

## Script generated by TTT

Title: groh: profile1 (17.06.2016)

Date: Fri Jun 17 13:32:55 CEST 2016

Duration: 68:03 min

Pages: 101

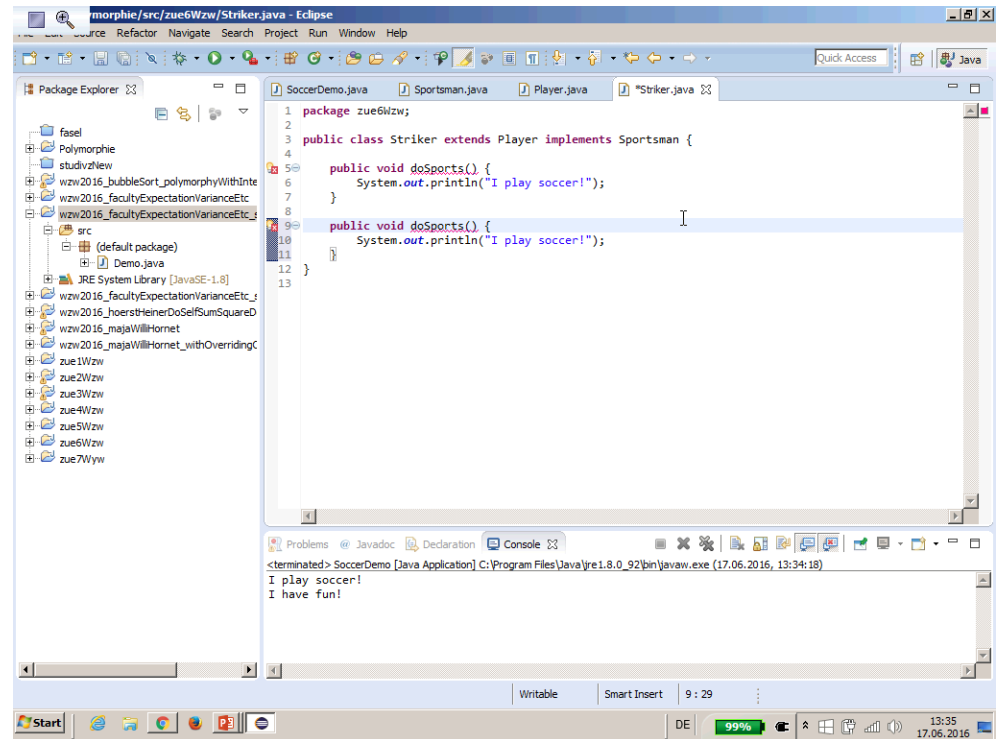
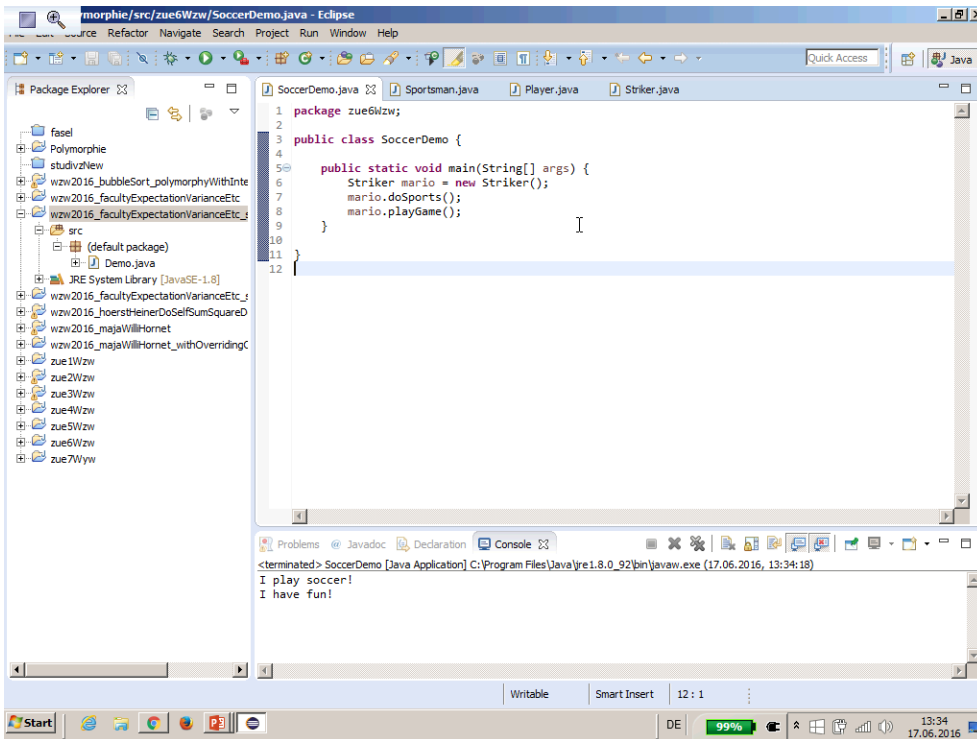


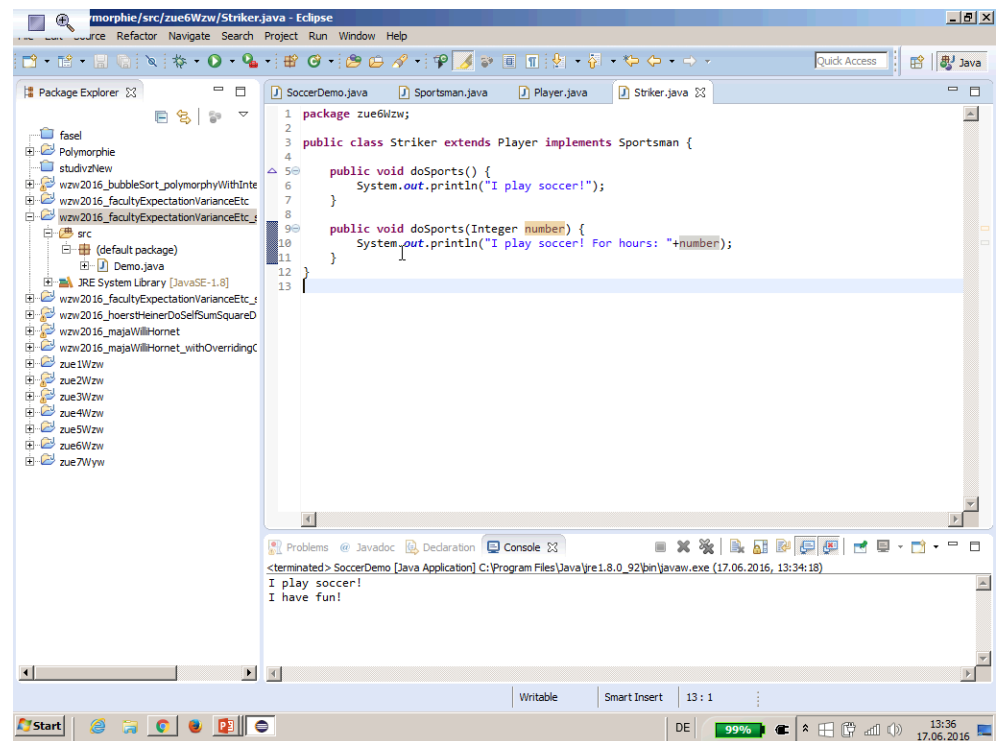
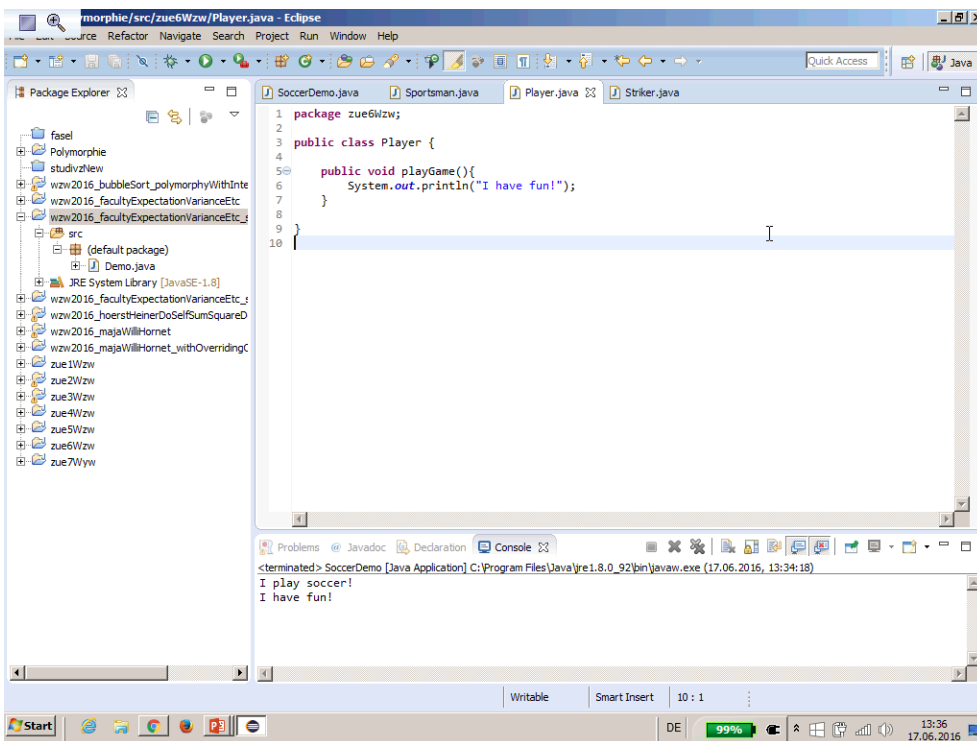
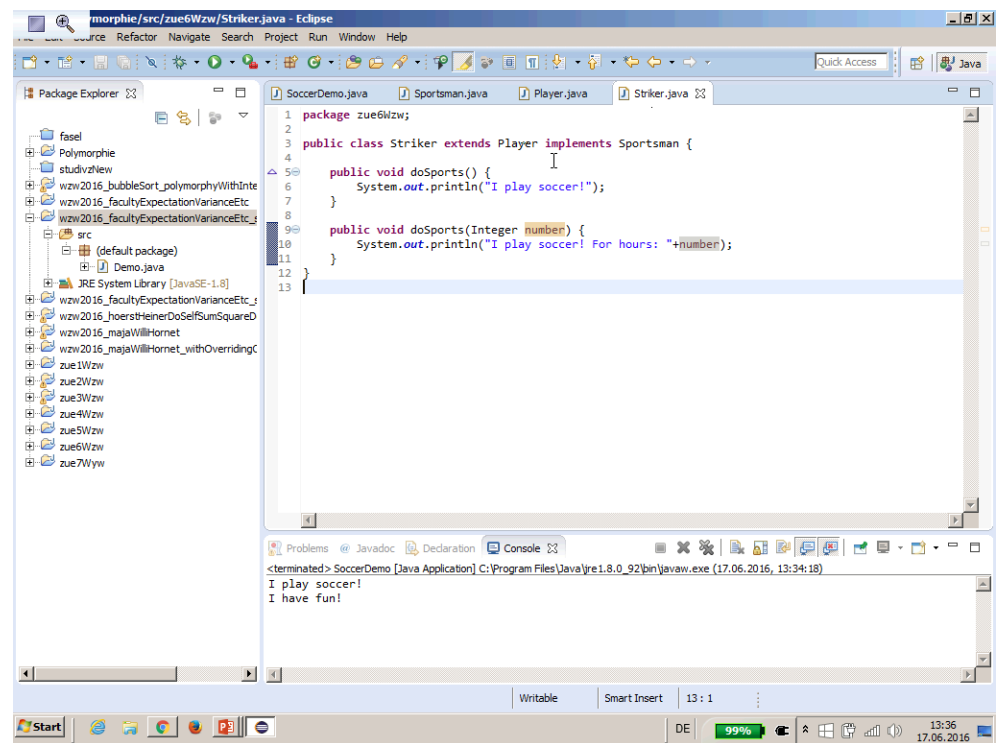
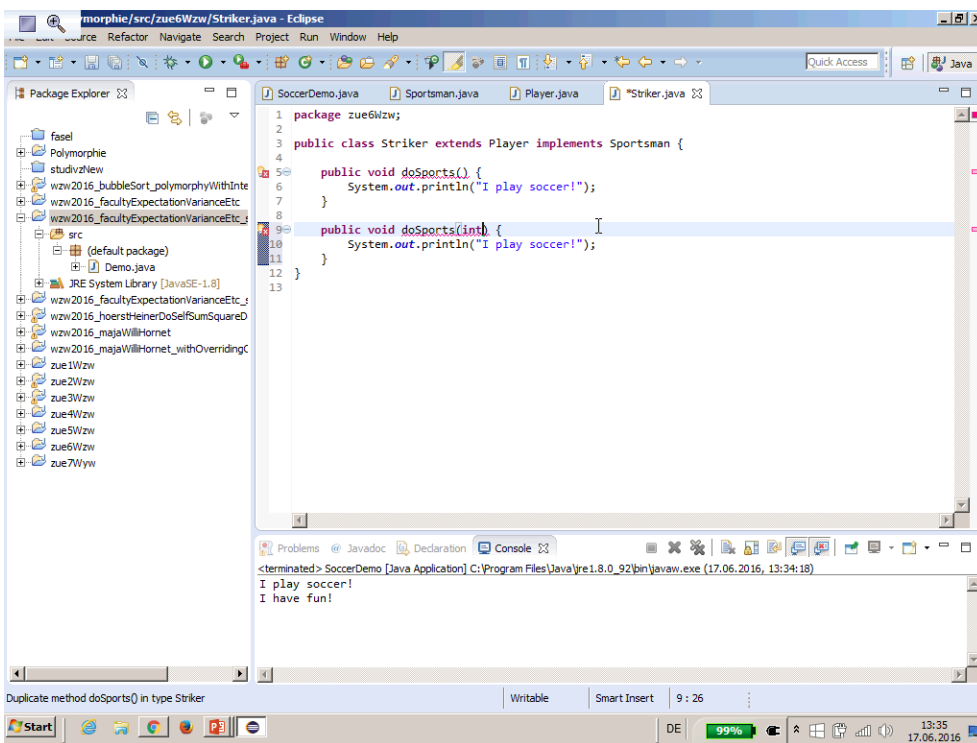
Gegeben sei folgender Java Code:

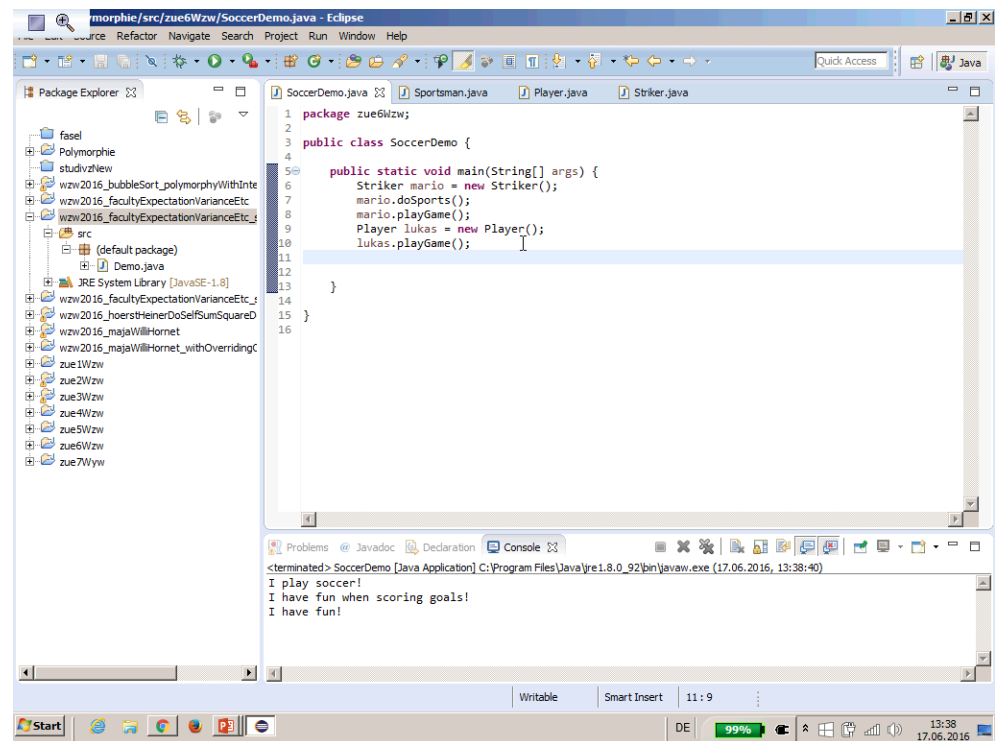
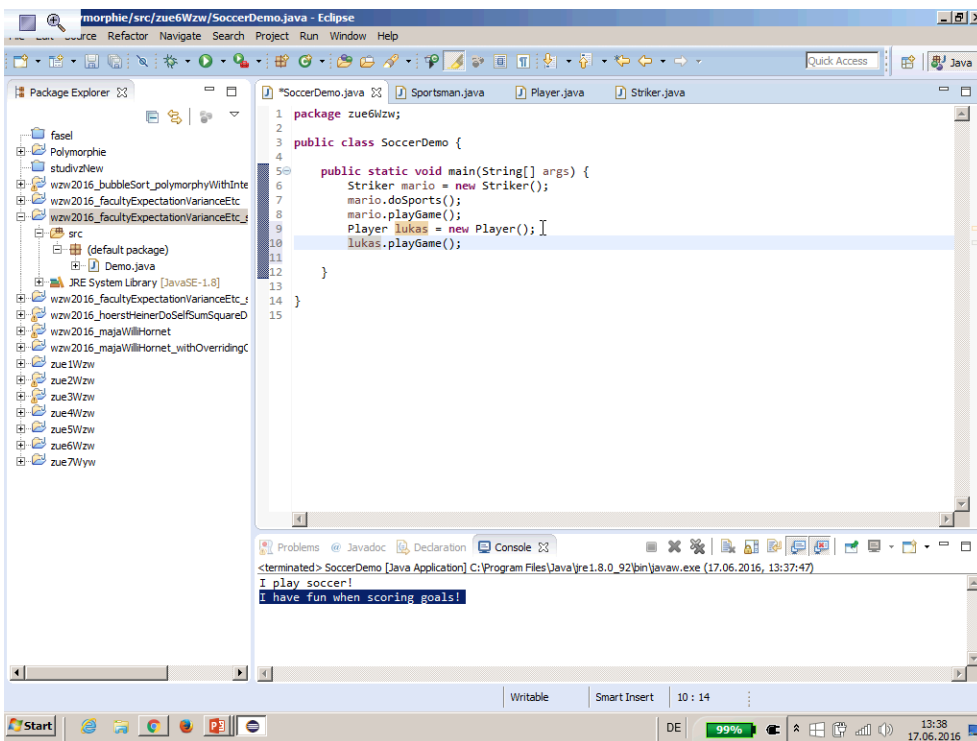
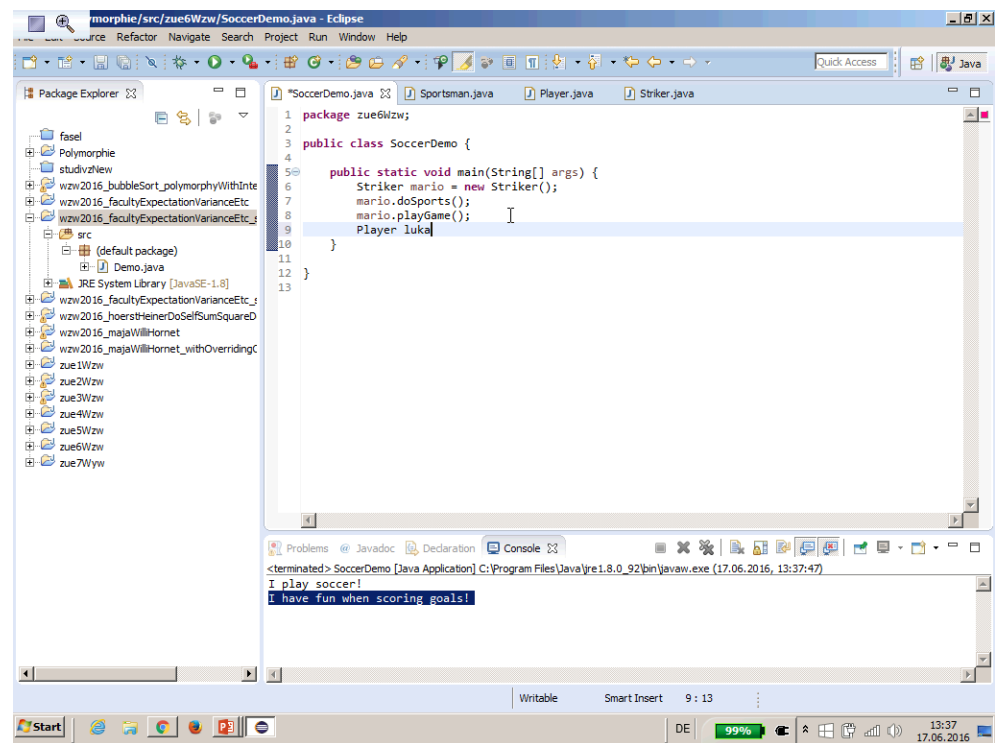
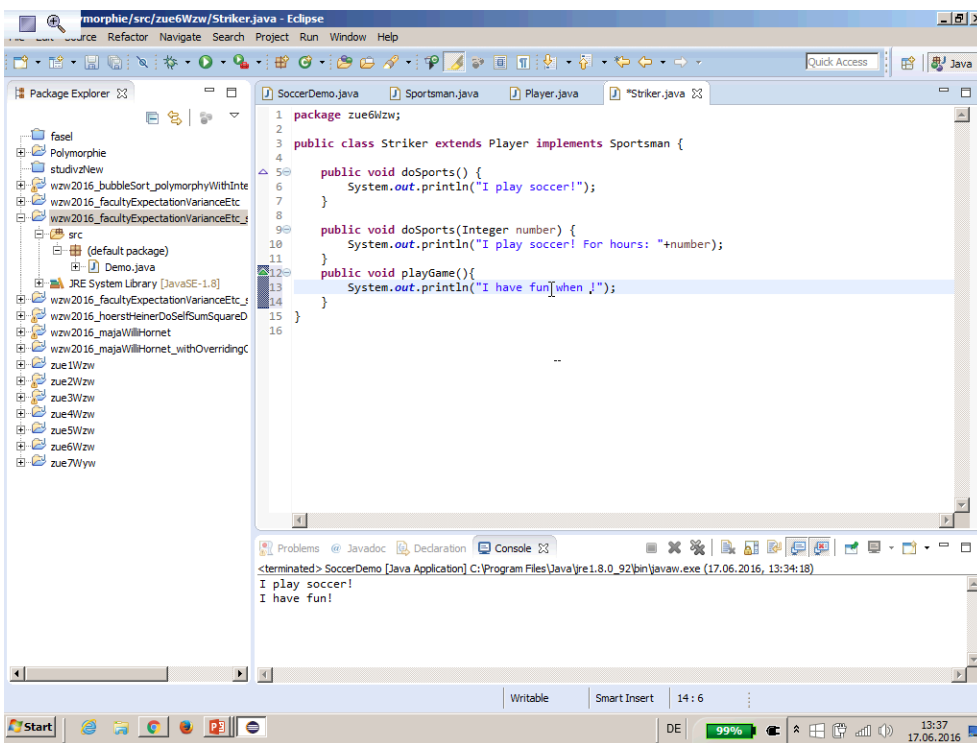
Welche Ausgabe produziert FinancialCrisisDemo?

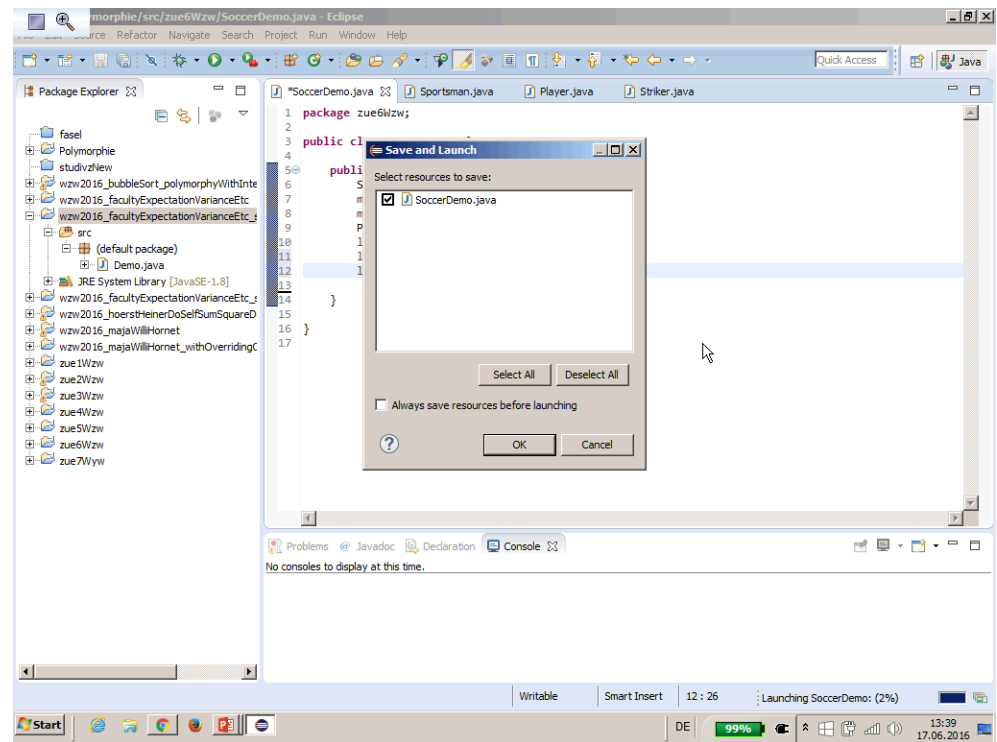
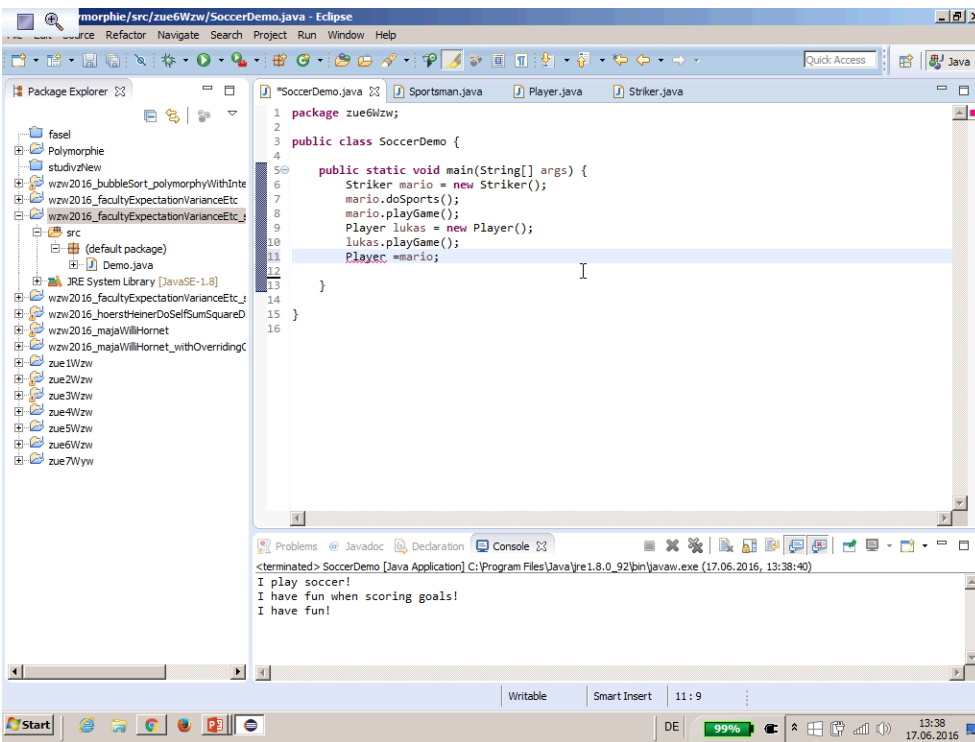
Welche der drei Konzepte **Overloading**, **Overriding**, **Polymorphie** werden im obigen Code benutzt, welche nicht? **BEGRÜNDEN** Sie jeweils kurz!

```
public class FinancialCrisisDemo {  
  
    public static void main(String[] args) {  
        MinisterOfFinance schaeuble = new MinisterOfFinance();  
        MinisterOfFinance varoufakis = new MinisterOfFinance();  
        schaeuble.commentOnCrisis();  
        GovernmentMember merkel = new GovernmentMember();  
        merkel.commentOnCrisis(" aehhh...");  
        GovernmentMember someOtherMember = schaeuble;  
        someOtherMember.commentOnCrisis(" aehhh...");  
        varoufakis.commentOnCrisis("whatever....i need more money!");  
    }  
}  
  
public class MinisterOfFinance extends GovernmentMember implements  
StateRepresentant{  
    public void commentOnCrisis(){  
        System.out.println("no comment at the moment");  
    }  
  
    public void commentOnCrisis(String what){  
        System.out.println(what);  
    }  
}  
  
public class GovernmentMember {  
    public void commentOnCrisis(String what){  
        System.out.println("i must say that " + what);  
    }  
}  
  
public interface StateRepresentant {  
    public void commentOnCrisis();  
}
```

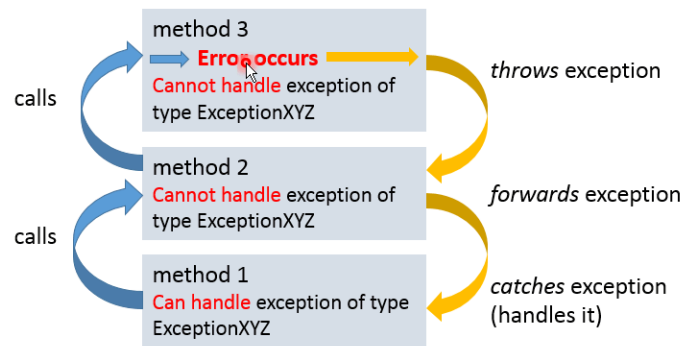




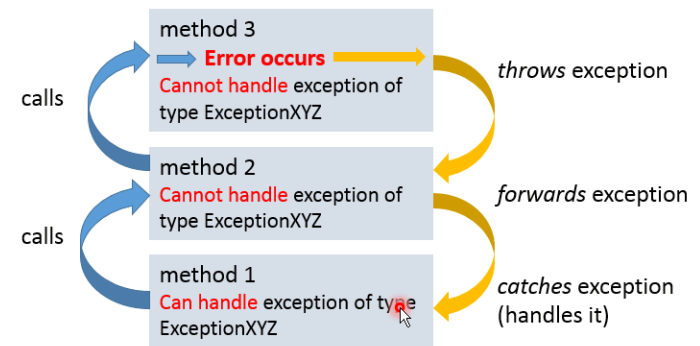




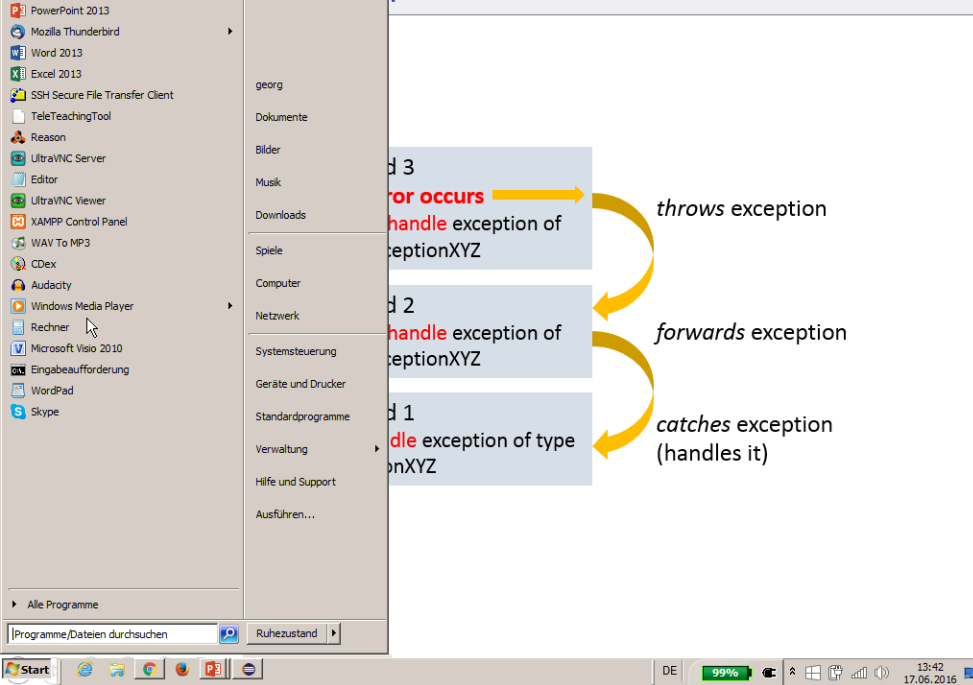
## Fehlerbehandlung - Exceptions



## Fehlerbehandlung - Exceptions



## Fehlerbehandlung - Exceptions



## 4.1: Exceptions

<http://docs.oracle.com/javase/tutorial/essential/exceptions/index.html>

mit den  
jeweiligen  
Unterseiten

160

## Exceptions: try catch

- **Fall 1:** x behandelt Exception selbst:  
**Probiere** gefährlichen Code aus und **fange** ggf. auftretende Exceptions selbst.

```
try {
    // ...
    FileWriter fileWriter = new FileWriter("someFileName.txt");
    fileWriter.write('a');
    // ...
} catch (IOException e) {
    // Exception handling code goes here
    e.printStackTrace();
} catch (SomeOtherException e) {
    // Exception handling code goes here
    e.printStackTrace();
} catch (Exception e) {
    // Exception handling code goes here
    e.printStackTrace();
}
```

163

## Exceptions: try catch

- **Fall 1:** x behandelt Exception selbst:  
**Probiere** gefährlichen Code aus und **fange** ggf. auftretende Exceptions selbst.

```
try {
    // ...
    FileWriter fileWriter = new FileWriter("someFileName.txt");
    fileWriter.write('a');
    // ...
} catch (IOException e) {
    // Exception handling code goes here
    e.printStackTrace();
} catch (SomeOtherException e) {
    // Exception handling code goes here
    e.printStackTrace();
} catch (Exception e) {
    // Exception handling code goes here
    e.printStackTrace();
}
```

163

## Exceptions: finally

- **Fall 1:** x behandelt Exception selbst:  
Probiere gefährlichen Code aus und **fange** ggf. auftretende Exceptions selbst.
  - **finally** – Block wird auf jeden Fall ausgeführt, egal ob Exception auftrat.

```
try {
    fileWriter = new FileWriter("someFileName.txt");
    fileWriter.write('a');
    // ...
} catch (Exception e) {
    // Exception handling code goes here
} finally {
    // Make sure that the file is closed, no matter whether
    // an exception occurred or not.
    if (fileWriter != null) {
        fileWriter.close();
    }
}
```

164

## Exceptions: finally

- **Fall 1:** x behandelt Exception selbst:  
Probiere gefährlichen Code aus und **fange** ggf. auftretende Exceptions selbst.
  - **finally** – Block wird auf jeden Fall ausgeführt, egal ob Exception auftrat.

```
try {
    fileWriter = new FileWriter("someFileName.txt");
    fileWriter.write('a');
    // ...
} catch (Exception e) {
    // Exception handling code goes here
} finally {
    // Make sure that the file is closed, no matter whether
    // an exception occurred or not.
    if (fileWriter != null) {
        fileWriter.close();
    }
}
```

164

## Exceptions: throws

- **Fall 2:** Jemand anders (aufrufende Methode y) soll Exception behandeln:  
→ füge **throws** clause zu Methodendefinition hinzu:

```
class Bicycle {
    int gear;

    Bicycle(int initialGear) throws InvalidGearException {
        if (initialGear > 0) {
            gear = initialGear;
        } else {
            throw new InvalidGearException();
        }
    }

    void inflateTires() throws TireExplodedException {
        // ...
    }
    ...
}
```

```
class InvalidGearException extends Exception {...}
```

```
class TireExplodedException extends Exception {...}
```

165

## Exceptions: throws

- **Fall 2:** Jemand anders (aufrufende Methode y) soll Exception behandeln:  
→ füge **throws** clause zu Methodendefinition hinzu:

```
class Bicycle {
    int gear;

    Bicycle(int initialGear) throws InvalidGearException {
        if (initialGear > 0) {
            gear = initialGear;
        } else {
            throw new InvalidGearException();
        }
    }

    void inflateTires() throws TireExplodedException {
        // ...
    }
    ...
}
```

```
class InvalidGearException extends Exception {...}
```

```
class TireExplodedException extends Exception {...}
```

165

- **Fall 2:** Jemand anders (aufrufende Methode y) soll Exception behandeln:  
→ füge **throws** clause zu Methodendefinition hinzu:

```
class Bicycle {
    int gear;

    Bicycle(int initialGear) throws InvalidGearException {
        if (initialGear > 0) {
            gear = initialGear;
        } else {
            throw new InvalidGearException();
        }
    }

    void inflateTires() throws TireExplodedException {
        // ...
    }
    ...
}
```

```
class InvalidGearException extends Exception {...}
```

```
class TireExplodedException extends Exception {...}
```

- **Checked Exceptions:**  
Unterklassen von `java.lang.Exception`.  
**MÜSSEN** gefangen oder weitergeworfen werden

- **Unchecked Exceptions:**  
Unterklassen von `java.lang.RuntimeException` or `java.lang.Error`  
**KÖNNEN** gefangen oder weitergeworfen werden .  
Kein try/catch oder throws clause notwendig.

```
static long factorial(int n) {
    if (n < 0) {
        throw new RuntimeException("Check your math!")
    } else if (n == 1) {
        return 1;
    }
    return n * factorial(n-1);
}
```

- **Checked Exceptions:**  
Unterklassen von `java.lang.Exception`.  
**MÜSSEN** gefangen oder weitergeworfen werden

- **Unchecked Exceptions:**  
Unterklassen von `java.lang.RuntimeException` or `java.lang.Error`  
**KÖNNEN** gefangen oder weitergeworfen werden .  
Kein try/catch oder throws clause notwendig.

```
static long factorial(int n) {
    if (n < 0) {
        throw new RuntimeException("Check your math!")
    } else if (n == 1) {
        return 1;
    }
    return n * factorial(n-1);
}
```

- **Checked Exceptions:**  
Unterklassen von `java.lang.Exception`.  
**MÜSSEN** gefangen oder weitergeworfen werden

- **Unchecked Exceptions:**  
Unterklassen von `java.lang.RuntimeException` or `java.lang.Error`  
**KÖNNEN** gefangen oder weitergeworfen werden .  
Kein try/catch oder throws clause notwendig.

```
static long factorial(int n) {
    if (n < 0) {
        throw new RuntimeException("Check your math!")
    } else if (n == 1) {
        return 1;
    }
    return n * factorial(n-1);
}
```

### Live Demo:

- Exceptions bei Methode faculty einbauen

### Studierenden-Aufgabe:

- Programmieren einer Methode quadraticEquationSolver mit Mitternachtsformel und Exceptions bei komplexwertigen Lösungen einbauen

```
116     return result;
117     }
118
119
120     public static int minimum2(int a, int b){
121         if(a < b)
122             return a;
123         else
124             return b;
125     }
126
127
128     public static long faculty(int a){
129         long result = 1;
130         while(a > 1){
131             result = result * a;
132             a = a - 1;
133         }
134         return result;
135     }
136
137     public static long facultyWithFor(int a){
138         long result = 1;
139         for(int i=1; i<=a; i++){
140             result = result * i;
141         }
142         return result;
143     }
```

Console Output:  
<terminated> SoccerDemo [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (17.06.2016, 13:39:49)  
I play soccer!  
I have fun when scoring goals!  
I have fun!  
I have fun when scoring goals!

167

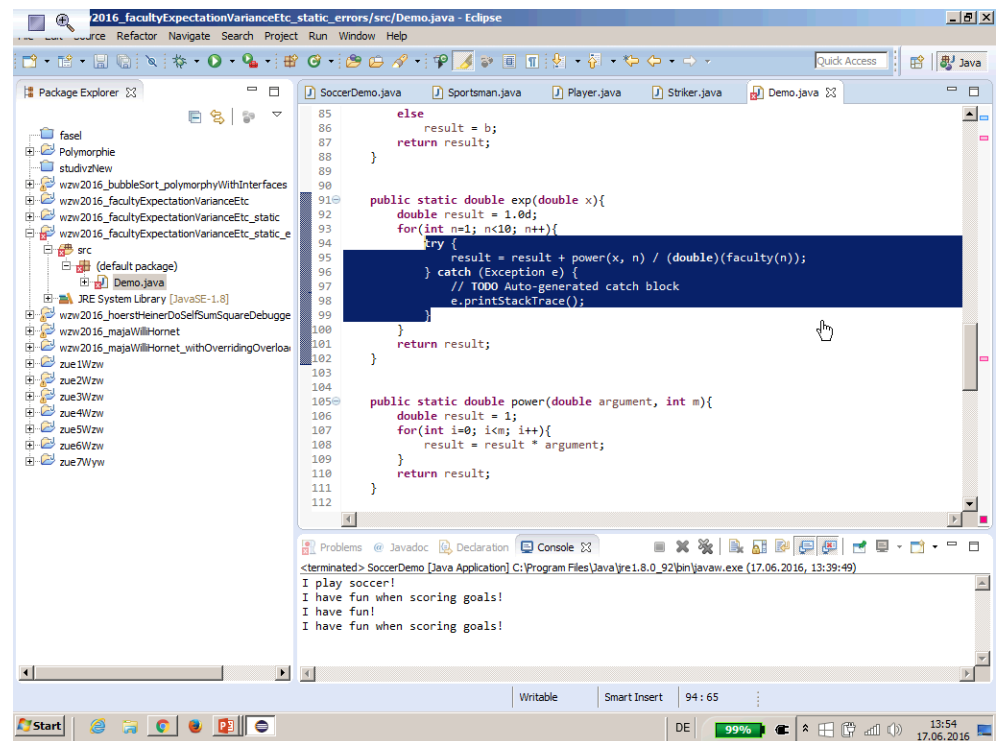
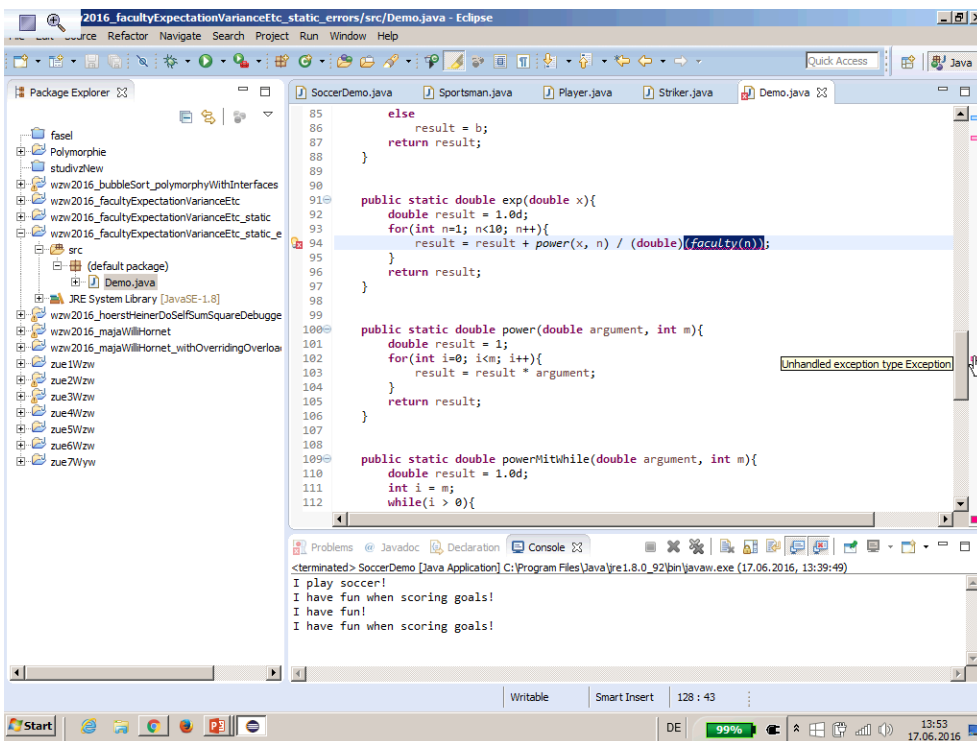
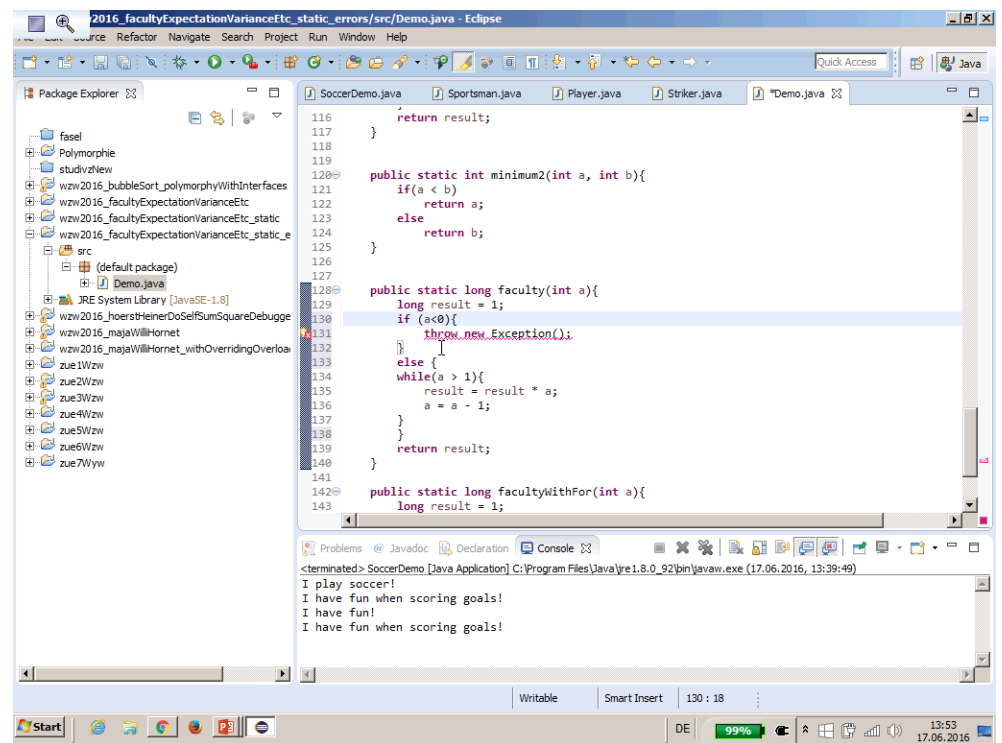
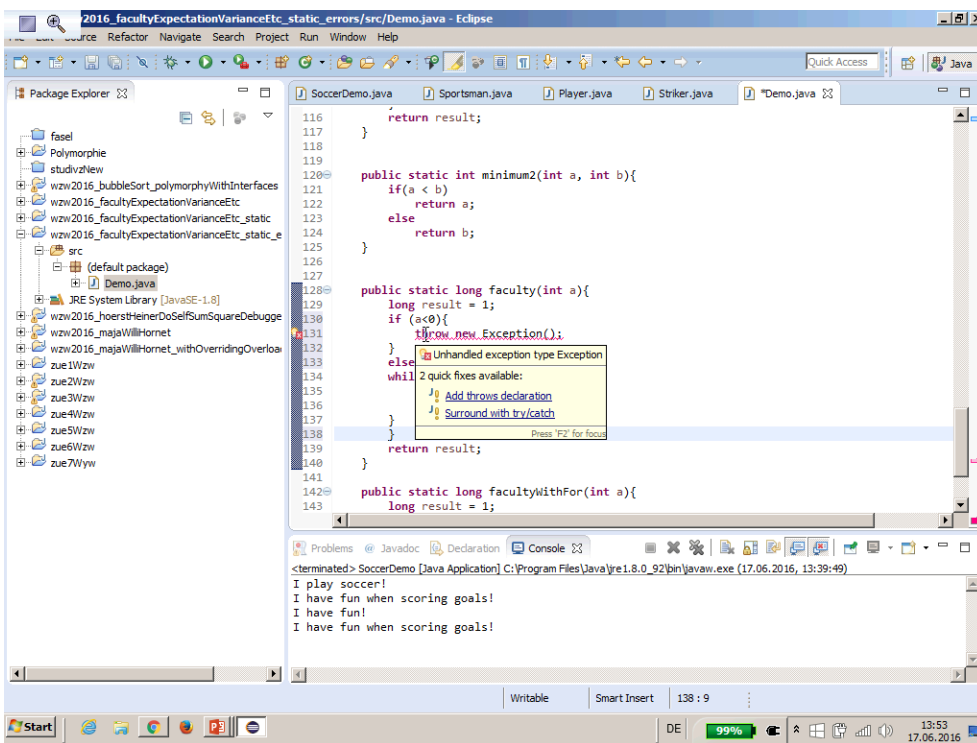
```
116     return result;
117     }
118
119
120     public static int minimum2(int a, int b){
121         if(a < b)
122             return a;
123         else
124             return b;
125     }
126
127
128     public static long faculty(int a){
129         long result = 1;
130         while(a > 1){
131             result = result * a;
132             a = a - 1;
133         }
134         return result;
135     }
136
137     public static long facultyWithFor(int a){
138         long result = 1;
139         for(int i=1; i<=a; i++){
140             result = result * i;
141         }
142         return result;
143     }
```

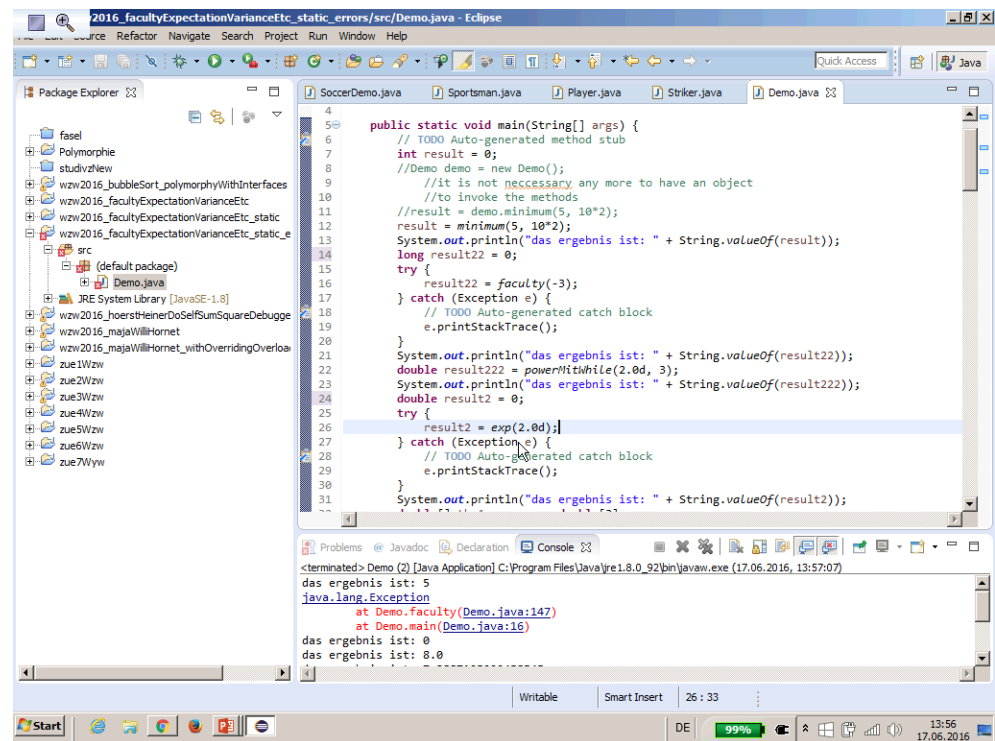
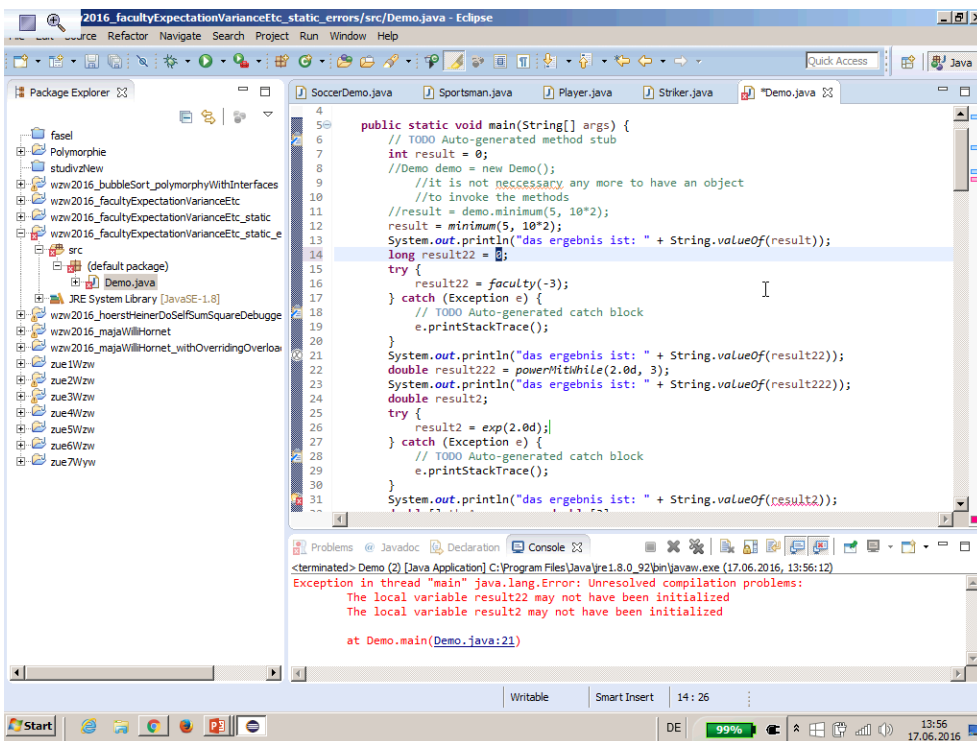
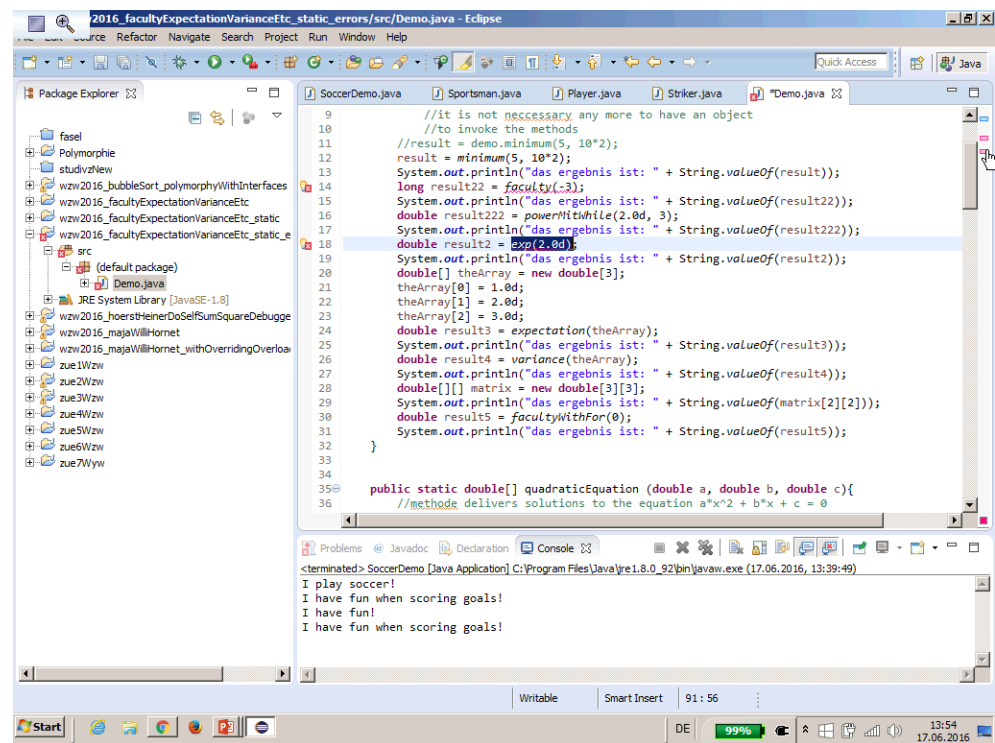
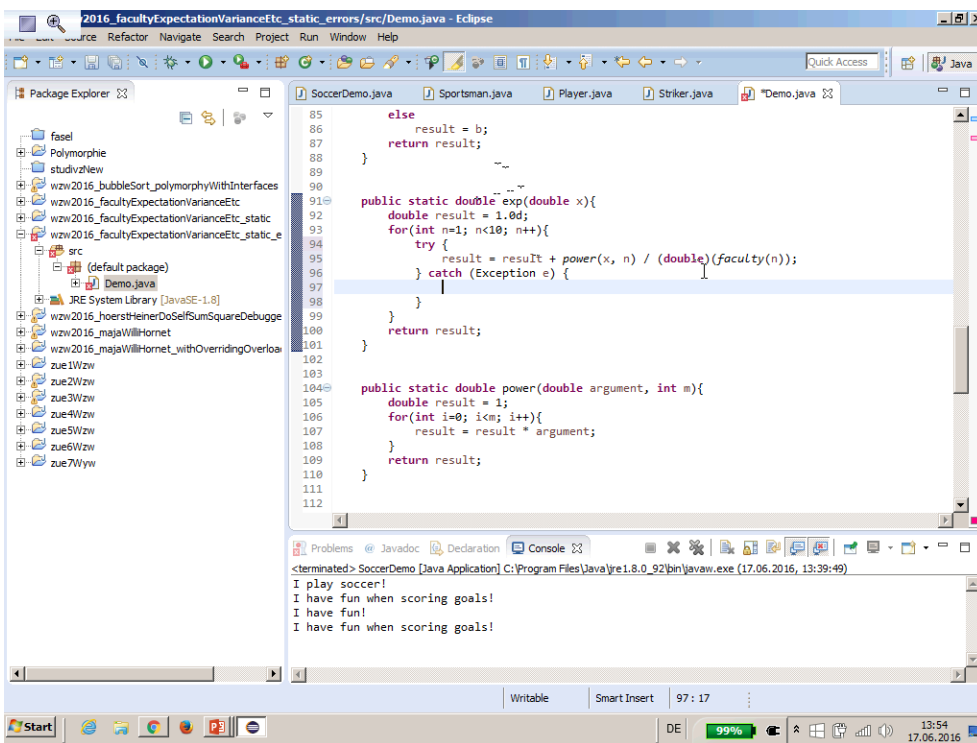
Console Output:  
<terminated> SoccerDemo [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (17.06.2016, 13:39:49)  
I play soccer!  
I have fun when scoring goals!  
I have fun!  
I have fun when scoring goals!

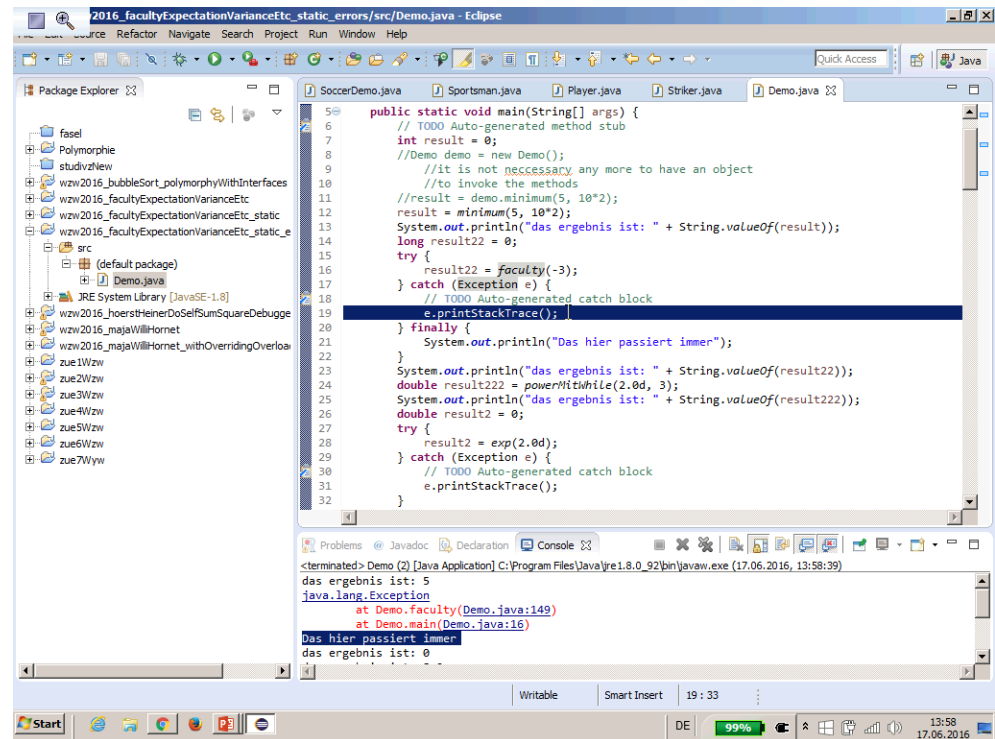
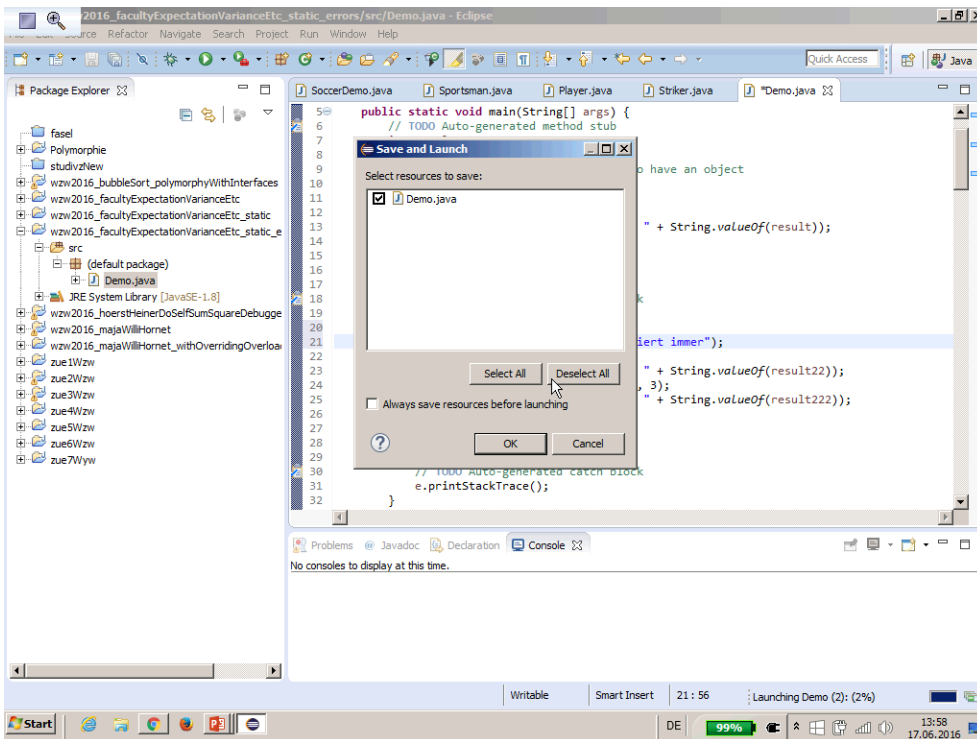
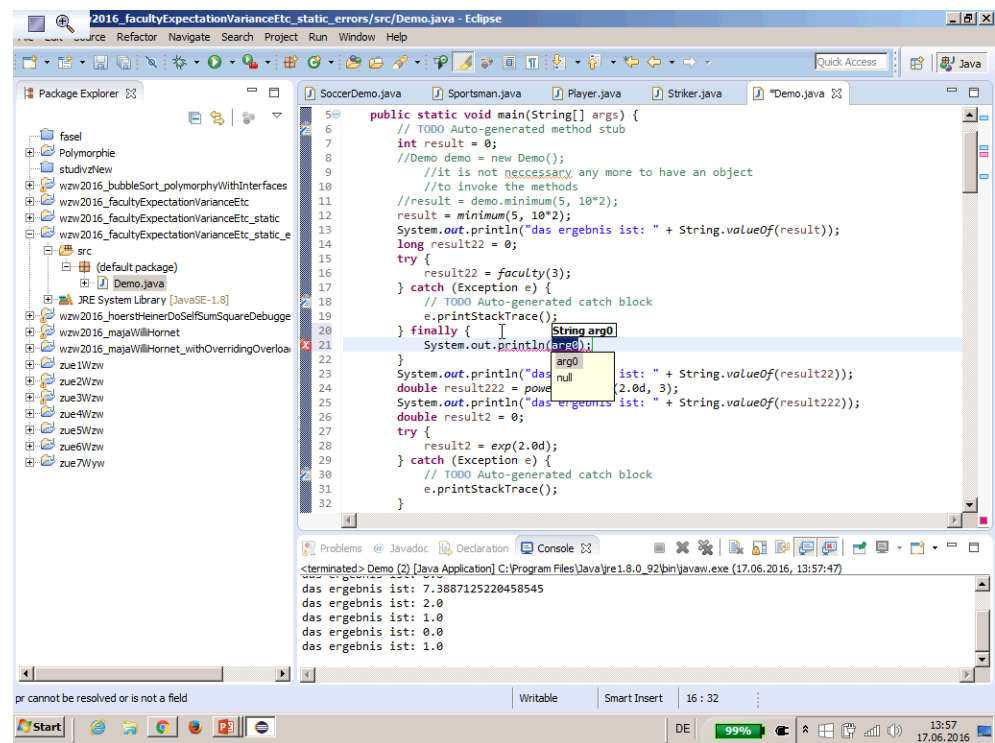
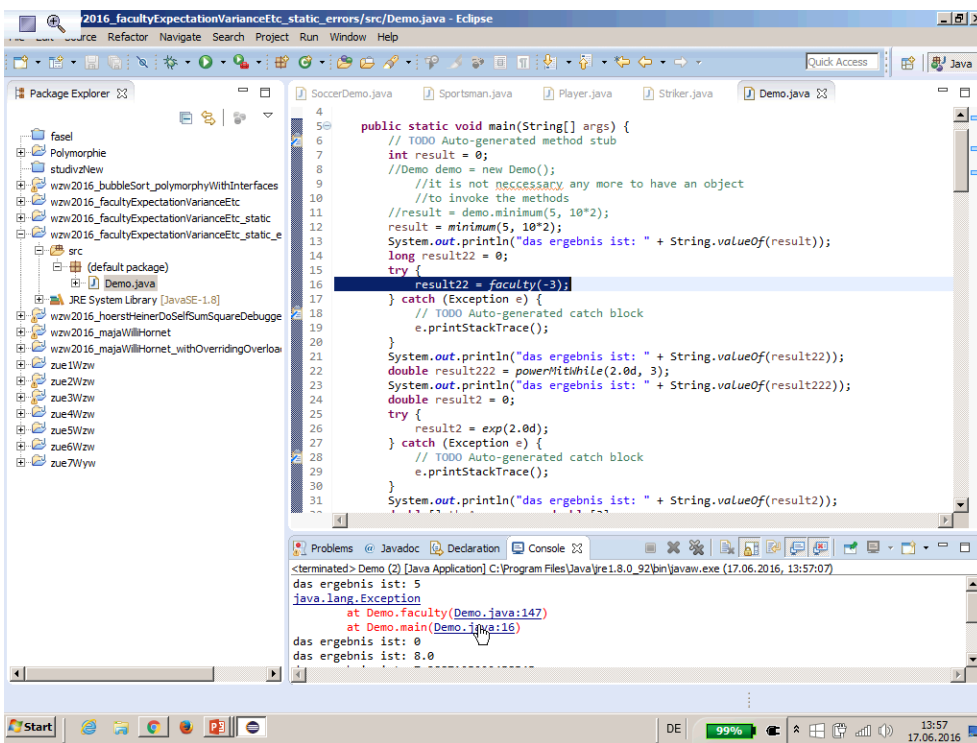
```
116     return result;
117     }
118
119
120     public static int minimum2(int a, int b){
121         if(a < b)
122             return a;
123         else
124             return b;
125     }
126
127
128     public static long faculty(int a){
129         long result = 1;
130         while(a > 1){
131             result = result * a;
132             a = a - 1;
133         }
134         return result;
135     }
136
137     public static long facultyWithFor(int a){
138         long result = 1;
139         for(int i=1; i<=a; i++){
140             result = result * i;
141         }
142         return result;
143     }
```

Console Output:  
<terminated> SoccerDemo [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (17.06.2016, 13:39:49)  
I play soccer!  
I have fun when scoring goals!  
I have fun!  
I have fun when scoring goals!











```

25 SoccerDemo.java | Sportsman.java | Player.java | Striker.java | Demo.java
25 // ...
26 System.out.println("das ergebnis ist: " + String.valueOf(result222));
27 double result2 = 0;
28 try {
29     result2 = exp(2.0d);
30 } catch (Exception e) {
31     // TODO Auto-generated catch block
32     e.printStackTrace();
33 }
34 System.out.println("das ergebnis ist: " + String.valueOf(result2));
35 double[] theArray = new double[3];
36 theArray[0] = 1.0d;
37 theArray[1] = 2.0d;
38 theArray[2] = 3.0d;
39 double result3 = expectation(theArray);
40 System.out.println("das ergebnis ist: " + String.valueOf(result3));
41 double result4 = variance(theArray);
42 System.out.println("das ergebnis ist: " + String.valueOf(result4));
43 double[][] matrix = new double[3][3];
44 System.out.println("das ergebnis ist: " + String.valueOf(matrix[2][2]));
45 double result5 = facultyWithFor(0);
46 System.out.println("das ergebnis ist: " + String.valueOf(result5));
47 try{double[] resultQuad = quadraticEquation (1,2,3);}
48 catch (Exception e){
49     System.out.println("War komplex!");
50 }
51
52

```

Problems @ Javadoc Declaration Console

```

<terminated> Demo (2) [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (17.06.2016, 13:58:39)
das ergebnis ist: 5
java.lang.Exception
    at Demo.faculty(Demo.java:149)
    at Demo.main(Demo.java:16)
Das hier passiert immer
das ergebnis ist: 0

```

```

25 SoccerDemo.java | Sportsman.java | Player.java | Striker.java | Demo.java
25 // ...
26 System.out.println("das ergebnis ist: " + String.valueOf(result222));
27 double result2 = 0;
28 try {
29     result2 = exp(2.0d);
30 } catch (Exception e) {
31     // TODO Auto-generated catch block
32     e.printStackTrace();
33 }
34 System.out.println("das ergebnis ist: " + String.valueOf(result2));
35 double[] theArray = new double[3];
36 theArray[0] = 1.0d;
37 theArray[1] = 2.0d;
38 theArray[2] = 3.0d;
39 double result3 = expectation(theArray);
40 System.out.println("das ergebnis ist: " + String.valueOf(result3));
41 double result4 = variance(theArray);
42 System.out.println("das ergebnis ist: " + String.valueOf(result4));
43 double[][] matrix = new double[3][3];
44 System.out.println("das ergebnis ist: " + String.valueOf(matrix[2][2]));
45 double result5 = facultyWithFor(0);
46 System.out.println("das ergebnis ist: " + String.valueOf(result5));
47 try{double[] resultQuad = quadraticEquation (1,2,3);}
48 catch (Exception e){
49     System.out.println("War komplex!");
50 }
51
52

```

Problems @ Javadoc Declaration Console

```

<terminated> Demo (2) [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (17.06.2016, 13:58:39)
das ergebnis ist: 5
java.lang.Exception
    at Demo.faculty(Demo.java:149)
    at Demo.main(Demo.java:16)
Das hier passiert immer
das ergebnis ist: 0

```

## Code-Formatierung

- Einrücken (4 spaces) und max. 80 Zeichen /Zeile

```

public void initializeMatrix(int[][] someMatrix) {
    for (int i = 0; i < someMatrix.length; i++) {
        for (int j = 0; j < someMatrix.length; j++) {
            if (i % 2 == 0) {
                if (i == 2 * j) {
                    someMatrix[i][j] = 7;
                } else {
                    someMatrix[i][j] = 3 * i + 72 * j +
                        (int)Math.floor(i / (j + 1));
                }
            }
        }
    }
    System.out.println("This is the end!");
}

```

```

25 SoccerDemo.java | Sportsman.java | Player.java | Striker.java | Demo.java
25 // ...
26 System.out.println("das ergebnis ist: " + String.valueOf(result222));
27 double result2 = 0;
28 try {
29     result2 = exp(2.0d);
30 } catch (Exception e) {
31     // TODO Auto-generated catch block
32     e.printStackTrace();
33 }
34 System.out.println("das ergebnis ist: " + String.valueOf(result2));
35 double[] theArray = new double[3];
36 theArray[0] = 1.0d;
37 theArray[1] = 2.0d;
38 theArray[2] = 3.0d;
39 double result3 = expectation(theArray);
40 System.out.println("das ergebnis ist: " + String.valueOf(result3));
41 double result4 = variance(theArray);
42 System.out.println("das ergebnis ist: " + String.valueOf(result4));
43 double[][] matrix = new double[3][3];
44 System.out.println("das ergebnis ist: " + String.valueOf(matrix[2][2]));
45 double result5 = facultyWithFor(0);
46 System.out.println("das ergebnis ist: " + String.valueOf(result5));
47 try{double[] resultQuad = quadraticEquation (1,2,3);}
48 catch (Exception e){
49     System.out.println("War komplex!");
50 }
51
52

```

Problems @ Javadoc Declaration Console

```

<terminated> Demo (2) [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (17.06.2016, 13:58:39)
das ergebnis ist: 5
java.lang.Exception
    at Demo.faculty(Demo.java:149)
    at Demo.main(Demo.java:16)
Das hier passiert immer
das ergebnis ist: 0

```

- **Generell:** kurz, einfach, für Dritte interpretierbar, deuten Sinn und Zweck an, immer auf Englisch, nur [a..z], [A..Z], [0..9] nutzen
- **Methoden und Variablen:** kleingeschrieben, interne Worte: Anfangsbuchstabe groß, Boolean mit prefix „is“

```
int rowIndex
int columnIndex
boolean isValid
boolean copyFile(URL from, URL to)
```

- **Konstanten:** Großbuchstaben, interne Worte mit „\_“

```
static final int MIN_ALLOWED_VALUE = 42;
```

- **Klassen und Interfaces:** wie Methoden u. Variablen nur erster Buchstabe groß

```
class Bicycle
class ImageSprite
interface RasterDelegate
```

172

- **Generell:** kurz, einfach, für Dritte interpretierbar, deuten Sinn und Zweck an, immer auf Englisch, nur [a..z], [A..Z], [0..9] nutzen
- **Methoden und Variablen:** kleingeschrieben, interne Worte: Anfangsbuchstabe groß, Boolean mit prefix „is“

```
int rowIndex
int columnIndex
boolean isValid
boolean copyFile(URL from, URL to)
```

- **Konstanten:** Großbuchstaben, interne Worte mit „\_“

```
static final int MIN_ALLOWED_VALUE = 42;
```

- **Klassen und Interfaces:** wie Methoden u. Variablen nur erster Buchstabe groß

```
class Bicycle
class ImageSprite
interface RasterDelegate
```

172

- **Generell:** kurz, einfach, für Dritte interpretierbar, deuten Sinn und Zweck an, immer auf Englisch, nur [a..z], [A..Z], [0..9] nutzen
- **Methoden und Variablen:** kleingeschrieben, interne Worte: Anfangsbuchstabe groß, Boolean mit prefix „is“

```
int rowIndex
int columnIndex
boolean isValid
boolean copyFile(URL from, URL to)
```

- **Konstanten:** Großbuchstaben, interne Worte mit „\_“

```
static final int MIN_ALLOWED_VALUE = 42;
```

- **Klassen und Interfaces:** wie Methoden u. Variablen nur erster Buchstabe groß

```
class Bicycle
class ImageSprite
interface RasterDelegate
```

172

- Häufig und sorgfältig **Kommentare** und **JavaDoc** benutzen!

```
//Recursive version of faculty
/**
 * Computes faculty f(n)=n!.
 * @param n: Argument of faculty
 * @return n!
 */
public long faculty(int n)
{
    if (n<0)
        return -1; // -1 as error code; fix me!
    else
        if (n==0)
            return 1; // 0! == 1;
        else
            return n*faculty(n-1); // recursive call
}
```

173

- Häufig und sorgfältig **Kommentare** und **JavaDoc** benutzen!

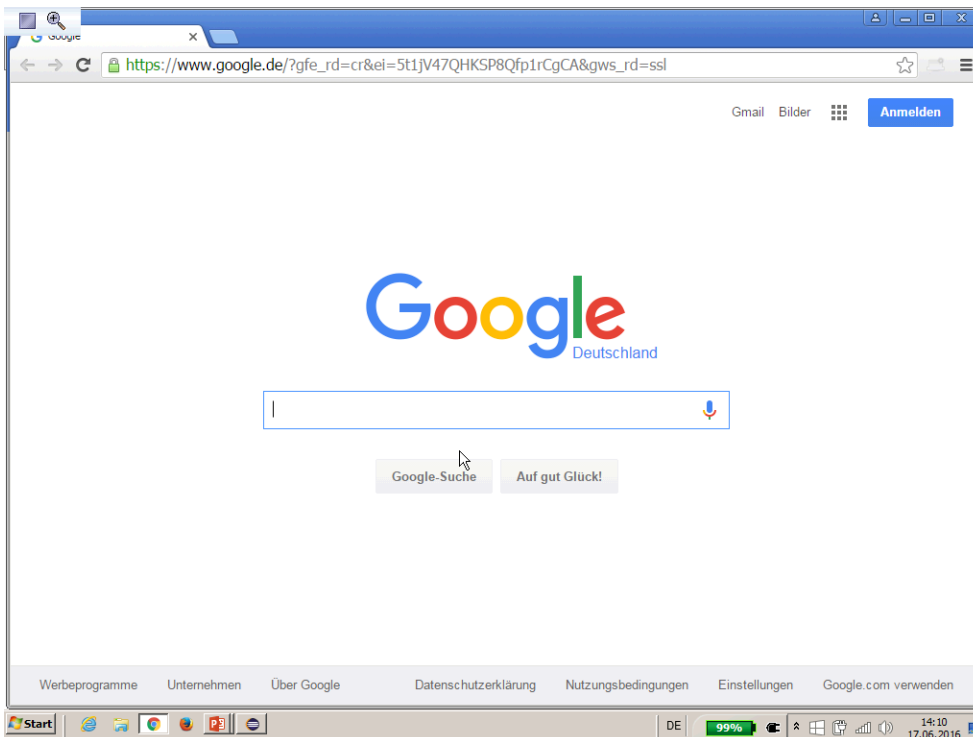
```
//Recursive version of faculty
/**
 * Computes faculty f(n)=n!.
 * @param n: Argument of faculty
 * @return n!
 */
public long faculty(int n)
{
    if (n<0)
        return -1; //-1 as error code; fix me!
    else
        if (n==0)
            return 1; //0!==1;
        else
            return n*faculty(n-1); //recursive call
}
```

173

- Häufig und sorgfältig **Kommentare** und **JavaDoc** benutzen!

```
//Recursive version of faculty
/**
 * Computes faculty f(n)=n!.
 * @param n: Argument of faculty
 * @return n!
 */
public long faculty(int n)
{
    if (n<0)
        return -1; //-1 as error code; fix me!
    else
        if (n==0)
            return 1; //0!==1;
        else
            return n*faculty(n-1); //recursive call
}
```

173



- Java Klassen sind in Hierarchie von **packages** organisiert

```
java.lang.String
java.net.URLConnection
java.util.Collection
javax.xml.parsers.SAXParser
org.w3c.dom.events.DocumentEvent
```

- Klassen die man nutzen möchte, muss man **importieren**.  
Alle Klassen aus java.lang werden automatisch importiert

```
import java.net.URLConnection;
import java.util.Collection;
import some.other.package.*; // * means all classes

class SomeClass{
    ...
}
```

177

- Manche Klassen kann man mit anderen Klassen **parametrisieren**
- Typisches **Beispiel: Datenstrukturen**
- **Vorteil:** Algorithmen und Datenstrukturen können **generisch** implementiert und genutzt werden.

ohne Generics:

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

mit Generics:

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```

- Manche Klassen kann man mit anderen Klassen **parametrisieren**
- Typisches **Beispiel: Datenstrukturen**
- **Vorteil:** Algorithmen und Datenstrukturen können **generisch** implementiert und genutzt werden.

ohne Generics:

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

mit Generics:

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```

- Manche Klassen kann man mit anderen Klassen **parametrisieren**
- Typisches **Beispiel: Datenstrukturen**
- **Vorteil:** Algorithmen und Datenstrukturen können **generisch** implementiert und genutzt werden.

ohne Generics:

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

mit Generics:

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```

- Für **jeden primitiven Typ** gibt es eine entsprechende **Wrapper-Klasse**

```
java.lang.Short
java.lang.Integer
java.lang.Long
java.lang.Float
java.lang.Double
java.lang.Boolean
...
```

- Zur Bequemlichkeit im Umgang mit diesen Wrapper Klassen gibt es **Boxing:**

statt:

```
Integer someInteger = new Integer(723);
```

kann man schreiben:

```
Integer someInteger = 723;
```



- Für jeden primitiven Typ gibt es eine entsprechende Wrapper-Klasse

```
java.lang.Short
java.lang.Integer
java.lang.Long
java.lang.Float
java.lang.Double
java.lang.Boolean
...
```

- Zur Bequemlichkeit im Umgang mit diesen Wrapper Klassen gibt es **Boxing**:

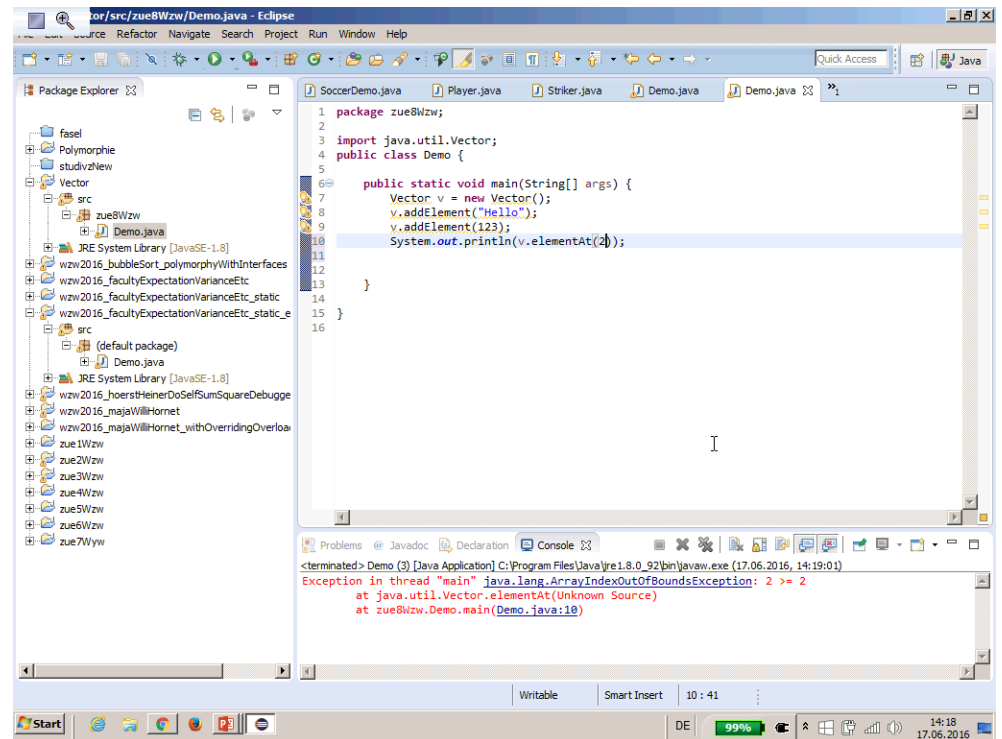
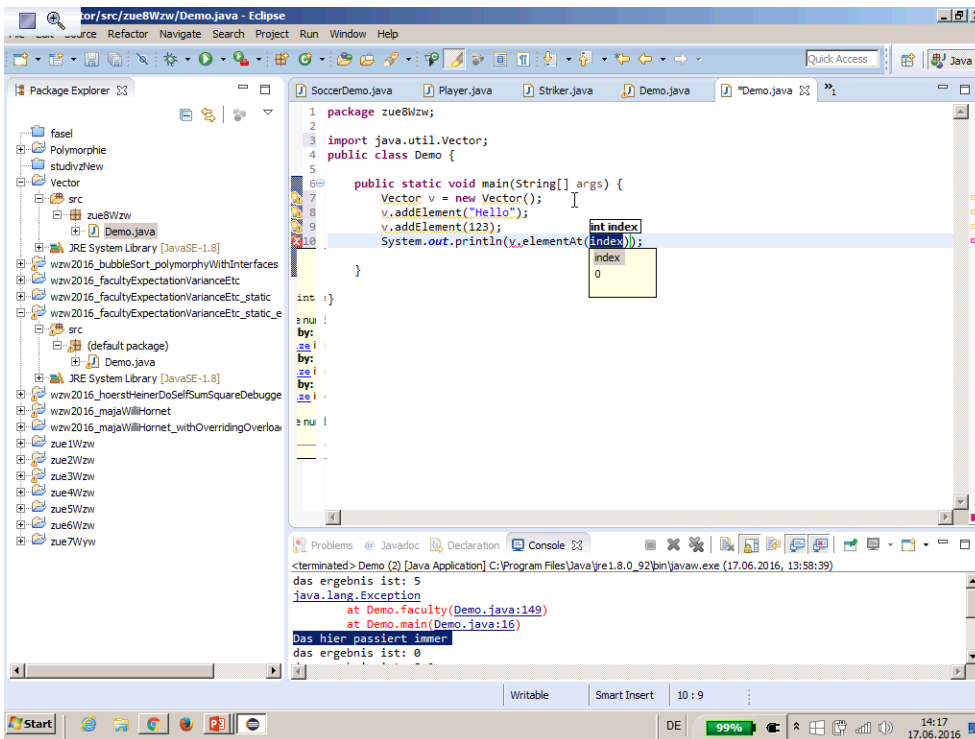
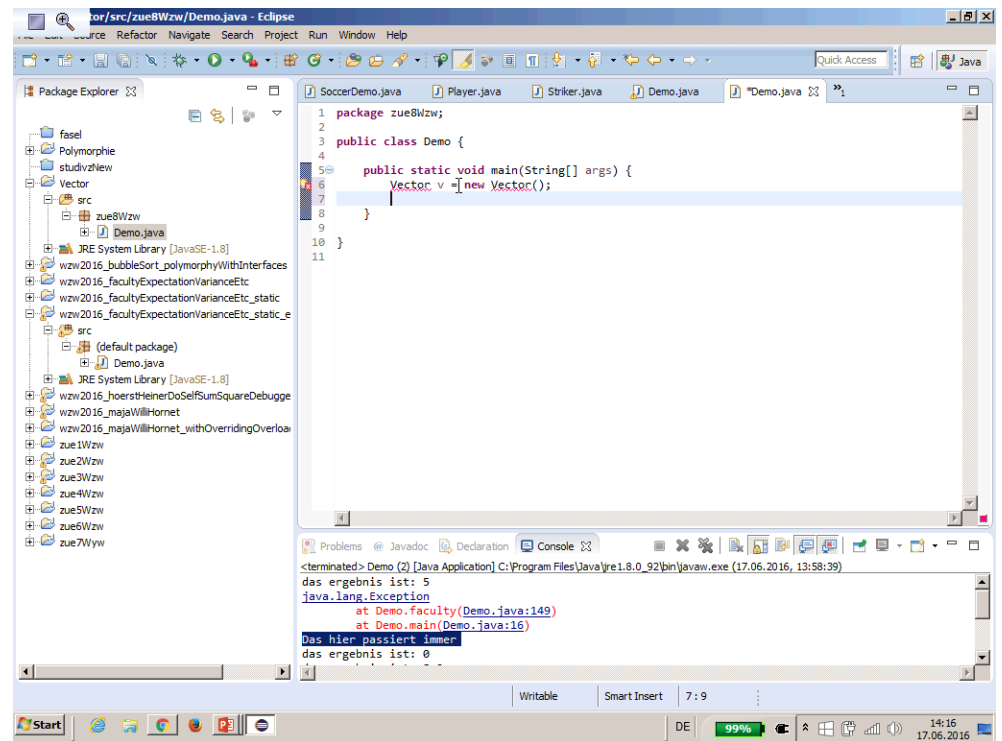
statt:

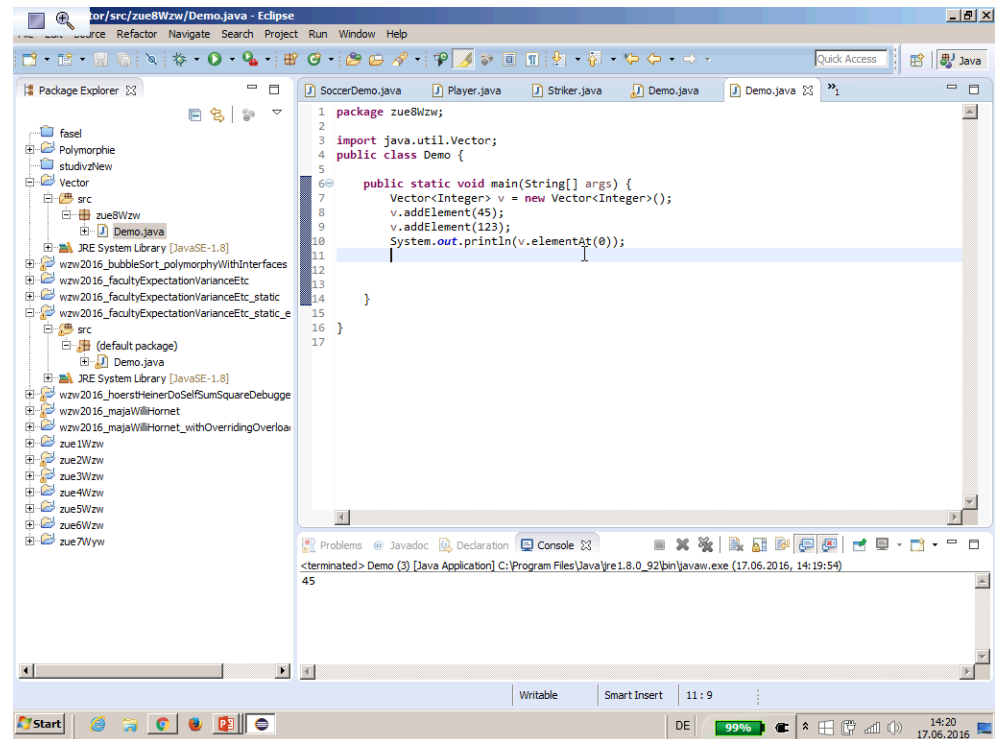
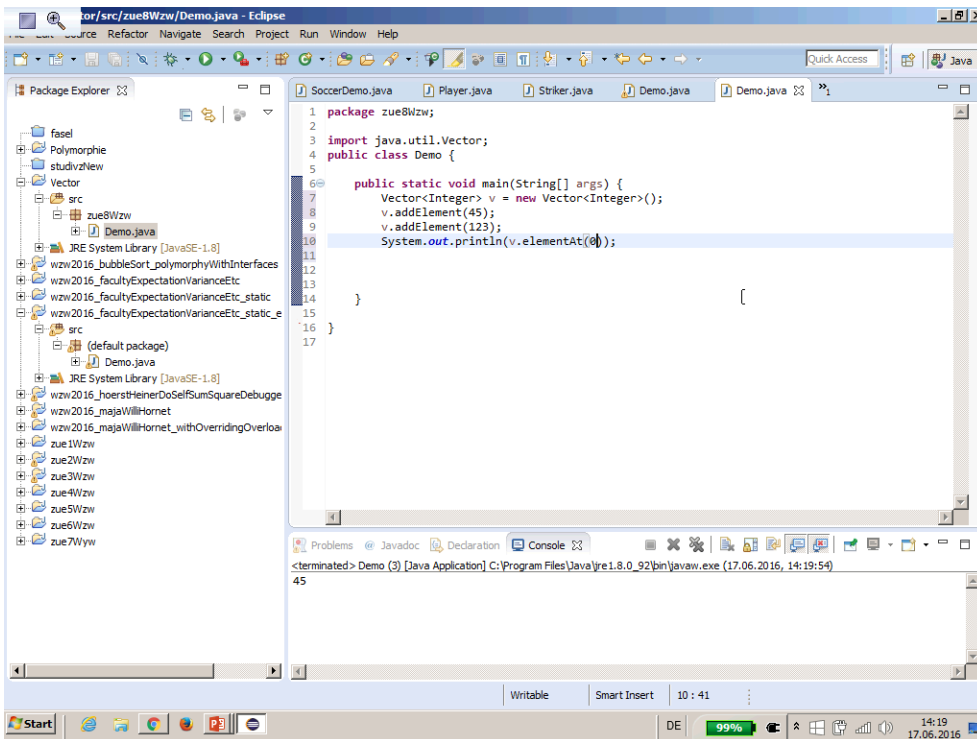
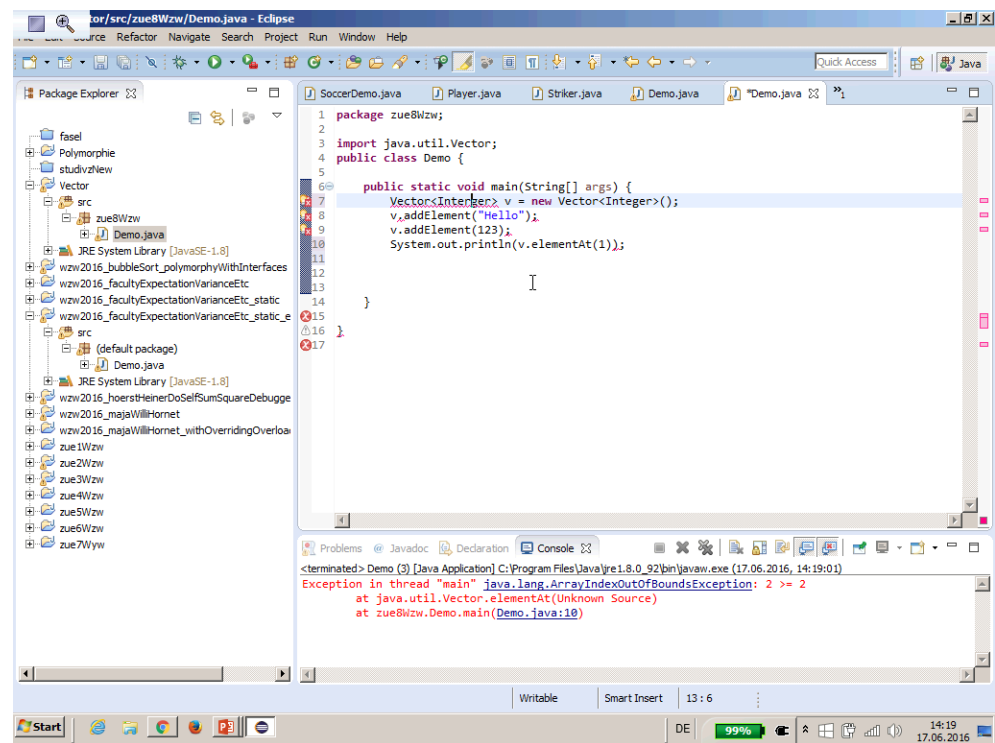
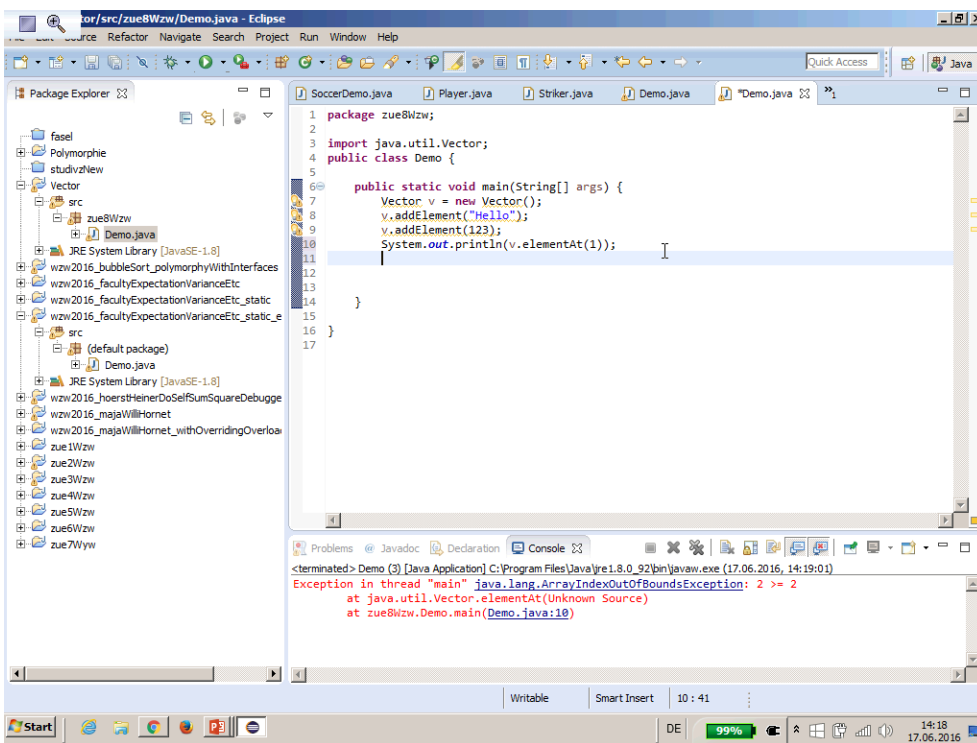
```
Integer someInteger = new Integer(723);
```

kann man schreiben:

```
Integer someInteger = 723;
```

179





```

1 package zue8Wzw;
2
3 import java.util.Vector;
4 public class Demo {
5
6     public static void main(String[] args) {
7         Vector<Integer> v = new Vector<Integer>();
8         v.addElement(45);
9         v.addElement(123);
10        System.out.println(v.elementAt(0));
11        for (Integer element: v){
12            System.out.println(element);
13        }
14    }
15 }
16
17 }
18
19 }
20

```

Console output: 45, 45, 123

```

1 package zue8Wzw;
2
3 import java.util.Vector;
4 public class Demo {
5
6     public static void main(String[] args) {
7         Vector<Integer> v = new Vector<Integer>();
8         v.addElement(45);
9         v.addElement(123);
10        System.out.println(v.elementAt(0));
11        for (Integer element: v){
12            System.out.println(element);
13        }
14    }
15 }
16
17 }
18
19 }
20

```

Console output: <terminated> Demo (3) [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (17.06.2016, 14:21:02)

## Lösen eines Beispiel-Problems

- Bsp.-Problem: Ein Bild aus dem Web laden und in einem Fenster darstellen
- Wie löse ich das Problem?
  1. Googlen
  2. Java Tutorial zu Rate ziehen
  3. Entsprechende Klassen, Methoden, Attribute in API-Doc nachschlagen
  4. Programm schreiben, testen
  5. Wenns nicht klappt : Gehe zu (1)

## Lösen eines Beispiel-Problems

- Bsp.-Problem: Ein Bild aus dem Web laden und in einem Fenster darstellen
- Wie löse ich das Problem?
  1. Googlen
  2. Java Tutorial zu Rate ziehen
  3. Entsprechende Klassen, Methoden, Attribute in API-Doc nachschlagen
  4. Programm schreiben, testen
  5. Wenns nicht klappt : Gehe zu (1)

# Lösen eines Beispiel-Problems

