Programmierung II

Assignments

Struktur

- Jeder fertigt 3 Individualleistungen an
- Eine schriftliche Ausarbeitung, 2 programmierte Ausarbeitungen
- Maximal sind 120 Punkte erreichbar

- Entschuldigte, verpasste Abgaben müssen ersetzt werden
- Unentschuldigte, verpasste Abgaben werden mit 0 Punkten bewertet
- Täuschungsversuche führen zu sofortigem Nichtbestehen

Disclaimer: Ich entschuldige euch nicht. Das müsst ihr mit dem Sekretariat und der Studiengangsleitung abklären.

Prüfungsthemen

1. Polymorphismus (40 Punkte, *prog.*)
2. Multithreading (40 Punkte, *schriftl.*)
3. Algorithmen (40 Punkte, *prog.*)

Terminplanung

	20.03	Strings & Code Quality
	27.03	Polymorphismus & Maven
	03.04	Exceptions
2 Wochen	10.04	---
	17.04	Multithreading
	24.04	Grundlagen Algorithmen & Datenstrukturen I
3 Wochen	01.05	---
	08.05	Sortieralgorithmen
	15.05	Graphenalgorithmen
4 Wochen	22.05	User Interfaces

Abgaberegeln

- Vor Deadline via Moodle

- Schriftl. Abgaben als „*Matrikelnr_Assignmentnr*.pdf“
- Prog. Abgaben als „*Matrikelnr_Assignmentnr*.zip“

- Bspw. „12345678_1.zip“ oder „12345678_2.pdf“

Bewertung des schriftl. Assignments (40P)

Die schriftliche Ausarbeitung wird nach folgendem Schema bewertet:

- Inhalt (20 Punkte)
  - Komplexität, Ergebnis, Bezug auf Fragestellung, …
- Fachlichkeit (10 Punkte)
  - Nutzung der korrekten Terminologie, Korrektheit der Aussagen, …
- Form (10 Punkte)
  - Quellenarbeit falls vorhanden, Umfang, Rechtschreibung & Grammatik (!) …

Bewertung der prog. Assignments (40P)

Programmierte Ausarbeitungen werden nach folgendem Schema bewertet:

- Programm (40 Punkte)
  - Je nach Menge der bestandenen Testfälle

Bewertung der prog. Assignments (40P)

- Code Abgaben werden mit Unit Tests geprüft
- Die von mir definierte Testfälle werde ausgeführt und ich notiere, welche davon erfolgreich sind und welche nicht – dieser Report dient der Bewertung (40P)

- Ihr erhaltet Interfaces mit vordefinierten Methoden und Beschreibungen von mir, die ihr dann „nur“ noch umsetzen müsst
- Die Tests prüfen eure Implementierungen der Interfaces

- Die Tests werden vorab nicht bekannt gegeben
- Ihr erhaltet immer ein Projekt mit allen Interfaces, die zu implementieren sind

Bewertung der prog. Assignments (40P)

- Die vordefinierten Interfaces dürfen nicht bearbeitet werden
- Eure Implementationen landen alle im Package **`impl`**
- Evtl. notwendige Hilfsklassen landen auch im Package **`impl`**

Vorgehen:

1. Öffnet das Maven-Projekt in eurer IDE
2. Macht euch mit der Struktur vertraut – ihr werdet Compilerfehler finden, die nach sauberer Implementierung der Interfaces verschwinden
3. Implementiert notwendige Klassen
4. Testet eure Implementierungen
5. Zippt euren **`impl`** Ordner und gebt die Datei online ab

Bewertung der prog. Assignments (40P)

In die .zip Datei packt ihr Folgendes:

- Den Inhalt eures **`impl`** Ordners
- Sonst nichts.

Ihr benennt eure .zip Datei:

- *Matrikelnr_Assignmentnr*.zip
- Nicht anders.

! Bei extrem schlechter Code Qualität/ Form halte ich es mir offen, euch schlechter zu bewerten. !

Aufgabe 1: Polymorphismus *(prog.)*

Folgendes Programm soll erstellt werden:

- Ein Programm, das die monatlichen Löhne und Boni verschiedener Mitarbeiter einer Firma berechnet
- Mitarbeiter sind klassifiziert als: Worker, Employee oder Manager
- Worker erhalten einen Stundenlohn und Überstundenbonus
- Employee erhalten ein Pauschalgehalt und Pauschalbonus
- Manager erhalten ein Pauschalgehalt und Kommission abhängig der Sales
- Alle Löhne und Boni werden in der MonthlyPayroll verwaltet und summiert
- Beim Anlegen der Payslips ist zusätzlich auf bestimmte Sonderfälle zu achten
- PayslipHelper steht zur Verfügung und hilft dabei, alle Klassen zu initialisieren.

Abgabe bis: 16.04.2023 23:59 via Moodle

Aufgabe 1: Polymorphismus *(prog.)*

Sonderfälle

`UNEXPECTED_NEGATIVE_VALUE:`
Alle Werte, außer der Bonus eines Employees, müssen größer oder gleich 0 sein

`COMMISSION_RATE_TOO_HIGH:`
Die prozentuale Kommission eines Managers darf 25 % der Sales nicht übersteigen

`NOT_ENOUGH_HOURS:`
Worker dürfen nicht weniger als 40 Stunden gearbeitet haben

`BONUS_TOO_HIGH:`
Worker & Employee Boni dürfen 50 % ihres regulären Lohns nicht übersteigen

`INDEX_TOO_HIGH:`
Falls getPayslip mit einem Index aufgerufen wird, der größer als maxPayslips ist

`PAYSLIP_OVERFLOW:`
addPayslip darf keine weiteren Payslips hinzufügen, nachdem maxPayslips erreicht wurde

Thank you.

lukas@fortual.com