Bereich: Probe-Programmentwurf

DartsCounter — Musterlösung

Package: `de.dhbwka.java.exercise.dartscounter`

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

/**
 * Field on the dart board
 */
public class Field {

   /**
    * Label (e.g. <code>D12</code>)
    */
   private final String label;

   /**
    * Value
    */
   private final int value;

   /**
    * Flag to indicate if field is a double field
    */
   private final boolean doubleField;

   /**
    * Create field
    *
    * @param label
    *            label of field
    * @param value
    *            points
    * @param doubleField
    *            double field flag
    */
   public Field( String label, int value, boolean doubleField ) {
      super();
      this.label = label;
      this.value = value;
      this.doubleField = doubleField;
   }

   /**
    * Get the label
    *
    * @return label
    */
   public String getLabel() {
      return this.label;
   }
```

```
   /**
    * Determine if this field is a double field
    *
    * @return <code>true</code> if field is a double field,
    * <code>false</code> otherwise
    */
   public boolean isDoubleField() {
      return this.doubleField;
   }

   /**
    * Get the points / value of this field
    *
    * @return points
    */
   public int getValue() {
      return this.value;
   }

}
```

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

/**
 * Dartboard with all fields (1-20 with single, double and triple,
 * Single Bull (25) + BULL and helper field for missed throws)
 */
public class Board {

   /**
    * All fields of the board
    */
   private final Field[] fields = new Field[20 * 3 + 2 + 1];

   /**
    * Create the board
    */
   public Board() {
      int value = 1;
      int i = 0;

      // Helper field for missed throw
      this.fields[i++] = new Field( "x", 0, false );

      while ( value <= 20 ) {
         this.fields[i++] = new Field( "" + value, value, false );
         this.fields[i++] = new Field( "D" + value, value * 2, true );
         this.fields[i++] = new Field( "T" + value, value * 3, false );
         value++;
      }
      this.fields[i++] = new Field( "25", 25, false );
      this.fields[i++] = new Field( "BULL", 50, true );
   }
```

```
   /**
    * Parse field
    *
    * @param label
    *            label value to parse (e.g. <code>T10</code>)
    * @return parsed field or <code>null</code> if none found
    */
   public Field parseField( String label ) {
      for ( final Field s : this.fields ) {
         if ( s.getLabel().equalsIgnoreCase( label ) ) {
            return s;
         }
      }
      return null;
   }

}
```

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

/**
 * Visit at the board, means (at most) 3 fields
 */
public class Visit {

   /**
    * Fields thrown
    */
   private final Field[] fields;

   /**
    * Create visit instance
    *
    * @param fields
    *            thrown fields
    * @throws IllegalArgumentException
    *             if fields count is greater than 3
    */
   public Visit( Field[] fields ) {
      if ( fields.length > 3 ) {
         throw new IllegalArgumentException( "Invalid count of fields!" );
      }
      this.fields = fields;
   }

   /**
    * Get the fields
    *
    * @return fields
    */
   public Field[] getFields() {
      return this.fields;
   }
```

```
   /**
    * Get the value
    *
    * @return value
    */
   public int getValue() {
      int sum = 0;
      for ( final Field s : this.fields ) {
         if ( s != null ) {
            sum += s.getValue();
         }
      }
      return sum;
   }

   /**
    * Get the last thrown field
    *
    * @return last thrown field
    */
   public Field getLastField() {
      if ( this.fields.length > 0 ) {
         return this.fields[this.fields.length - 1];
      }
      return null;
   }

}
```

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

/**
 * Player
 */
public class Player {

   /**
    * Visits
    */
   private final Visit[] visits = new Visit[10];

   /**
    * Count of thrown darts
    */
   private int countDartsThrown = 0;

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

```java
/**
 * Create player
 *
 * @param name
 *            name of player
 */
public Player( String name ) {
   super();
   this.name = name;
}

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

/**
 * Add a visit
 *
 * @param visit
 *            visit to add
 * @return <code>true</code> if visit was added, <code>false</code> otherwise
 */
public boolean addVisit( Visit visit ) {
   final boolean doubleOut = true; // extension

   final int result = this.getRemainingPoints() - visit.getValue();

   // Less than zero points or 1 point is invalid!
   if ( result < 0 || doubleOut && result == 1 ) {
      return false;
   }

   // Result would be zero, check for double field if required
   // by double out flag
   if ( doubleOut && result == 0 && !visit.getLastField().isDoubleField() ) {
      return false;
   }

   this.countDartsThrown += visit.getFields().length;

   // Add visit at last position
   for ( int i = 0; i < this.visits.length; i++ ) {
      if ( this.visits[i] == null ) {
         this.visits[i] = visit;
         return true;
      }
   }
   return false;
}
```

```
   /**
    * Get the count of thrown darts
    *
    * @return count of thrown darts
    */
   public int getCountDartsThrown() {
      return this.countDartsThrown;
   }

   /**
    * Get the remaining points
    *
    * @return remaining points
    */
   public int getRemainingPoints() {
      int points = Game.POINTS;
      for ( final Visit s : this.visits ) {
         if ( s != null ) {
            points -= s.getValue();
         }
      }
      return points;
   }

}
```

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

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

/**
 * Game class
 */
public class Game {

   /**
    * Points to be scored
    */
   public final static int POINTS = 501;

   /**
    * Global scanner instance
    */
   private static Scanner SCANNER = new Scanner( System.in );

   /**
    * Darts board to use
    */
   private final Board board;
```

```
/**
 * Players
 */
private final Player[] players;

/**
 * Checkouts table
 */
private final String[] checkouts;

/**
 * Create the game
 *
 * @param board
 *            board instance
 * @param players
 *            players
 */
public Game( Board board, Player[] players ) {
   this.players = players;
   this.board = board;

   this.checkouts = this.readCheckouts();
}

/**
 * Read the checkout table
 *
 * @return checkouts
 */
private String[] readCheckouts() {
   final String[] checkouts = new String[170];
   try ( BufferedReader reader = new BufferedReader(
                                        new FileReader( "checkouts.txt" ) ) ) {
      int index = 0;
      String line = null;
      while ( (line=reader.readLine()) != null && index < checkouts.length ) {
         checkouts[index++] = line;
      }
   }
   catch ( final Exception e ) {
      e.printStackTrace();
   }
   return checkouts;
}
```

```
/**
 * Start the game
 */
public void start() {
   Player winner = null;
   int visitCount = 0;

   while ( winner == null && visitCount < 10 ) {
      for ( final Player p : this.players ) {
         final Visit v = this.scanVisit( p );
         if ( p.addVisit( v ) ) {
            System.out.println( "Scored: " + v.getValue() );
            if ( p.getRemainingPoints() == 0 ) {
               System.out.println("\nGame shot and the leg, " +
                                                      p.getName() + "!" );
               winner = p;
               break; // breaks only for loop!
            }
         }
         else {
            System.out.println( "No score!" );
         }
         System.out.println( "======================" );
      }
      visitCount++;

   }

   if ( winner == null ) {
      System.out.println( "\nYou're too bad for this game!" );
   }
   else {
      try ( BufferedWriter bw = new BufferedWriter(
                                  new FileWriter( "highscore.txt", true ) ) ) {
         bw.write( winner.getName() + " won with " +
                                    winner.getCountDartsThrown() + " darts." );
      }
      catch ( final IOException e ) {
         e.printStackTrace();
      }
   }

}
```

```
   /**
    * Scan a visit to the board
    *
    * @param p
    *            player to scan for
    * @return scanned visit instance
    */
   public Visit scanVisit( Player p ) {
      final int remaining = p.getRemainingPoints();

      System.out.println( "Player: " + p.getName() + ", " +
                                              remaining + " points remaining." );
      if ( remaining <= 170 ) {

         final String checkout = this.checkouts[remaining - 1];
         if ( checkout != null && !"-".equals( checkout ) ) {
            System.out.println( "Possible checkout: " + checkout );
         }
      }

      System.out.print( "Enter visit: " );
      final String line = Game.SCANNER.nextLine();
      final String[] fieldStrings = line.split( " " );

      final Field[] fields = new Field[fieldStrings.length];
      for ( int i = 0; i < fieldStrings.length; i++ ) {
         fields[i] = this.board.parseField( fieldStrings[i] );
         // check if field string was valid, if not => exception
         if ( fields[i] == null ) {
            throw new IllegalArgumentException( fieldStrings[i] + "
                                           is not a valid input for a field!" );
         }
      }

      return new Visit( fields );
   }

}
```

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

/**
 * Darts counter application (provided)
 */
public class DartsCounter {

   /**
    * Application entry point
    *
    * @param args
    *            command line arguments
    */
   public static void main( String[] args ) {

      final Board b = new Board();
```

```
        final Player[] players = new Player[] {
                    new Player( "Michael van Gerwen" ),
                    new Player( "Rob Cross" )
        };

        final Game g = new Game( b, players );
        g.start();

    }

}
```

Inhalt der Datei „checkouts.txt“

```
-
D1
1 D1
D2
1 D2
D3
1 D3
D4
1 D4
D5
1 D5
D6
1 D6
D7
1 D7
D8
1 D8
D9
1 D9
D10
1 D10
D11
3 D10
D12
5 D10
6 D10
7 D10
D14
9 D10
D15
1 D15
D16
1 D16
D17
3 D16
D18
5 D16
6 D16
7 D16
D20
1 D20
2 D20
3 D20
4 D20
5 D20
6 D20
7 D20
16 D16
9 D20
10 D20
11 D20
12 D20
13 D20
14 D20
15 D20
16 D20
17 D20
18 D20
19 D20
20 D20
25 D18
T10 D16
T13 D12
T16 D8
25 D20
T10 D18
T17 D8
T20 D4
T13 D15
T10 D20
T17 D10
T12 D18
T19 D8
T18 D10
T17 D12
T20 D8
T19 D10
T18 D12
T13 D20
T20 D10
T15 D18
T14 D20
T17 D16
T20 D12
T15 D20
```

```
T18 D16
T17 D18
T16 D20
T19 D16
T18 D18
T17 D20
T20 D16
T19 D18
T18 D20
T19 D19
T20 D18
T19 D20
T20 D19
T20 7 D16
T20 D20
T17 BULL
T20 10 D16
T20 3 D20
T18 BULL
T20 13 D16
T20 10 D18
T19 BULL
T20 16 D16
T20 17 D16
T20 18 D16
T20 19 D16
T20 20 D16
T20 13 D20
T20 14 D20
T20 15 D20
T20 16 D20
T20 17 D20
T20 18 D20
T20 19 D20
T20 20 D20
T20 T7 D20
T18 T12 D16
T19 T10 D18
T20 14 BULL
25 T20 D20
T19 19 BULL
T20 T17 D8
T18 T14 D16
T19 T12 D18
T20 20 BULL
T20 T17 D10
T20 T12 D18
T20 T11 D20
T20 T14 D16
T17 T20 D12
T20 T20 D8
T19 T20 D10
T20 T18 D12
T19 T14 D20
T20 T20 D10
T20 T19 D12
T20 T14 D20
T20 T17 D16
T20 T20 D12
T20 T15 D20
T20 T18 D16
T19 T18 D18
T20 T16 D20
T20 T19 D16
T20 T18 D18
T20 T17 D20
T20 T20 D16
T20 T19 D18
T20 T18 D20
T20 T19 D19
T20 T20 D18
T20 T19 D20
T20 T20 D19
-
T20 T20 D20
T20 T17 BULL
-
-
T20 T18 BULL
-
-
T20 T19 BULL
-
-
T20 T20 BULL
```