Technische Hochschule Ingolstadt

Studiengang Wirtschaftsinformatik

***** Lösungsvorschlag *****

# Prüfung
# Grundlagen der Programmierung 2

| Aufgabe | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Summe |
|---|---|---|---|---|---|---|---|---|
| Punkte (max.) | (25) | (25) | (25) | (25) | | | | (100) |

| Note: |
|---|
| |

Prüfer: H.-M. Windisch
Prüfungsdauer: 90 Min
Hilfsmittel: keine

| Studiengang | Dozent | Matrikelnummer | Fachsem. | Raum | Platz |
|---|---|---|---|---|---|
| | | | | | |

Dieses Geheft enthält sowohl die Aufgabenstellung als auch den Platz für Ihre Antworten. Schreiben Sie möglichst nur in die vorgegebenen Antwortrahmen. Geben Sie am Ende der Prüfung das vollständige Geheft wieder ab.

Füllen Sie die erste Seite noch vor der Prüfung mit Ihren persönlichen Angaben aus.

Die Darbietung Ihrer Aufgabenbearbeitung muss gut lesbar, folgerichtig und verständlich sein.

Viel Erfolg!

Task 1 (Various questions: ca. 25%)

Subtask a)

Given the folowwing statement:

```
button.setOnAction(event -> {
    Collections.sort(liste,
      (Person p1, Person p2) -> -p1.getName().compareTo(p2.getName()));
    listView.setItems(liste);
});
```

Formulate the above statement without Lamba expressions (8 P):

```
button.setOnAction(new EventHandler<ActionEvent>() {                    1
    public void handleEvent(ActionEvent event) {                        1
        Collections.sort(liste, new Comparator<Person>() {              2
            public int compare(Person p1, Person p2) {                  1
                return -p1.getName().compareTo(p2.getName());           2
            }
        });
        listeView.setItems(liste);                                      1
    }
}
```

| Sum | 8 |
|---|---|

Subtask b)

Given the following statements:

```
ObservableList<Person> liste = FXCollections.observableArrayList();
final int maxLaenge = Integer.valueOf(laengeFeld.getText());
personenListe.stream()
.filter(p -> p.getName().length() <= maxLaenge)
.forEach(p -> { liste.add(p); System.out.println(p); });
```

Formulate the above instructions without Lamba expressions or streams (5 P):

```
for (Person p : personenListe) {                                        2
    if (p.getName().length() <= maxLaenge) {                            1
      liste.add(p);                                                     1
      System.out.println(p);                                            1
    }
}
```

| Summe | 5 |
|---|---|

Subtask c)

The following instructions for starting a thread are given:

```
Thread t = new Thread(() -> {
    for (int i = 0; i < 1000; ++i) {
      System.out.println("i=" + i);
    }
});
t.start();
```

Formulate the above instructions without a lamba expression and define a separate thread class so that the thread instance can be stopped by the creator. Call the corresponding method after starting the thread. (12 P)

Thread Class:

| | |
|---|---|
| `class MeinThread extends Thread {` | 2 |
| `    private boolean terminiert = false;` | 1 |
| `    public void run() {` | 1 |
| `        for (int i = 0; i < 1000 && !terminiert; ++i) {` | 2 |
| `          System.out.println("i=" + i);` | 1 |
| `        }` | |
| `    }` | |
| `    public void terminiere() {` | 1 |
| `        terminiert = true;` | 1 |
| `    }` | |
| `}` | |
| **Summe** | **9** |

Creation, start and termination of the thread (e.g. in main()):

| | |
|---|---|
| `MeinThread t = new MeinThread();` | 1 |
| `t.start();` | 1 |
| `t.terminiere();` | 1 |
| Sum | 3 |

Task 2 (Collections: ca. 25%)

The following classes for managing football tournaments are given. Getter and setter methods can be assumed to be given.

```
/* Result of a match */
class SpielErgebnis {
      private int toreGegner1, toreGegner2;
      public SpielErgebnis(int toreGegner1, int toreGegner2) { ... }
}

/* Describes a team with country, score and goal difference */
class Mannschaft {
      private String land; // z.B. „Deutschland“
      private int punkte;
      private int tordifferenz; // z.B. 10:3 Tore => Tordifferenz 7 (=10-3)

      public Mannschaft(String land) { ... }
      public void addToTordifferenz(int wert) {
            this.tordifferenz += wert;
      }
}

/* Describes a match between two teams with a result */
class Match {
      private Mannschaft gegner1, gegner2;
      private SpielErgebnis ergebnis;
      private char gruppe; // Spielgruppe, z.B. 'A'

      public Match(char g, Mannschaft m1, Mannschaft m2, SpielErgebnis e) {
          ... }
}

/* Tournament with name, year and a list of matches */
class Turnier {
      private String name;        // Nameof tournament
      private int jahr;           // Year in which tournament takes place
      private List<Match> spiele; // all matches

      public Turnier(String name, int jahr) {
            this.name = name;  this.jahr = jahr;
            this.spiele = new ArrayList<>();
      }
      public void druckeBegegnungen() {
          ... /* see subtask 2a) */ }
      public List<Mannschaft> tabelle() {
          ... /* see subtask 2b) */ }
}
```

Subtask a)

The printMatches method prints the matches of the tournament to the console.

The following is an example:

```
Begegnungen der WM 2116
B:       Spanien -       Ungarn: 1:1
B:    Frankreich -     Portugal: 3:4
B:       Spanien -     Portugal: 2:1
A:  Deutschland -       Türkei: 1:4
A:       England -     Russland: 3:5
A:  Deutschland -     Russland: 2:5
B:       Spanien -   Frankreich: 2:2
B:        Ungarn -     Portugal: 0:1
B:        Ungarn -   Frankreich: 3:2
A:  Deutschland -      England: 4:2
A:        Türkei -     Russland: 1:0
A:        Türkei -      England: 3:5
```

Specify the method druckeBegegnungen(). (6 P)

```
public void druckeBegegnungen() {

    System.out.println("Begegnungen der " + name + " " + jahr);
    for (Match m : spiele) {
      System.out.printf("%c: %12s - %12s: %d:%d\n",
          m.getGruppe(), m.getGegner1().getLand(),
          m.getGegner2().getLand(), m.getErgebnis().getToreGegner1(),
          m.getErgebnis().getToreGegner2());
    }




}
```

1
1
2
2
(alle Param.)

6

Subtask b)

The table method returns an unsorted list of the teams in the tournament. The resulting score and goal difference are calculated for each team. The usual sorting by points/goal difference is done later in subtask 3a)!

A main programme and the generated output are shown below:

```
public static void main(String[] args) {
      Turnier turnier = initTurnierdaten(); // Turnierdaten erzeugen
      System.out.println("\n\tMannschaft\tPunkte\tTorverhältnis");
      int i = 1;
      for (Mannschaft m : turnier.tabelle())
        System.out.printf("%d.\t%-12s\t%d\t%d\n", i++,
            m.getLand(), m.getPunkte(), m.getTorverhaeltnis());
}
```

Output (still unsorted, since the comparator is missing!):

```
       Mannschaft  Punkte      Torverhältnis
1.     Spanien           5     1
2.     Ungarn            4     0
3.     Frankreich        1     -2
4.     Portugal          6     1
5.     Deutschland       3     -4
6.     Türkei            6     2
7.     England           3     -2
8.     Russland          6     4
```

Specify the method tabelle(). (19 P)

| | |
|---|---|
| `public List<Mannschaft> tabelle() {` | |
| `List<Mannschaft> ergebnis = new ArrayList<>();` | 1 |
| `for (Match m : spiele) {` | 1 |
| `SpielErgebnis erg = m.getErgebnis();` | 2 für |
| `Mannschaft m1 = m.getGegner1();` | alle |
| `Mannschaft m2 = m.getGegner2();` | Var. |
| `int tore1 = erg.getToreGegner1(),` | |
| `tore2 = erg.getToreGegner2();` | |
| `if (tore1 == tore2) {` | 1 |
| `m1.setPunkte(m1.getPunkte() + 1);` | 1 |
| `m2.setPunkte(m2.getPunkte() + 1);` | 1 |
| `} else if (tore1 > tore2) {` | 1 |
| `m1.setPunkte(m1.getPunkte() + 3);` | 1 |
| `m1.addToTorverhaeltnis(tore1 - tore2);` | 1 |
| `m2.addToTorverhaeltnis(tore2 - tore1);` | 1 |
| `} else {` | |
| `m2.setPunkte(m2.getPunkte() + 3);` | 1 |
| `m1.addToTorverhaeltnis(tore1 - tore2);` | 1 |
| `m2.addToTorverhaeltnis(tore2 - tore1);` | 1 |
| `}` | |
| `if (!ergebnis.contains(m1))` | 1 |
| `ergebnis.add(m1);` | 1 |
| `if (!ergebnis.contains(m2))` | 1 |
| `ergebnis.add(m2);` | 1 |
| `}` | |
| `return ergebnis;` | 1 |
| `}` | |
| Summe | 19 |

Task 3 (Interfaces: ca. 25%)

The classes from task 2 are given.

Subtask a) Comparator for sorting the teams in method table
The team list should now be sorted. To do this, the following lines are added to the method. Add the missing comparator definition!
(8 P.)

```
...
Comparator<Mannschaft> mannschaftsComparator =
/* Add your comparator definition here */


new Comparator<Mannschaft>() {                                          1
      public int compare(Mannschaft m1, Mannschaft m2) {                1
        if (m1.getPunkte() == m2.getPunkte()) {                         1
          return m2.getTorverhaeltnis() - m1.getTorverhaeltnis();       2
        } else if (m1.getPunkte() < m2.getPunkte())                     1
          return 1;                                                     1
        else
          return -1;                                                    1
        }
      };



Collections.sort(<result variable>, mannschaftsComparator);
```

| Summe | 8 |
|---|---|

Subtask b) Iterator for tournaments
The matches of a tournament should now be displayed alphabetically in ascending order by group. For this purpose, an iterator is to be developed that makes it possible to iterate over tournament objects:

In main:

```
...
for (Match m : turnier)
  System.out.printf("%c: %12s - %12s\n",
    m.getGruppe(),
    m.getGegner1().getLand(),
    m.getGegner2().getLand());
...
```

Output:

```
A:  Deutschland -       Türkei
A:      England -     Russland
A:  Deutschland -     Russland
A:  Deutschland -      England
A:       Türkei -     Russland
A:       Türkei -      England
B:      Spanien -       Ungarn
B:    Frankreich -     Portugal
B:      Spanien -     Portugal
B:      Spanien -   Frankreich
B:       Ungarn -     Portugal
B:       Ungarn -   Frankreich
```

How to add to the Tournament class to make it iterable? (3 P.)

```
Implements:
class Turnier implements Iterable<Match> {

Method:
	public Iterator<Match> iterator() {
		return new TurnierIterator(spiele);
	}
```

1

1
1

| Sum | 3 |
|---|---|

Define the tournament iterator below: (14 P.)

```
class TurnierIterator implements Iterator<Match> {

	private List<Match> spiele;
	private int index = 0;


	// CTOR
	public TurnierIterator(List<Match> spiele) {
		this.spiele = spiele;
		Collections.sort(spiele, new Comparator<Match>() {
			public int compare(Match m1, Match m2) {
				return m1.getGruppe() - m2.getGruppe();
			}
		});
		index = 0;
	}

	public boolean hasNext() {
		return index < spiele.size();
	}

	public Match next() {
		if (hasNext()) {
			return spiele.get(index++);
		}
		return null;
	}




}
```

1
1
1
1
2
1
1
2
1

| Sum | 14 |
|---|---|

Task 4 (GUI: ca. 25 %)

A TicTacToe game is to be developed with JavaFX.

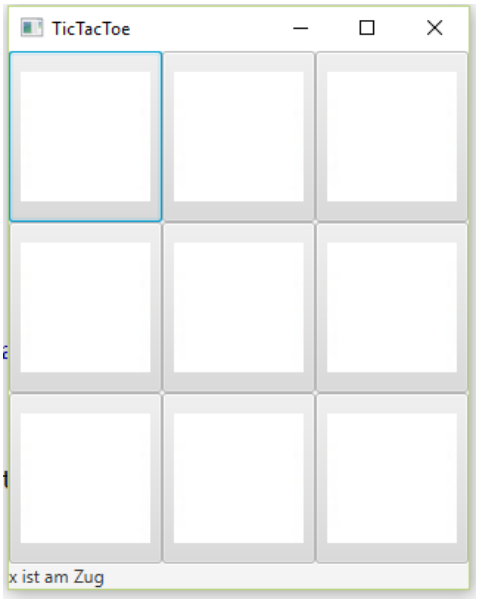


*Figure 3: Start of the game*

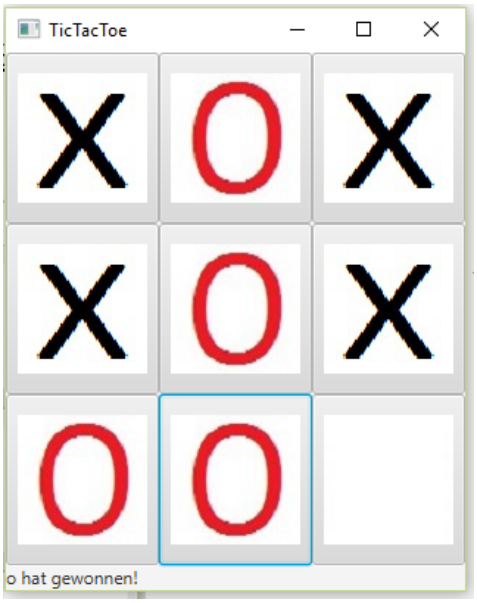


*Figure 4: x has won!*

The game works as follows:
- After the start, all fields are empty, X begins (see Figure 3).
- Each click on a square causes the square to be filled with the character of the player whose turn it is.
- After each click, the system checks whether the current move has decided the game, i.e. whether there are 3 identical characters horizontally, vertically or diagonally. If this is the case, a message is displayed (see Figure 4) and further button clicks are ignored (use the variable spielGewonnen, see below).
- The game is ended by clicking on the cross (top right-hand corner).

Use the predefined class TicTacToeButton to implement the buttons. It offers the following methods
- public TicTacToe(char char): Creates a button that is filled with an image according to the passed character value. Valid character values are 'x', 'o' and the space character. The latter causes the button to be displayed as a white area.
- Getter and setter for the "character" attribute

Subtask a) (14 P.)
Add a start method to the TicTacToe class given below, which sets up the game as shown above and displays it on the screen. The method gewonnen() can be taken as given.

Attention: the assignment of the variable buttonHandler should be made in subtask b)! However, it should be used in start().

```
public class TicTacToe extends Application {
      Label nachrichtLabel; // Message text
      private TicTacToeButton[][] buttons;
      private final int dimension = 3;
      private boolean spielGewonnen = false;
      private char zeichen = 'x';
      private EventHandler<ActionEvent> buttonHandler = ...;
             /* see subtask b) !!! */

      public static void main(String[] args) {
            launch(args);
      }


      // returns true if a winning position was created by the last move
      private boolean gewonnen() {
        // kann als gegeben vorausgesetzt werden!
      }

      public void start(Stage stage) {


      stage.setTitle("TicTacToe");                                          1
      nachrichtLabel = new Label("x ist am Zug");                           1
      buttons = new TicTacToeButton[dimension][dimension];                  1
      GridPane gridPane = new GridPane();                                   1

      for (int i = 0; i < dimension; ++i) {                                 1
            for (int j = 0; j < dimension; ++j) {                           1
                  buttons[i][j] = new TicTacToeButton(' ');                 1
                  buttons[i][j].setOnAction(buttonHandler);                 1
                  gridPane.add(buttons[i][j], j, i);                        1
            }
      }

      BorderPane mainPane = new BorderPane();                               1
      Scene scene = new Scene(mainPane);                                    1
      mainPane.setCenter(gridPane);                                         1
      mainPane.setBottom(nachrichtLabel);                                   1
      stage.setScene(scene);                                                ½
      stage.show();                                                         ½









      }
}
```

| Summe | 14 |
|---|---|

Subtask b) (11 P.)

Now add the event handling to implement the functionality of the TicTacToe game described above. To do this, assign a suitable lambda expression to the buttonHandler variable.

| | |
|---|---|
| `private EventHandler<ActionEvent> buttonHandler =` | |
| `event -> {` | 1 |
| `if (spielGewonnen)` | 1 |
| `    return;` | ½ |
| `TicTacToeButton button = (TicTacToeButton) event.getSource();` | 1 |
| `button.setZeichen(zeichen);` | 1 |
| `if (gewonnen()) {` | 1 |
| `    nachrichtLabel.setText(zeichen + " hat gewonnen!");` | 1 |
| `    spielGewonnen = true;` | 1 |
| `    return;` | ½ |
| `}` | |
| `if (zeichen == 'x')` | 1 |
| `    zeichen = 'o';` | ½ |
| `else` | |
| `    zeichen = 'x';` | ½ |
| `nachrichtLabel.setText(zeichen + " ist am Zug");` | 1 |
| `};` | |
| `;` | |
| Sum | 11 |