Bereich: Java 8

Arbeiten mit Streams	**Musterlösung**
Package: `de.dhbwka.java.exercise.java8.soccer`	**Klasse:** `Soccer`

```java
package de.dhbwka.java.exercise.java8.soccer;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2018 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.0
 */
public class Player {

   /**
    * Number of player
    */
   private final int number;

   /**
    * Name of player
    */
   private final String name;

   /**
    * Position of player
    */
   private final String position;

   /**
    * Birthday of player
    */
   private final String birthday;

   /**
    * Club of player
    */
   private final String club;

   /**
    * Games of player
    */
   private final int games;

   /**
    * Goals of player
    */
   private final int goals;
```

```java
/**
 * Create player
 *
 * @param number number of the player
 * @param name name of the player
 * @param position position of the player
 * @param birthday birthday of the player
 * @param club club of the player
 * @param games games of the player
 * @param goals goals of the player
 */
public Player( int number, String name, String position, String birthday,
      String club, int games, int goals ) {
   super();
   this.number = number;
   this.name = name;
   this.position = position;
   this.birthday = birthday;
   this.club = club;
   this.games = games;
   this.goals = goals;
}

/**
 * Get the number of the player
 * @return number of the player
 */
public int getNumber() {
   return this.number;
}

/**
 * Get the name of the player
 * @return name of the player
 */
public String getName() {
   return this.name;
}

/**
 * Get the position of the player
 * @return position of the player
 */
public String getPosition() {
   return this.position;
}

/**
 * Get the birthday of the player
 * @return birthday of the player
 */
public String getBirthday() {
   return this.birthday;
}
```

```
    /**
     * Get the club of the player
     * @return club of the player
     */
    public String getClub() {
        return this.club;
    }

    /**
     * Get the games of the player
     * @return games of the player
     */
    public int getGames() {
        return this.games;
    }

    /**
     * Get the goals of the player
     * @return goals of the player
     */
    public int getGoals() {
        return this.goals;
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public String toString() {
        return String.format( "%3d", this.number ) + " | " + this.name + ", "
                + this.position + ", " + this.birthday + ", " + this.club + ", "
                + this.games + " games, " + this.goals + " goals";
    }

}
```

```
package de.dhbwka.java.exercise.java8.soccer;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Comparator;
import java.util.List;
import java.util.Objects;
import java.util.stream.Collectors;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2018 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.0
 */
```

```java
public class Soccer {

   /**
    * Application entry point
    *
    * @param args
    *           command line arguments
    */
   public static void main( String[] args ) {
      try {
         Path path = Paths.get( "33_Java8_Aufgaben_TeamDE.txt" );

         List<Player> players =
               Files.readAllLines( path, StandardCharsets.UTF_8 )
                     .stream().map( Soccer::parsePlayer )
                     .filter( Objects::nonNull )
                     .collect( Collectors.toList() );

         System.out.println( "Players sorted by number:" );
         players.stream().sorted( Soccer::comparePlayerByNumber )
               .forEach( System.out::println );

         System.out.println( "-----" );
         System.out
               .println( "Players with more than 50 games, sorted by name:" );
         players.stream().filter( p -> p.getGames() > 50 )
               .sorted( Soccer::comparePlayerByName )
               .forEach( System.out::println );

         System.out.println( "-----" );
         System.out.println( "All clubs of the players:" );
         players.stream().map( Player::getClub ).distinct()
         .forEach( System.out::println );

         System.out.println( "-----" );
         System.out.println( "Count of players with less than 5 goals: "
               + players.stream().filter( p -> p.getGoals() < 5 ).count() );
         System.out.println( "Count of goals of all players: "
               + players.stream().mapToInt( Player::getGoals ).sum() );
      } catch ( IOException e ) {
         e.printStackTrace();
      }
   }



   // continued on next page
```

```
    /**
     * Parse player from CSV line
     *
     * @param line
     *           line to parse
     * @return created player instance
     */
    public static Player parsePlayer( String line ) {
       String[] p = line.split( ";" );

       if ( p.length == 7 ) {
          return new Player( Integer.parseInt( p[0] ), p[1], p[2], p[3], p[4],
                Integer.parseInt( p[5] ), Integer.parseInt( p[6] ) );
       } else {
          return null;
       }
    }

    /**
     * Compare player by number
     *
     * @param p1
     *           player 1
     * @param p2
     *           player 2
     * @return result for {@link Comparator#compare(Object, Object) comparison}
     *         of number of the players
     */
    public static int comparePlayerByNumber( Player p1, Player p2 ) {
       return p1.getNumber() - p2.getNumber();
    }

    /**
     * Compare player by name
     *
     * @param p1
     *           player 1
     * @param p2
     *           player 2
     * @return result for {@link Comparator#compare(Object, Object) comparison}
     *         of name of the players
     */
    public static int comparePlayerByName( Player p1, Player p2 ) {
       return p1.getName().compareTo( p2.getName() );
    }

}
```

Bereich: Java 8

Zahlenraten (3)	**Musterlösung**
Package: `de.dhbwka.java.exercise.java8`	**Klasse:** `NumberGuessJava8`

```java
package de.dhbwka.java.exercise.java8;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.io.FileWriter;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Optional;
import java.util.StringTokenizer;
import java.util.function.Consumer;
import java.util.stream.Stream;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2018 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.0
 */
@SuppressWarnings( "serial" )
public class NumberGuessJava8 extends JFrame {

   private int numberToGuess;
   private int countAttempts;
   private int limit = 1000;

   private JButton btnExit = new JButton( "Exit" );
   private JButton btnOk = new JButton( "OK" );
   private JButton btnNew = new JButton( "New Game" );
   private JButton btnStat = new JButton( "Best Player" );
   private JTextField txtName = new JTextField( "Name", 20 );

   private JTextField txtGuess = new JTextField( 10 );
   private JTextField txtOutput = new JTextField( 40 );

   private String statFileName = "stat.txt";



   // continued on next page
```

```
public NumberGuessJava8() {
   super( "Number Guessing Game" );

   JPanel panName = new JPanel();
   JPanel panNumberinput = new JPanel();
   JPanel panButtons = new JPanel();
   JPanel panOutput = new JPanel();

   panName.add( new JLabel( "Player Name" ) );
   panName.add( this.txtName );

   panNumberinput.add(
         new JLabel( "Enter number beween 1 and " + this.limit ) );
   panNumberinput.add( this.txtGuess );

   // add all four buttons by streaming
   // and invoke panButtons.add for all of them
   Stream.of( this.btnNew, this.btnOk, this.btnStat, this.btnExit )
         .forEach( panButtons::add );

   panOutput.add( this.txtOutput );

   this.setLayout( new GridLayout( 4, 1 ) );

   // add all four panels by streaming
   // and invoke this.add for all of them
   Stream.of( panName, panNumberinput, panButtons, panOutput )
         .forEach( this::add );

   this.addEventHandling();
   this.createRandomNumber();

   this.setSize( 500, 250 );
   this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   this.setVisible( true );
}

/**
 * Add event handling and use lambda expressions to do so
 */
public void addEventHandling() {
   this.btnNew.addActionListener( e -> this.createRandomNumber() );
   this.btnExit.addActionListener( e -> System.exit( 0 ) );

   // okActionListener matches signature of ActionListener#actionPerformed
   this.txtGuess.addActionListener( this::okActionListener );
   this.btnOk.addActionListener( this::okActionListener );

   this.btnStat.addActionListener( e -> this.showBestPlayer() );
}



// continued on next page
```

```
public void okActionListener( ActionEvent event ) {
   try {
      int guess = Integer.parseInt( this.txtGuess.getText() );
      this.countAttempts++;
      this.txtGuess.setText( "" );

      // since all texts for output are being constructed from
      // this.countAttempts and guess, for correct inputs,
      // it is possible to define a consumer which just takes
      // the string to pass to String.format and then sets the
      // correct text for this.txtOutput
      Consumer<String> outputSetter = s -> {
         this.txtOutput
               .setText( String.format( s, this.countAttempts, guess ) );
      };

      if ( guess > this.numberToGuess ) {
         outputSetter.accept( "Attempt #%s: %s => too big!" );
      }

      else if ( guess < this.numberToGuess ) {
         outputSetter.accept( "Attempt #%s: %s => too small!" );
      }

      else {
         outputSetter.accept( "Attempt #%s: %s => correct!!! New Game!" );
         this.writeStatFile();
         this.createRandomNumber();
      }
   } catch ( NumberFormatException nfe ) {
      this.txtOutput.setText( "Bad input!" );
   }
}

private void writeStatFile() {
   try ( FileWriter f = new FileWriter( this.statFileName, true ) ) {
      String name = this.txtName.getText();
      f.write( name + " " + this.countAttempts + " attempts\n" );
   } catch ( Exception e ) {
   }
}


// continued on next page
```

```
/**
 * Show best player and use NIO + streaming to find minimum attempts
 */
private void showBestPlayer() {
   try {
      Optional<PlayerHelper> bestPlayer =
            // read all lines as list
            Files.readAllLines( Paths.get( this.statFileName ) )
                  // stream this list of strings
                  .stream()
                  // Convert string line to player helper object,
                  .map( PlayerHelper::new )
                  // Stream now contains these helper object items,
                  // no longer strings!
                  // => Find minimum attempts by comparing
                  .min( ( a, b ) -> Integer.compare( a.getAttempts(),
                        b.getAttempts() ) );

      if ( bestPlayer.isPresent() ) {
         PlayerHelper player = bestPlayer.get();
         this.txtOutput
               .setText( "Best Player: " + player.getName() + ",  "
                     + player.getAttempts() + " attempts" );
      }

   } catch ( Exception ex ) {
   }
}

private void createRandomNumber() {
   this.txtGuess.setText( "" );
   this.txtOutput.setText( "New Game!" );
   this.numberToGuess = (int) (Math.random() * this.limit + 1);
   this.countAttempts = 0;

   // For debugging purposes
   System.out.println( "Number to guess: " + this.numberToGuess );
}

public static void main( String args[] ) {
   new NumberGuessJava8();
}
```

// continued on next page

```
   /**
    * Helper class to store name and points separately to better utilize
    * streaming features
    */
   private class PlayerHelper {

      private String name;

      private int attempts = Integer.MAX_VALUE;

      public PlayerHelper( String line ) {
         StringTokenizer st = new StringTokenizer( line );
         this.name = st.nextToken();
         try {
            this.attempts = Integer.parseInt( st.nextToken() );
         } catch ( Exception e ) {
         }
      }

      public int getAttempts() {
         return this.attempts;
      }

      public String getName() {
         return this.name;
      }
   }
}
```