1 Übungsbetrieb

1.1 Rahmenbedingungen

Sie erhalten insgesamt 4 Aufgabenblätter zur Begleitung der Informatik 3 Vorlesung. Die Übungsaufgaben sollen Sie nutzen, um die in den theoretischen Inhalten der Vorlesung untergebrachten Konzepte der Softwareentwicklung zu vertiefen. Diese Konzepte sind insbesondere Compiler, Von-Neumann-Architektur, Spezifikation durch Vertrag, Objektorientierung, Parallelisierung, funktionale Programmierung, Rekursion. Sie sind eingeladen, diese Programmieraufgaben in Gruppen zu bearbeiten. Zur Gruppenbildung und Begleitung sind die Praktikumseinführungstermine zu berücksichtigen.
Sie sollen in folgenden Programmiersprachen Erfahrungen sammeln:

- C++
- C#
- Java
- Python

Dokumentation Es ist immer eine Dokumentation als PDF für jede Aufgabe abzugeben. Dieses Dokument enthält eine kurze Beschreibung Ihrer Lösung mit einem UML Diagramm und ggf. Code Metriken. Das Dokument muss auch die URL ihres Repositories als HTTP-Link enthalten. Jedes abgegebene Dokument muss weiterhin alle notwendigen Metadaten enthalten. Diese sind:

1. Vollständiger Name und Matrikelnummern aller Teammitglieder
2. Modulname (also Info3)
3. Semester (z.B. SoSe 2023) und
4. aktuelles Datum

1.2 Entwicklungsumgebung

Für die Info3 Projekte steht eine integrierte Entwicklungsumgebung auf Basis von Visual Studio bereit, die offline (lokal) ausgeführt werden kann, oder online durch ein Server ergänzt wird. Diese Entwicklungsumgebung ist zu nutzen. Melden Sie sich umgehend in RELAX, wenn die Entwicklungsumgebung nicht wie erwartet funktioniert (und nicht erst kurz vor Abgabe oder nach mehreren Wochen). Der abgegebene Code muss in dieser Entwicklungsumgebung lauffähig sein.

1.3 Punktevergabe und -abzug

Für die Entwicklung der Software nutzen Sie *Git* zur Versionsverwaltung. Legen Sie sich dafür als Gruppe ein Repository an und erstellen für jedes Aufgabenblatt **einen Projektordner** in diesem Repository. Erstellen Sie nicht für jedes Aufgabenblatt ein neues Repository.

Für die Bewertung der Aufgabenblätter werden zwei Aspekte betrachtet: (1) die eigentliche Lösung der Aufgabe anhand der im Arbeitsblatt gegebenen Punkte, und (2) die Einhaltung der Coding Guidelines, für die es die unten genannten Abzüge geben kann.

2 Coding-Guidelines

Pro Verstoß wird in der Regel 1 Punkt abgezogen. Mehrfachverstöße sind möglich. Bei einigen Kriterien können mehrere Punkte abgezogen werden. Dies ist nachstehende dokumentiert.

1. Gecodet, dokumentiert und kommentiert wird auf Englisch. Die abzugebende Dokumentation in RELAX soll auf deutsch geschrieben sein.

2. Code ist weitestgehend selbstdokumentierend und bei Bedarf mit zusätzlichen Kommentaren versehen. Das bedeutet auch, dass Variablen, Methoden und Klassen sinnvoll benannt sind. Gut lesbarer Code ist dabei wichtiger als mit Kommentaren gespickter Code. Im Zweifel lieber den Code umstrukturieren oder Variablen und Methoden umbenennen, um den Code besser lesbar zu machen. Davon unberührt bleibt Guideline Nr. 3.

3. Die Dokumentation aller Methoden ist auszufüllen (kurze Beschreibung der Funktion, eventuelle Parameter und Rückgabewerte). Die zu dokumentierenden Elemente sind: Methoden jeglicher Art, Konstruktoren sowie Getter und Setter.

4. Code ist immer strukturiert. Das heißt: Blöcke haben genau einen Ein- und einen Ausgang. Mehrfache return-Anweisungen sind zu vermeiden. In Schleifenkörpern gibt es generell weder return- noch break-Anweisungen.

5. Code ist effizient zu schreiben. So sind etwa foreach-Schleifen nur genau dann zu benutzen, wenn alle Elemente eines Feldes betrachtet werden.

6. Boolsche Variablen werden nicht explizit mit true oder false verglichen.

7. Funktionalität ist zu kapseln, wann immer möglich (Don't Repeat Yourself).

8. Bezeichnungen sind einheitlich zu halten: entweder camel-case, oder underscore innerhalb eines Projekts.

9. Auch Blöcke, die nur eine Anweisung beinhalten, sind zu klammern.

10. Zählvariablen von Schleifen sind nicht zu missbrauchen (= nicht im Schleifenkörper zu verändern). Im Zweifelsfall lieber einen Iterator verwenden.

11. Literale Konstanten im Code sind zu vermeiden. Stattdessen sind an sinnvollen Stellen entsprechende Konstanten anzulegen.

12. Verweise auf referenzierte Dateien (Bibliotheken, Dummy-Daten, Konfigurationsdateien, etc.) werden relativ zum Projekt referenziert, niemals mit absoluten Pfadangaben.

13. Git-Commits haben einen sinnvollen Kommentar zu erhalten, der zumindest kurz beschreibt, was durch den Commit verändert wurde.

14. Die zusätzliche Dokumentation in RELAX gibt eine Übersicht über den Code durch Verwendung von Metriken, UML Diagrammen und einer kurzen Beschreibung der Klassen oder Module. (max. -3 Punkte)

15. Der abgegebene Code ist für die angekündigten Entwicklungsumgebungen lauffähig, besitzt eine Main-Methode oder einen Testtreiber (max. - 2 Punkte)

16. Wann immer möglich, verwenden Sie *Spezifikation durch Vertrag* (erst nach Einführung des Konzepts in der Veranstaltung) (max. -2 Punkte)

17. Sorgen Sie für ein tracing des Programmablaufs. Dazu können Sie geeignete `print`-Anweisungen verwenden, oder noch besser ein Logging-Framework wie `Log4J` oder das `logging`-Package.