Prof. Dr. Christoph Bockisch
MSc. Teresa Dreyer
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Programmierpraktikum: Sommersemester 2025

Aufgaben Tag 1

Aufgabe 1: Setup (verpflichtend) — 5 Punkte

In diesem Block des Programmierpraktikums kommt ein Plugin zum Einsatz. Wir erfassen mit dem Plugin auf anonymer Basis Daten über die Verwendung der IDE, die wir zur Verbesserung der Lehre und für Forschungszwecke einsetzen möchten. Die Daten werden lokal gespeichert und am Ende des Praktikumsblocks von Ihnen exportiert und eingereicht. Sie werden in der Umfrage am Ende gefragt, ob Sie der Verwendung der Daten zu Forschungszwecken zustimmen. Das Plugin ist ein Prototyp. Es kann sein, dass es sich an einigen Stellen unerwartet verhält. Falls Probleme auftreten, wenden Sie sich bitte an Ihren Tutor oder Ihre Tutorin. Im Folgenden finden Sie die Instruktionen für den Setup.
Diese Aufgabe ist verpflichtend und muss erledigt und einem Tutor oder einer Tutorin vorgestellt worden sein, bevor Sie weiter machen können.
Die Folien von der Aufgabenvorstellung sind hochgeladen und beinhalten Screenshots, die Ihnen bei der Einrichtung helfen können.
Helfen Sie sich auch gerne gegenseitig, wenn Sie Hilfe benötigen oder anbieten können.

a) Installieren Sie die folgende Version von IntelliJ Idea: 2024.3.2 Den Download finden Sie auf dieser Webseite: `https://www.jetbrains.com/idea/download/other.html`. **1**
Überprüfen Sie, dass Sie die korrekte IntelliJ-Version installiert und geöffnet haben, indem Sie unter `Help>About` nachschauen. Hier sollte Version 2024.3.2 angezeigt werden.

b) Im Iliaskurs finden Sie eine zip-Datei mit dem Namen `pltLabPlugin`. Laden Sie diese herunter. **1**
Installieren Sie das pltLabPlugin. Dafür führen Sie die folgenden Schritte aus:

1. Navigieren Sie in IntelliJ nach `File>Settings>Plugins` (bzw. auf Deutsch `Datei>Einstellungen>Plugins`).
2. Klicken Sie dort auf das kleine Zahnrad und dann `Install Plugin from Disk` bzw. auf Deutsch `Plugin von der Festplatte installieren`.
3. Wählen Sie die heruntergeladene zip-Datei aus und installieren Sie diese mit `apply` (`anwenden`). Eventuell wird Ihnen eine Warnung über Third-Party-Plugins angezeigt. Klicken Sie bei dieser auf `OK`.
4. Das Plugin sollte jetzt in der Liste installierter Plugins auftauchen. Falls Sie das Plugin nicht in der Liste sehen, starten Sie die IDE neu. Falls Sie es immer noch nicht sehen können, wiederholen Sie Schritte 2 und 3.

Sollte es Probleme geben, wenden Sie sich bitte an einen Tutor oder eine Tutorin.

c) Am rechten Rand Ihrer IDE sollte jetzt ein neues Icon verfügbar sein: Eine kleine Spirale. Klicken Sie auf das Item. Es öffnet sich ein kleines ToolWindow, das rechts angebunden ist. Ziehen Sie das ToolWindow groß genug, so dass Sie den `Export`-Button unten bequem sehen können. **1**
Aktivieren Sie jetzt das Plugin, indem Sie auf `Tools/Werkzeuge > pltLabPlugin > Aktivierung` klicken. Jetzt müssen Sie bitte Ihren Code eingeben, der Ihnen am Anfang ausgeteilt wurde. Sollten Sie **keinen Code** haben, wenden Sie sich bitte an einen Tutor oder eine Tutorin.

d) Erstellen Sie jetzt ein neues IntelliJ-Projekt (mit Buildsystem IntelliJ). Lassen Sie die Default-Einstellungen (außer ggf. Speicherort und Projektname) unverändert. IntelliJ erzeugt für Sie eine Projektstruktur mit einem source-Ordner, in dem eine Main-Datei liegt. In der Main-Datei ist der folgende Code-Schnipsel bereits vorhanden: **1**

```
public class Main {
  public static void main(String[] args) {
    //TIP Press Alt Enter with your caret at the highlighted text
    // to see how IntelliJ IDEA suggests fixing it.
    System.out.printf("Hello and welcome!");

    for (int i = 1; i <= 5; i++) {
      //TIP Press <shortcut actionId="Debug"/>
      //to start debugging your code. We have set one breakpoint
      //for you, but you can always add more by clicking
      //the respective line number.
      System.out.println("i = " + i);
    }
  }
}
```

Sollte bei Ihnen der Codeschnipsel nicht vorhanden sein, legen Sie ihn bitte selbstständig an.

e) Implementieren Sie den vorgeschlagenen Fix und ergänzen Sie eine Zeile Code mit beliebiger Funktionalität. Wenn Ihnen nichts einfällt, können Sie die folgende Zeile ergänzen: **1**

```
int j = 8;
```

Fügen Sie in Ihrer neuen Zeile einen Breakpoint ein. Starten Sie den Debug-Modus (Auf den meisten Systemen per Default mit Umschalt und F9). Was passiert? Benutzen Sie in den folgenden Aufgaben den Debugger bei der Fehlersuche. Weitere Informationen zur Verwendung des Debuggers im Allgemeinen finden Sie hier: `https://www.jetbrains.com/help/idea/debugging-code.html` Nähere Informationen zu Breakpoints in IntelliJ finden Sie hier: `https://www.jetbrains.com/help/idea/using-breakpoints.html`

Stellen Sie die Aufgabe einem Tutor oder einer Tutorin vor bevor Sie weitermachen.

Info zu den Aufgaben in diesem Block:

In diesem Teil des Programmierpraktikums sollen Sie eine Klasse `Shape` schreiben und einen Parser entwickeln, mit dem Objekte dieser Klasse aus einem externen Dateiformat eingelesen werden können. Sie beginnen mit einer schlichten Implementierung und einem selbstgeschriebenen Parser dafür und erweitern beides im Lauf der zwei Tage. Zuletzt lernen Sie eine Bibliothek fürs Einlesen und Erzeugen des Dateiformats kennen und vergleichen Ihre eigene Implementierung mit der Bibliotheksvariante.

Aufgabe 2: Die Klasse Shape — 15 Punkte

a) Schreiben Sie zunächst eine Klasse, die simple Formen und ein paar zugehörige Eigenschaften repräsentieren kann. Die Klasse soll `Shape` heißen. Sie soll die folgenden Eigenschaften haben: **5**
   - Einen Typ `shapeType`, der nur einen von drei Werten annehmen kann: TRIANGLE, CIRCLE oder QUAD.
   - Eine Farbe `color`, die die Linienfarbe textuell (z.B. im Format RGBA) repräsentieren soll. Das Format selbst soll hier frei bleiben und verschiedene Formen von Text zulassen.
   - Eine Position im zweidimensionalen Raum, aufgeteilt in die Eigenschaften `posX` und `posY` (natürliche Zahlen).
   - Eine Liste `tags` von textuellen Tags, die das Objekt beschreiben, zum Beispiel `(Lieblingsobjekt", "komische Farbe", "halbtransparent")`

b) Schreiben Sie Konstruktor, getter und setter. **2**

c) Überschreiben Sie die `toString`-Methode. Es müssen erst mal nur der Typ des Objekts und die Tags ausgegeben werden. **2**

d) Schreiben Sie eine `main`-Methode in einer Klasse `Main`, in der Sie ein Objekt vom Typ `Shape` erzeugen. Lassen Sie sich das Objekt auf der Konsole ausgeben und überprüfen Sie, ob Sie mit Ihrer `toString`-Methode zufrieden sind. Falls nicht, können Sie sie ergänzen. **2**

e) Schreiben Sie in der Main-Klasse oder in einer Util-Klasse eine statische Methode `writeToFile`, die einen String und einen Dateinamen übergeben bekommt. Zusätzlich zur Konsolenausgabe lassen Sie den Text aus der vorigen Teilaufgabe in einer `txt`-Datei festhalten. So steht er auch nach dem Beenden des Programms noch zur Verfügung. **4**

Aufgabe 3: JSON Basics — 25 Punkte

Betrachten Sie die Datei `example1.json` aus dem Ilias-Ordner. Bei `JSON` handelt es sich um ein weit verbreitetes Dateiformat, in dem unter anderem Java-Objekte menschen- und computerlesbar kodiert und weitergegeben werden können. Im Gegensatz zu einem schlichten `txt`-File ist die Kodierung hier nach bestimmten Regeln strukturiert. Machen Sie sich mit dem `JSON`-Format vertraut.

a) Schreiben Sie eine Methode oder teilen Sie Ihren Code sinnvoll auf mehrere Methoden auf, womit aus `JSON`-Dateien mit dem gleichen Format wie `example1.json` Informationen ausgelesen werden können. Sie sollen für diese und die folgenden Aufgaben **keine JSON-Bibliotheken** verwenden solange es nicht explizit in der Aufgabe gefordert ist. **20**
Hinweise: JSON-Dateien können wie besondere Textdateien behandelt werden. Beachten Sie, dass die Reihenfolge der Felder innerhalb des `JSON`-Strings nicht festgelegt ist! Sie können Sich also nicht auf die Reihenfolge verlassen.
Es ist sinnvoll, den Code in mindestens zwei Teile aufzuteilen: einen Teil, der die `JSON`-Datei einliest und ein Zwischenformat ausgibt und einen Teil, der das Objekt vom Typ `Shape` daraus macht

b) Testen Sie Ihre Implementierung in der `main`-Methode, indem Sie aus der Datei ein Objekt vom Typ `Shape` erzeugen. Um den Inhalt lesbar darstellen zu können, sollten Sie Ihre `toString`-Methode so ergänzen, dass alle Felderinhalte ausgegeben werden. Sobald Sie hiermit fertig sind, testen Sie Ihre Implementierung mit den Dateien `example1b.json` und `example1c.json`. **5**

Aufgabe 4: Detailliertere Shapes — 10 Punkte

a) Die `Shape`-Klasse ist bisher relativ schlicht. Ergänzen Sie die Klasse um die folgenden Eigenschaften: **3**
   - Eine Farbe `fillColor`, die sich analog zu `color` verhalten soll und die Füllfarbe des Objekts repräsentiert.
   - Eine natürliche Zahl `lineWidth`.
   - Eine Verzerrung des Objekt entlang der x- und y-Achsen, die mit den Eigenschaften `scaleX` und `scaleY` repräsentiert wird (beides natürliche Zahlen).
   - Eine ganze Zahl `rotation`, die sich zwischen 0 und 365 befinden soll.

b) Ergänzen Sie Ihren Konstruktor entsprechend und erweitern Siedie Klasse um die neuen getter- und setter-Methoden. Erweitern Sie auch die `toString`-Methode um die neuen Felder. **2**

c) Auskommentieren Sie Ihre ursprünglichen `Shape`-Objekte und `JSON`-Importe in der `main`-Methode und schreiben Sie ein neues Testobjekt, das alle jetzt nötigen Felder hat. **1**

d) Damit Sie ganz einfach mehrere Objekte mit gleichen Eigenschaften erzeugen können, schreiben Sie einen sogenannten Copy-Konstruktor. Beachten Sie, dass Sie dafür sorgen müssen, dass von der `tags`-Liste eine Kopie erzeugt wird und **nicht** beide Objekte auf die selbe Liste zeigen. Reicht hierfür eine `shallow copy` oder wird eine `deep copy` benötigt? Erklären Sie beim Vorstellen Ihrer Tutorin oder Ihrem Tutor Ihre Wahl. **3**

e) Erzeugen Sie jetzt mit Hilfe des Copy-Konstruktors noch drei weitere Objekte vom Typ `Shape`. Ändern Sie jeweils mindestens eine Eigenschaft und lassen Sie sich anschließend die neuen Objekte auf der Konsole ausgeben. **1**