Git bzw. Github

1. Unterschied zwischen Git und Github?

Git ist ein System zur Versionsverwaltung in Entwicklungsprojekten. Git kann auf eigenen Server installiert werden oder alternativ bei einem Hoster anmieten. Github ist ein solcher Webbassierter Hostingdientst zum Hosting von Git-Repositorys. Github wird oft von Leuten verwendet um Open Source Projekte öffentlicht verwalten zu können. Git ist also das Tool und Github der Service für Projekte, die Git einsetzen wollen.

1. Verwendungszweck von Git bzw. Github?  
   Git bzw. Github werden verwendet um parallel am selben Projekt zu arbeiten. Natürlich kann Github auch dafür verwendet werden, um den geschriebenen Code zu sichern. Github kann auch verwendet werden, um jederzeit an seinem Projekt weiter zu arbeiten(entsprechende Netzwerkverbindung und Programmiersoftware vorausgesetzt), ohne, dass man sich auf dem eigentlichen Arbeitscomputer befindet.
2. Grundlegende Begriffe beim Arbeiten mit Git bzw. Github!  
   Repository: Ein Git Repository kann als „Projekt“ verstanden werden, in dem die Dateien für ein Softwareprojekt abgelegt werden.

Master Branch: Das Verzeichnis aus dem die Änderungen an den Dateien von

Softwareprojekten hochgeladen werden. Ein Masterbranch wird erstellt  
 indem man mit cd (change directory) in das gewünschte Verzeichnis geht   
 und dort dann den Befehl git init eingibt. Das Master Branch ist ein lokales   
 Repository.

Gitignore: Sie können Dateien in ihrem master branch zur gitignore Datei hinzufügen,   
 damit diese beim comitten nicht berücksichtig werden. Dafür erstellen wir   
 eine normale Textdatei(.gitignore.txt) und schreiben in diese die Dateien   
 hinein, welche nicht berücksichtigt werden sollen (Bei uns bin und \*.class).   
 Mit dem Befehl mv .gitignore.txt .gitignore.gitignore verwandelt wir die   
 Textdatei in eine gitignore Datei. Wenn wir die gitignore Datei mit dem   
 Befehl git add .gitignore.gitignore hinzufügen, werden die von uns   
 angegebenen Dateien beim Commit nicht berücksichtigt.

Add: Veränderte Dateien werden zum Hochladen hinzugefügt. Vor dem   
 Hinzufügen kann mit git status überprüft werden, welche Dateien verändert  
 wurden. (Diese sind rot gekennzeichnet). Nach dem Hinzufügen mit dem  
 Befehl git add <dateiname> werden diese bei git status grün gekennzeichnet,   
 was bedeutet die Dateien können „committed“ werden.

Commit: Ist der Vorgang, um die neueste Version eines Branches bei der   
 Versionsverwaltung einzureichen. Das heißt nachdem die Änderungen an   
 einem Projekt vorgenommen wurden, werden diese „committed“.   
 Üblicherweise wird auch noch eine Nachricht, welche den Commit beschreibt  
 erstellt. Befehl: git commit –m „zu verfassende Nachricht“.

Push: Mit dem Befehl git push origin master können Änderungen an ein entferntes  
 Repository gesendet werden (vom lokalen Repository = Master Branch zu   
 beispielweise einem Repository auf Github).

Remote Add: Um dein lokales Repository(Master Branch) mit einem entferntem Repository  
 beispielweise auf Github verbinden zu wollen verwendet man den Befehl   
 git remote add origin <URL>.

1. Welche Hilfsmittel können verwendet werden, um Dateien hochzuladen?  
   Mittels der Gitbash (siehe oben!) und mittels Plugins in Software Programmen wie Eclipse.

JUNIT Testing

1. Was ist Testen?  
   Testen ist der Prozess der Überprüfung der Funktionalität der Anwendung, ob es nach Anforderungen funktioniert.
2. Was ist Unit Testing?

Unit Testing ist das Testen einer einzelnen Entität (Klasse oder Methode).

1. Was ist manuelles Testen?  
   Das manuelle Ausführen der Testfälle ohne Werkzeugunterstützung wird als manuelles Testen bezeichnet.
2. Was ist automatisiertes Testen?  
   Die Werkzeugunterstützung und die Ausführung der Testfälle mithilfe eines Automatisierungswerkzeugs werden als Automatisierungstests bezeichnet.
3. Nachteile vom Manuellen Testen?  
   Zeitrauben und Mühsam  
   Enorme Investitionen in Personalwesen  
   Weniger Zuverlässig  
   Nicht programmierbar
4. Vorteile vom Automatisierten Testen?  
   Schnell  
   Geringe Investitionen  
   Zuverlässiger  
   Programmierbar
5. Was ist JUnit?  
   JUnit ist ein Regressionstest-Framework, das von Entwicklern verwendet wird, um Unit-Tests in Java zu implementieren und die Programmiergeschwindigkeit zu beschleunigen und die Qualität des Codes zu erhöhen.
6. Wichtige Eigenschaften von JUnit!  
   Es ist ein Open-Source-Framework.  
   Stellt eine Anmerkung bereit, um die Testmethoden zu identifizieren.  
   Bietet Assertions zum Testen erwarteter Ergebnisse.  
   Stellt Test runner für die Ausführung von Tests bereit.  
   JUnit-Tests können automatisch ausgeführt werden und sie überprüfen ihre eigenen Ergebnisse und liefern sofortige Rückmeldungen.  
   JUnit zeigt den Testfortschritt in einem Balken an, der grün ist, wenn der Test in Ordnung ist, und er wird rot, wenn ein Test fehlschlägt.
7. Was ist ein Unit Test Case?  
   Ein Unit Test Case ist ein Teil des Codes, der sicherstellt, dass der andere Teil des Codes (Methode) wie erwartet funktioniert. Um diese gewünschten Ergebnisse schnell zu erreichen, ist ein Test-Framework erforderlich. JUnit ist ein perfektes Unit-Test-Framework für Java-Programmiersprachen.
8. Was sind die besten Anwendungen für einen JUnit Test?  
   Ein formaler geschriebener Einheitstestfall ist durch eine bekannte Eingabe und durch eine erwartete Ausgabe gekennzeichnet, die vor der Ausführung des Tests ausgearbeitet wird. Der bekannte Eingang sollte eine Vorbedingung testen und die erwartete Ausgabe sollte eine Nachbedingung testen.  
   Für jede Anforderung müssen mindestens zwei Einheitstestfälle vorhanden sein: ein positiver Test und ein negativer Test. Wenn eine Anforderung Unteranforderungen hat, muss jede Unteranforderung mindestens zwei Testfälle als positiv und negativ haben.
9. Wenn im Entwicklungszyklus werden die Unit Tests geschrieben?  
   Tests werden vor dem Code während der Entwicklung geschrieben, um Kodierern zu helfen, den besten Code zu schreiben.
10. Warum meldet JUnit nur den ersten Fehler in einem einzigen Test?  
    Die Meldung mehrerer Fehler in einem einzelnen Test ist im Allgemeinen ein Zeichen dafür, dass der Test zu viel macht und der Test zu groß ist. JUnit wurde entwickelt, um mit einer Reihe von kleinen Tests zu arbeiten. Es führt jeden Test innerhalb einer separaten Instanz der Testklasse aus. Es meldet einen Fehler bei jedem Test.
11. Was sind die Kernfeatures von JUnit?  
    Das JUnit-Test-Framework bietet folgende wichtige Funktionen:

fixture

Test suites

Test runners

JUnit-Klassen

1. Was ist eine fixture?  
   Fixture ist ein fester Zustand einer Gruppe von Objekten, die als Basis für die Ausführung von Tests verwendet werden. Der Zweck einer Testvorrichtung ist es sicherzustellen, dass es eine bekannte und feste Umgebung gibt, in der Tests durchgeführt werden, so dass die Ergebnisse wiederholbar sind. Es beinhaltet folgende Methoden -

Die Methode setUp (), die vor jedem Testaufruf ausgeführt wird.

Die Methode tearDown (), die nach jeder Testmethode ausgeführt wird.

1. Was ist ein Test suite?  
   Testsuite bedeutet, einige Unit-Testfälle zu bündeln und zusammen auszuführen. In JUnit werden sowohl @RunWith als auch @Suite Annotation verwendet, um den Suite-Test auszuführen.
2. Was ist ein Test runner?  
   Test runner wird zum Ausführen der Testfälle verwendet.
3. Was sind JUnit Klassen – nenne ein paar Beispiele!  
   JUnit-Klassen sind wichtige Klassen, die beim Schreiben und Testen von JUnits verwendet werden. Einige der wichtigen Klassen sind -

Assert - Es enthält eine Reihe von Assert-Methoden.

TestCase - Es enthält einen Testfall definiert mit der Ausstattung mehrere Tests ausführen.

TestResult - Es enthält Methoden zum Sammeln der Ergebnisse der Ausführung eines Testfalls.

TestSuite - Es ist ein Testverbund.

1. Was sind Annotations und für was sind sie nützlich in JUnit?

Annotationen sind wie Metatags, die Sie Ihrem Code hinzufügen und auf Methoden oder in der Klasse anwenden können. Die Annotation in JUnit gibt uns Informationen über Testmethoden, welche Methoden vor und nach Testmethoden ausgeführt werden, welche Methoden vor und nach allen Methoden ausgeführt werden, welche Methoden oder Klassen während der Ausführung ignoriert werden.

1. Was ist eine @Ingnore annotation und für was ist sie nützlich?

Im Folgenden finden Sie einige nützliche Informationen zur Annotation von @Ignore -

Sie können alle @ Ignore-Annotationen im Quellcode leicht identifizieren, während unannotierte oder auskommentierte Tests nicht so einfach zu finden sind.

Es gibt Fälle, in denen Sie einen Code, der fehlschlägt, nicht beheben können, aber Sie möchten immer noch umhergehen, damit er nicht vergessen wird. In solchen Fällen macht @Ignore Sinn.

1. Erläutern Sie die Ausführungsprozedur der JUint-Test-API-Methoden!  
   Im Folgenden wird gezeigt, wie das JUnit-Ausführungsverfahren funktioniert -

Die als @BeforeClass annotierte Methode wird nur einmal ausgeführt.

Zuletzt wird die mit @AfterClass annotierte Methode nur einmal ausgeführt.

Methode, die mit @Before annotiert ist, wird für jeden Testfall ausgeführt, jedoch vor der Ausführung des Testfalls.

Methode annotiert als @After wird für jeden Testfall, aber nach der Ausführung des Testfalls ausgeführt.

Zwischen der als @Before annotierten Methode und der als @After annotierten Methode wird jeder Testfall ausgeführt.