Allgemeines

Hinweise

Diskussionsteil: Geben Sie für die Aufgaben 1-6 bitte die Lösungen in Form einer PDF-Datei ab. Es wird nur ein einziges PDF-Dokument für alle Aufgaben 1-6 akzeptiert. Schreiben Sie auf jedes Lösungsblatt Ihren Namen, Ihre E-Mail-Adresse und Ihre Matrikelnummer. Danach können Sie frei formatiert (aber erkennbar, auf welche Aufgabe sich Ihre Antwort bezieht) Ihre Antworten auflisten. Bitte nennen Sie die Datei "DT.pdf".

Programmierteil: Geben Sie für die Aufgaben 7-9 bitte eine einzelne ZIP-Datei ab, in der ausschließlich die in den Aufgaben genannten JAVA-Dateien vorhanden sind. Bitte nennen Sie die Datei "PT.zip".

Laden Sie beide Dateien "DT.pdf" und "PT.zip" in Ilias unter "03_Abgabenbereich" – "Aufgabenblatt 5" einzeln nacheinander hoch.

Diskussionsteil

Aufgabe 1: Spaßbad Hollerbach

Der Eintrittspreis für das Spaßbad Hollerbach ist altersabhängig. Kinder bis einschließlich sechs Jahre sind frei, Schulkinder bis einschließlich zwölf Jahre müssen 2,00 EUR bezahlen, 3,00 EUR bezahlen Jugendliche bis 18 Jahre, Erwachsene bis einschließlich 59 Jahre bezahlen 5,00 EUR und Senioren ab 60 Jahren bezahlen 3,00 EUR.

Sie sollen ein Programm entwickeln, dass nach Eingabe des Alters den Eintrittspreis ausgibt. Falsche Eingaben werden vom Programm nicht berücksichtigt.

1. Bestimmen Sie die Äquivalenzklassen über die Eingabe.
2. Leiten Sie aus den Äquivalenzklassen geeignete Testfälle ab. Vergessen Sie nicht, die Grenzwerte zu testen.

3. Wie ändern sich die Testfälle, wenn statt des Alters das Geburtsdatum eingegeben wird?
4. Wie ändern sich die Äquivalenzklassen, wenn für Schüler, Studenten und Wehr- bzw. Ersatzdienstleistende ein vergünstigter Eintrittspreis festgelegt wird?

Aufgabe 2: Äquivalenzklassen

Implementieren Sie ein Programm, das die quadratische Gleichung $ax^2+bx+c=0 (x \in \mathbb{R})$ korrekt löst.

Schreiben Sie auch Testdaten für das Programm nach folgendem Schema:

Äquivalenzklasse (natürlichsprachliche Beschreibung)	Parameterkombination (a, b, c)	Erwartetes Ergebnis
...		

Aufgabe 3: Molwanische Steuern

Der Finanzminister Molwaniens hat Sie beauftragt, die Implementierung der Berechnung der molwanischen Einkommenssteuer zu testen. Die Routine für die Berechnung der Einkommenssteuer ist folgendermaßen definiert:

```
1 public int getTax(
2       int income, boolean married, int children) {...}
```

Die Berechnungsregeln sind wie folgt:

- Auf alle Einkommen wird ein Freibetrag in Höhe von 4000,00 € angerechnet.
- Für jedes Kind gibt es einen Freibetrag in Höhe von 1700,00 €. Ab fünf Kindern wird das Einkommen nicht versteuert.

- Verheiratete Paare werden gemeinsam veranlagt, für sie gilt ein Splitting-Tarif. D. h. das Gesamteinkommen des Paares wird halbiert, darauf wird der Steuersatz berechnet, das Ergebnis wird zum Schluss verdoppelt.
- Für das zu versteuernde Einkommen gelten die folgenden Steuersätze:
  - 1,00 € bis 42000,00 €: 10 %
  - 42001 € bis 55000,00 €: 25 %
  - Mehr als 55000,00 €: 40 %

Die Berechnung der Steuerlast wird folgendermaßen ausgeführt:

1. Zuerst werden alle Freibeträge abgezogen,
2. dann wird, wenn notwendig, das Einkommen gesplittet
3. und anschließend wird der Steuersatz festgelegt
4. und die zu zahlende Einkommenssteuer berechnet.
5. Wurde der Splitting-Tarif angewendet, wird die zu zahlende Einkommenssteuer verdoppelt.

Der Einfachheit halber werden alle Beträge auf ganze Zahlen abgerundet, negative Einkommen werden nicht besteuert.

1. **Implementieren Sie die Methode getTax.**

Aufgabe 4: JUnit-Testfälle

Legen Sie die Äquivalenzklassen für die Methode getTax aus Aufgabe 3 fest.

Entwickeln Sie aus den Äquivalenzklassen geeignete Testfälle und implementieren Sie diese als JUnit-Testfälle.

Fügen Sie die JUnit-Testfälle in das PDF-Dokument ein.

Aufgabe 5: Geometrie

Nachfolgend ist eine Funktion abgedruckt, die prüft, ob ein Dreieck rechtwinklig ist.

```
public boolean isRightangled(Triangle triangle) {
        final double RIGHT_ANGLE = 90.0;
        if (triangle.getAlpha() == RIGHT_ANGLE) {
```

```
            return true;
        } else if (triangle.getBeta() == RIGHT_ANGLE) {
            return true;
        } else if (triangle.getGamma() == RIGHT_ANGLE) {
            return true;
        }
        return false;
}
```

1. Erstellen Sie Testfälle, mit denen Sie 100 % Anweisungsüberdeckung erreichen.
2. Die Ersetzung der if-Kaskade in der Routine durch einen logischen Ausdruck würde die Implementierung stark verkürzen.
    1. Wie sieht diese Implementierung aus?
    2. Wie ändern sich die Testfälle für die Anweisungsüberdeckung von gerade eben?
    3. Wie bewerten Sie die Aussagekraft der Anweisungsüberdeckung für die verkürzte Variante?

Aufgabe 6: Typische Fehler und ihre Korrektur

Für ein Bonusprogramm werten Sie die Einkäufe Ihrer Kunden aus. Wenn die Einkäufe eines Kunden einen bestimmten Mindestumsatz überschreiten, dann erhält der Kunde einen Kugelschreiber mit Ihrem Firmenlogo – Motto: Kleine Geschenke erhalten die Freundschaft.

Der Programmcode für die Berechnung des Umsatzes eines Kunden ist nachfolgend dargestellt. Im Programmcode sind zwei Fehler.

1. Welche Fehler sind das?
2. Wie können die Fehler korrigiert werden?
3. Welche weiteren Unzulänglichkeiten (mindestens drei) hat der Code?

```
1 public int getSales(int customerId, int receiptId) {
2   int customerSales = getPriorSales(customerId);
3   int [] receipt = getReceipt(receiptId);
```

```
4    int receiptLength = getReceiptLength(receiptId);
5    int itemCount = 0;
6
7    while (itemCount <= receiptLength) {
8       customerSales += getPrice(receipt[itemCnt]);
9       itemCnt = itemCnt + 1;
10   }
11
12   return customerSales;
13 }
```

Programmierteil

Der Programmierteil umfasst drei Aufgaben, bei denen Sie Änderungen an der Software „PackageCalculator“ durchführen. Änderungen am Quellcode, die Sie in den vorhergehenden Aufgabenblättern durchgeführt haben, dürfen Sie im Quellcode belassen. Ansonsten dürfen Sie nur Änderungen an Dateien durchführen, die in den einzelnen Aufgaben genannt sind. Packen Sie alle drei Dateien aus den Aufgaben 7, 8, und 9 in eine ZIP-Datei und geben Sie diese als "PT.zip" ab.

Aufgabe 7: Radiobuttons

Abgabe (zu ZIP-Datei hinzufügen): CalculatorArea.java

Lesen Sie zunächst die JavaFX-Dokumentation zur Klasse `RadioButton`:

http://docs.oracle.com/javafx/2/ui_controls/radio-button.htm

Fügen Sie in der Klasse CalculatorArea zwei Radiobuttons ein. Der eine Radiobutton soll „DHL“ benannt sein und der andere „Hermes“. Positionieren Sie die Radiobuttons im Fenster oberhalb der Größenangabe des Pakets (d. h. ganz oben in der GridPane CalculatorArea). Konfigurieren (d. h. gruppieren) Sie die beiden Radiobuttons so, dass jeweils immer nur einer der Buttons ausgewählt ist.

Fügen Sie eine öffentliche (public) Methode mit dem Rückgabetyp String hinzu, mit der man abfragen kann, welcher Paketdienst ausgewählt ist.

Aufgabe 8: Anpassung der Portokosten

Abgabe (zu ZIP-Datei hinzufügen): Calculator.java

In Aufgabe 7 des vorigen Übungsblattes haben Sie eine Datei „shippingCosts.csv“ erstellt. Ersetzen Sie den Inhalt dieser Datei durch folgende Zeile (bitte geben Sie diese Datei nicht ab):

```
DHL;3.89;4.39;5.99;7.99;14.99;Hermes;3.69;4.19;5.79;6.99;10.99;
```

Die Zahlen nach dem Text „DHL“ stehen für die Versandkosten mit DHL; die Zahlen nach „Hermes“ seinen hier die Versandkosten mit dem Paketdienst Hermes. Die Kategorien der Pakete (Größe/Gewicht/Gurtmaß) seinen hier für alle Paketdienste gleich d. h. z. B. Pakete, die bei DHL 4,39 Euro kosten, kosten bei Hermes hier 4,19 Euro (Hinweis: In Wirklichkeit ist das nicht der Fall).

Schreiben Sie die Klasse Calculator.java so um, dass die Preise desjenigen Paketdienstes verwendet werden, die der Benutzer der Anwendung mit den in Aufgabe 7 dieses Übungsblattes hinzugekommenen Radiobuttons ausgewählt hat.

Aufgabe 9: Spielwiese

Abgabe (zu ZIP-Datei hinzufügen): InspectorArea.java

Suchen Sie sich aus der Dokumentation von JavaFX, Abschnitt UI Controls (http://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm) **drei** verschiedene UI Controls aus, mit denen der Benutzer Eingaben tätigen kann (z. B. ein Button) **und drei** verschiedene Controls zur Ausgabe (z. B. ProgressBar). Fügen Sie diese sechs Elemente zur Klasse InspectorArea hinzu, so dass diese in der Anwendung angezeigt werden. Verknüpfen Sie die Eingaben mit den Ausgaben auf beliebige Weise, d. h. z. B. wenn man auf den Button klickt, soll die Progressbar um 10% erhöht werden. Jede Änderung einer Eingabe Control UI muss eine sichtbare Änderung mindestens einer Ausgabe Control UI zur Folge haben.

Beachten Sie bitte, dass diese Aufgabe nichts mit der Funktionalität der Anwendung (Berechnung des Paketportos) zu tun hat, sondern nur dem Kennenlernen weiterer UI Elemente und ihrer Verknüpfungen dient.