Regeln zu Bonusaufgaben

Für die Bonusaufgaben gelten folgende Regeln, ausser die Aufgabenbeschreibung verlangt explizit etwas anderes. Einige der Regeln betreffen Konzepte, deren Java-Konstrukte erst später im Semester behandelt werden.

- Die Abgabe wird mit Java 17 getestet. Wenn Ihre Lösung nur mit einer anderen Java-Version funktioniert (also nicht mit Java 17), dann führt dies zu 0 Punkten.
- Bonusaufgaben werden automatisiert getestet und darauf basierend bewertet. Kommentare im Programm werden deshalb grundsätzlich nicht beachtet und vorgegebene Ausgaben des Programms müssen exakt wiedergegeben werden. Insbesondere können Programme, die nicht kompilieren, nicht bewertet werden und erhalten keine Punkte. Das gilt auch, wenn sich der Kompilierfehler in einer von der Lösung unabhängigen Methode in derselben Datei befindet!
- Java ist Case Sensitive, weshalb die vorgegebene Gross- und Kleinschreibung von der Aufgabenstellung exakt übernommen werden muss. Wenn beispielsweise Methodennamen nicht exakt den Vorgaben entsprechen, kompiliert das Programm möglicherweise nicht zusammen mit unseren Tests (was zu 0 Punkten führen kann).
- Was zum Abgabetermin auf dem Server der ETH ist, d.h. erfolgreich **vor** dem Abgabetermin **gepusht** (nicht nur committet) wurde, bestimmt die Abgabe. Es ist Teil der Aufgabe zu wissen, wie man etwas committet und pusht. Auf der GitLab-Seite Ihres Repositorys (https://gitlab.inf.ethz.ch/course-eprog2023/students/nethz-id/) kann nachgeschaut werden, was auf dem Server ist. Die Lösung muss sich im ‘main’-Branch von Git befinden. Am Ende dieses Dokumentes ist eine Anleitung, wie man nachschauen kann, wann genau etwas gepusht wurde.
- Die gesamte Lösung muss sich im ‘src’-Ordner des entsprechenden Bonusprojekts befinden. Ausser den Dateien der Lösung dürfen sich im Ordner keine weiteren Dateien befinden.
- Gegebene Methoden- und Konstruktoren-Deklarationen dürfen nicht geändert werden. Zu einer Deklaration gehören: Name, Rückgabe- und Parametertypen, Modifier (“public / static / protected / private”), deklarierte Exceptions (“throws”). Auch Namen und Modifier von gegebenen Klassen und Interfaces dürfen nicht geändert werden.
- Es ist grundsätzlich erlaubt, weitere Methoden, Konstruktoren, Attribute, Klassen und Interfaces hinzuzufügen.
- Das abgegebene Programm darf nicht mehr machen als von der Aufgabe verlangt. Beispielsweise gelten zusätzliche “System.out.print()”-Anweisungen als falsch, wenn die Aufgabe diese nicht vorschreibt. Auch das Modifizieren von übergebenen Argument-Objekten ist falsch, wenn dies nicht von der Aufgabe verlangt wird.
- Bei Unklarheiten fragen Sie bei Ihrem/Ihrer Assistent/in nach.
- Das Verwenden von ‘static’-Attributen ist grundsätzlich falsch. Rechnen Sie damit, dass wir das abgegebene Programm mehrfach ausführen, ohne dass ‘static’-Attribute neu initialisiert werden.
- Es dürfen keine unnötigen Import-Anweisungen im abgegebenen Programm sein. Das bedeutet, dass nur Klassen importiert werden sollen, die auch verwendet

werden. Eclipse kann unnötige Imports entfernen (Menüeintrag “Source → Organize Imports”).

- Die Aufgabe muss eigenhändig und alleine gelöst werden. Das von Ihnen eingereichte Programm muss von Ihnen selbst geschrieben worden sein. Das Kopieren von Programm(teil)en ist nicht erlaubt (unabhängig von wo kopiert wurde). Genauso ist es nicht erlaubt, das eigene Programm anderen zur Verfügung zu stellen (siehe die Regeln im Vorlesungsverzeichnis).
- Sie können mit anderen Studierenden über die Aufgaben **reden**. Wenn Sie aber gemeinsame Notizen machen (z.B. zeichnen Sie eine Skizze auf eine Tafel, ein Blatt, ein Tablet, oder Sie entwickeln ein Programm(segment)), dann dürfen Sie keine Aufzeichnungen aus dem Treffen mitnehmen.
  Werfen Sie alle Aufzeichnungen weg und warten Sie (mindestens) 1 Stunde nach dem Treffen/der Besprechung, bevor Sie etwas aufschreiben oder Ihr Programm entwickeln.
- Wir schlagen vor, dass Sie zuerst versuchen, die Bonusaufgabe selbständig zu lösen, bevor Sie auf Literatur/Webseiten, Diskussionen mit Studierenden, oder andere Hilfsmittel zurückgreifen (in der Prüfung müssen Sie auch selbständig arbeiten). Wenn Sie aber eine Idee in der Literatur finden, oder Hilfe durch einen Programmer Assistant (z. B. Copilot o. ä.) in Anspruch nehmen, dann müssen Sie 1 Stunde warten, bevor Sie *selbständig* an Ihrem Programm weiterarbeiten. Sie dürfen auf keinen Fall von irgendwo Code kopieren, oder von Copilot generierten Code in Ihr Programm einfügen.
- Wenn Sie einen Programmer Assistant verwenden, dann müssen Sie alle Eingaben selbst formuliert haben. Es ist nicht erlaubt, den Text (oder Teile des Textes) der Aufgabenstellung einem Programmer Assistant als Eingabe zu geben. Es gibt zwei Gründe für diese Regel: (1) Dadurch, dass Sie die Eingabe selbst formulieren, setzen Sie sich mit der Aufgabe auseinander. (2) Alle Eingaben werden (potentiell) vom Anbieter des Programmer Assistants benutzt, ohne die Rechte der ETH in Betracht zu ziehen, und wir wollen das nicht (unkontrolliert) zulassen.
- Es gilt die “Disziplinarordnung der Eidgenössischen Technischen Hochschule Zürich”.

Pushzeitpunkt ist für die Bewertung relevant

Bei den Bonusaufgaben korrigieren wir die letzte Lösung in Ihrem Git-Repository, welche **vor** dem Abgabetermin auf den GitLab-Server **gepusht** wurde (d.h. durch “push” in Git). Wenn Sie vor dem Abgabetermin Ihre Lösung **committet** haben (d.h. durch “commit” in Git), aber den gleichen Commit erst **nach** dem Abgabetermin **gepusht** haben, dann wird diese Lösung **nicht** in Betracht gezogen (denn wir können in so einem Fall gar nicht auf Ihre Lösung zugreifen, da Ihre Lösung vor dem Abgabetermin nur auf Ihrem lokalen Rechner existiert).

Beachten Sie: Wenn Sie um 11:00 lokal committen und dann um 12:30 den Commit pushen, dann wird auf Ihrer Git-History auf GitLab [`https://gitlab.inf.ethz.ch/course-eprog2023/students/nethz-id/-/commits/main`] der Commitzeitpunkt (also 11:00) und nicht der Pushzeitpunkt angezeigt. Es ist möglich,

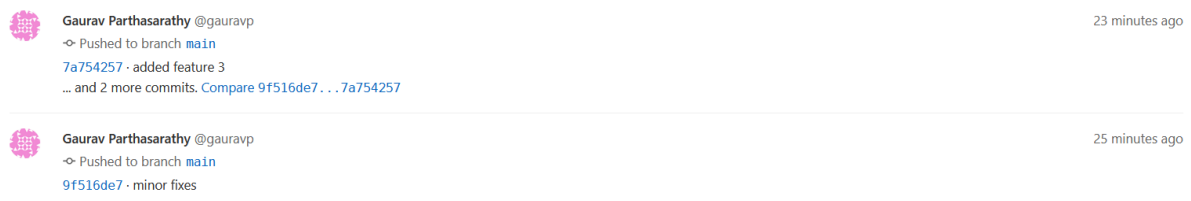
Commitzeitpunkte vor einem Push zu manipulieren. Deswegen werden Commitzeitpunkte für die Benotung der Bonusaufgaben nicht in Betracht gezogen.

Es gibt mindestens zwei Optionen, um herauszufinden, ob Ihre Lösung vor dem Abgabetermin gepusht wurde:

1. Sie gehen vor dem Abgabetermin auf Ihr Repository (`https://gitlab.inf.ethz.ch/course-eprog2023/students/nethz-id/`)und schauen, ob Ihre gewünschten Änderungen ersichtlich sind.
2. Wenn Sie vor (oder nach) dem Abgabetermin schauen wollen, wann Ihre Lösung genau **gepusht** wurde, dann können Sie dies über die Activity-Ansicht auf gitlab.inf.ethz.ch anschauen (unten folgt eine Beschreibung).

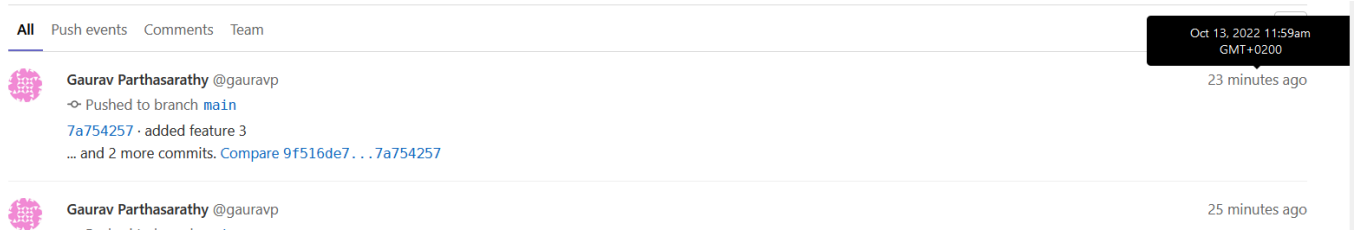
Pushzeitpunkt via Activity-Ansicht herausfinden
Gehen Sie in Ihrem Repository auf die Activity-Ansicht:
`https://gitlab.inf.ethz.ch/course-eprog2023/students/nethz-id/activity`

In der Activity-Ansicht sieht man alle Pushereignisse (und auch andere Ereignisse). Wenn Sie vor dem nächsten Push zuerst dreimal committen und erst dann pushen, dann gilt dies als einzelnes Pushereignis. Hier ein Beispiel:



In dieser Abbildung sind zwei Pushereignisse aufgeführt. Zuerst wurde nur ein Commit (mit Commitnachricht “minor fixes”) gepusht und dann wurden drei Commits zusammen gepusht (wobei der neueste Commit die Nachricht “added feature 3” hat).

Den Pushzeitpunkt kann man bekommen, indem man mit der Maus über die Zeitbeschreibung geht:



Hier wurden die drei Commits zusammen am 13. Oktober um 11:59 gepusht.
(Erstellt am 17. 10. 23)