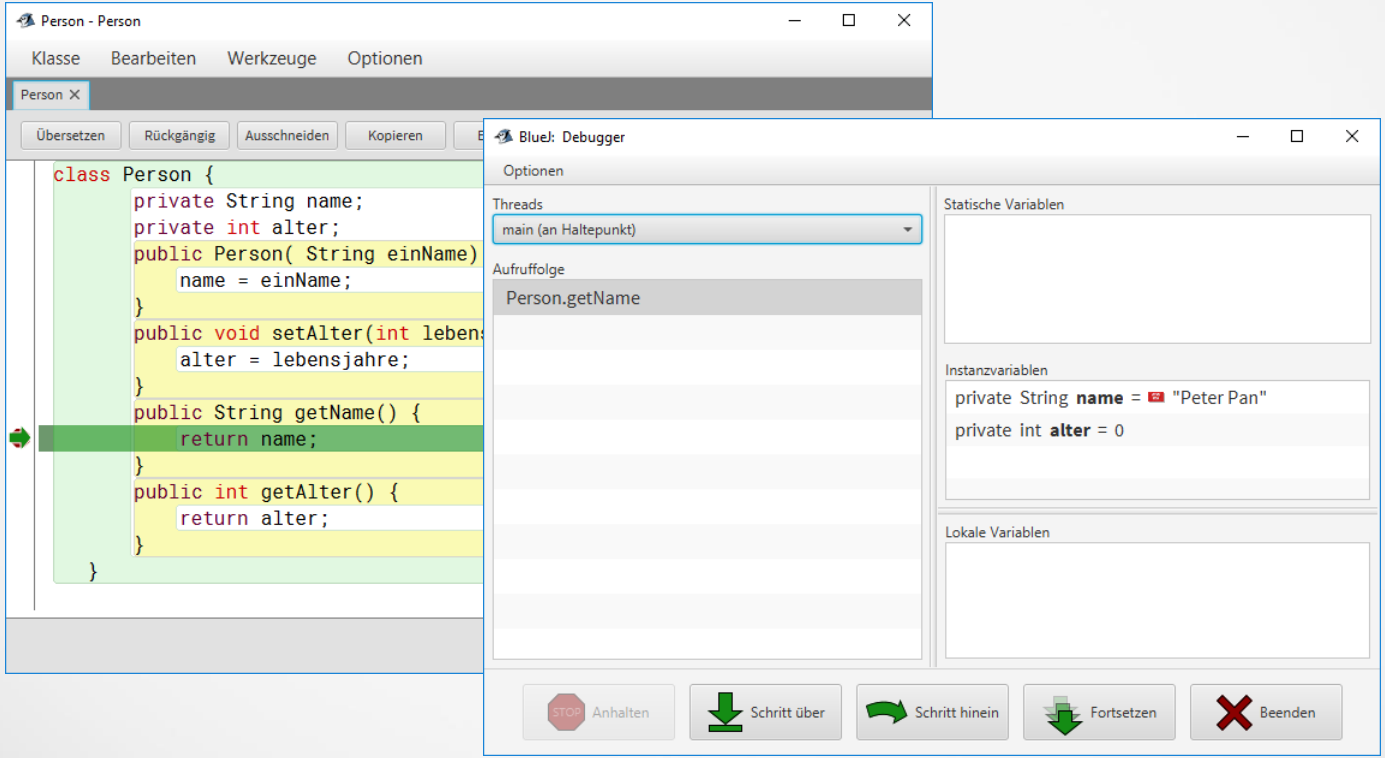
Debugging – Haltepunkte setzen

Klick auf Zeile
oder Menüpunkt

Neues Objekt;
Aufruf von
getName();



Debugging – Haltepunkte verwenden



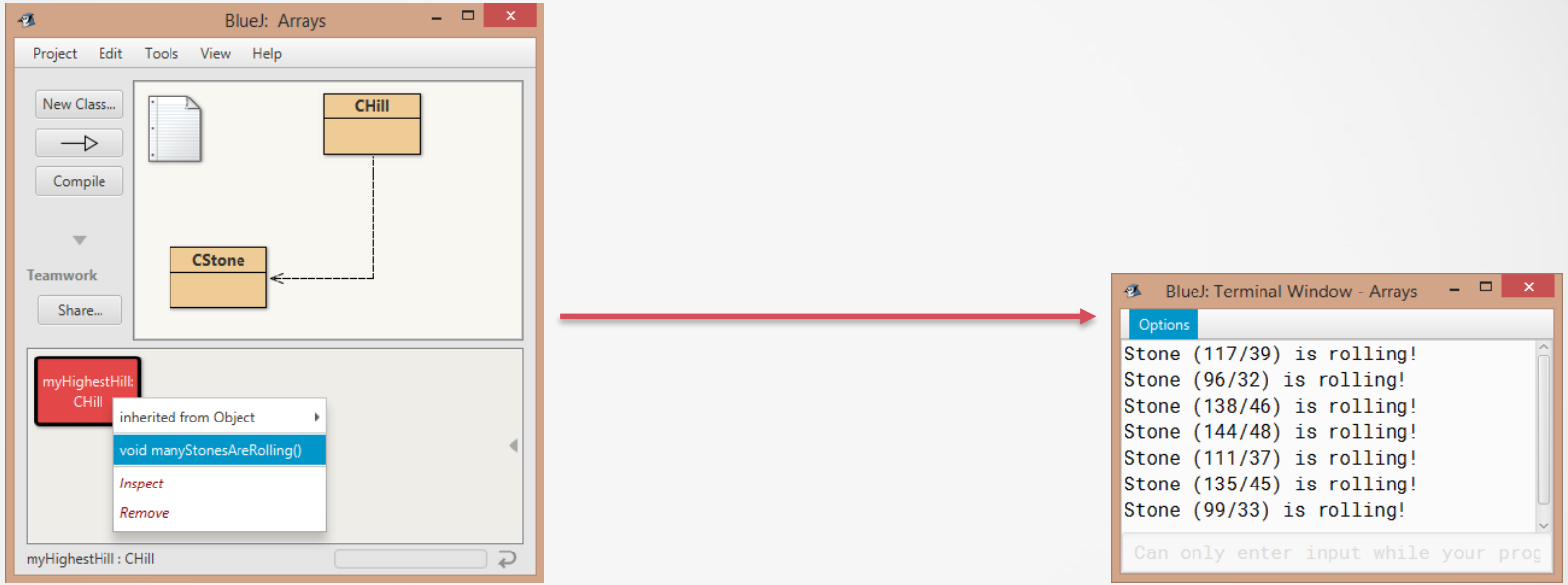
Debugging mit Arrays I

```
public class CStone{
  public int size, weight;
  public CStone(){}
  public void roll(){
    System.out.println("Stone ("+weight+"/"+size+") is rolling!");
  }
}
```

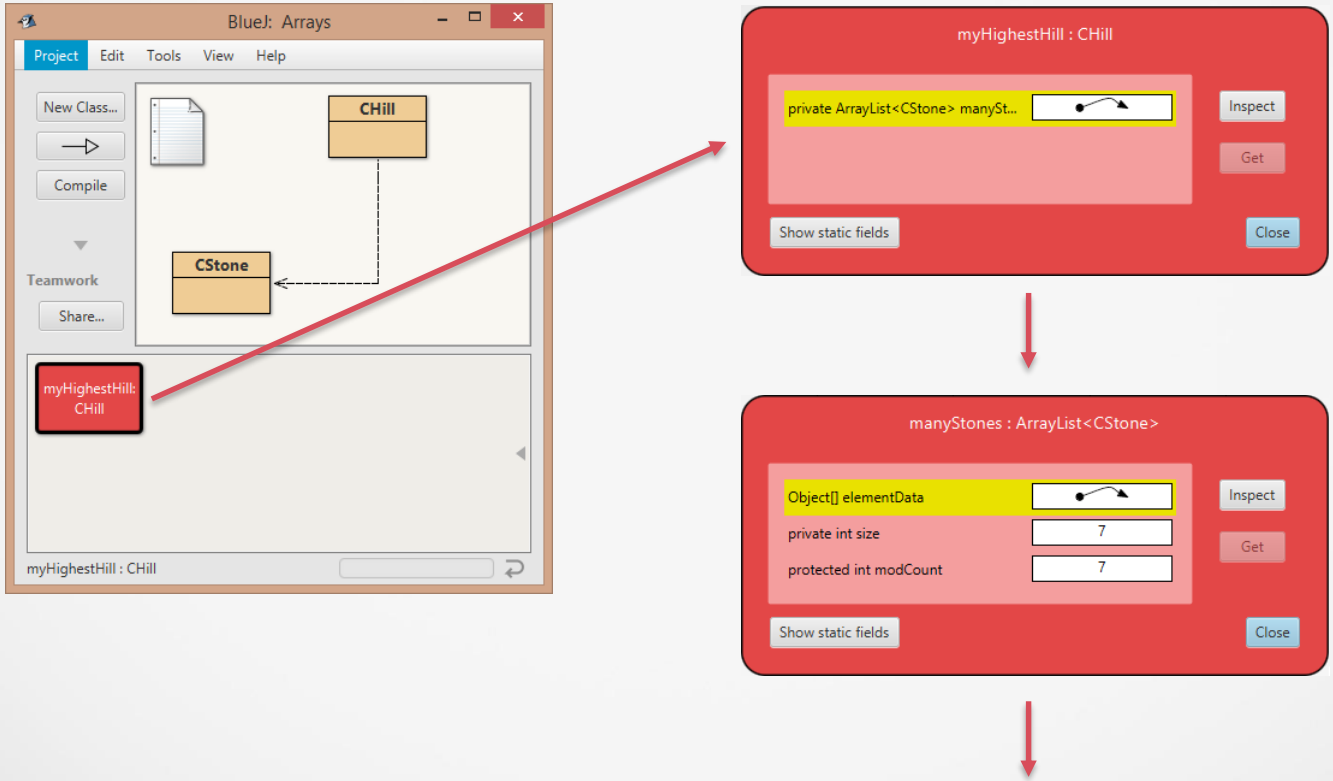
Debugging mit Arrays II

```
import java.util.*;
public class CHill{
   private ArrayList<CStone> manyStones= new ArrayList<CStone>();
   public CHill(){
      for (int i=0;i<(int)(Math.random()*49+1);i++){
         CStone stone=new CStone();
         stone.size=(int)(Math.random()*49+1);
         stone.weight=stone.size*3;
         manyStones.add(stone);
      }
   }
   public void manyStonesAreRolling(){
      for (CStone stone: manyStones){
         stone.roll();
      }
   }
}
```

Debugging mit Arrays III



Debugging mit Arrays IV



Debugging mit Arrays V