Konzept Software-Qualitätssicherung

Projektziel

Das Ziel unseres Programmierprojektes ist es ein Multiplayerspiel namens Edgy mit Java zu programmieren. Das Spiel soll eine möglichst einfach gehaltene und für dieses Spiel optimierte Client-Server Struktur besitzen. Ein Spieler wird sich anfangs des Spiels mit seinem eigenen Benutzernamen und Passwort anmelden beziehungsweise registrieren können. Das Regelwerk soll so kompakt wie möglich implementiert werden. Die grafische Oberfläche wird in 2D gestaltet und beinhaltet zusätzlich eine aktuelle Rangliste sowie einen Overall-score der beteiligten Spieler. Damit wir diese Ziele möglichst effizient erreichen können werden wir einige Massnahmen anwenden.

Massnahmen

1. Wir werden *sechs* Massnahmen ergreifen, die Qualität unseres Programmes aufrecht zu erhalten. Wir werden Javadoc verwenden um unseren Code bestmöglich, anhand einer ausführlichen Dokumentation, für jeden verständlich und nachvollziehbar zu machen. Javadoc wird wöchentlich aktualisiert und mit einem Vermerk «Javadoc» commited. Jedes Teammitglied ist angehalten seinen Code selbst zu kommentieren. Der QM-Manager generiert die JavaDocs vor jedem Gruppenmeeting am Montag und weist auf Verstösse hin.
2. Damit unser Code einheitlich gegliedert ist werden wir alle im «Google style» programmieren und haben Richtlinien definiert wie wir unseren Code gestalten. Um dies regelmässig überprüfen zu können haben wir in IntelliJ IDEA die Plugins Checkstyle sowie google-java-format Settings installiert, um alle kritischen Meldungen zu beheben. Dies wird wöchentlich überprüft und mit einem zugehörigen commit gepusht. Der QM-Manager überprüft den Code style und weist am Montag auf Verstösse hin.
3. Mit der Absicht die Lesbarkeit und unser Verständnis des Codes zu verbessern, werden wir kontinuierlich Code-Reviews durchführen und dem Partner konstruktive Feedbacks liefern und Code-Richtlinienverstösse im Plenum diskutieren. Diese werden jeweils mit einem zugehörigen Protokoll im File «ProtocolCodeReview.txt» im Quality Management Ordner dokumentiert und ins Repository gestellt. Zudem führt der Quality Manager jedes Wochenende einen Review des Codes durch und kommentiert Code Breaking Conventions, sowie unübersichtliche Code Abschnitte und ungenügend kommentierte Stellen. Diese Reviews werden mit «review» im Text kommentiert und mit demselben Vermerk ins Repository gestellt**.**
4. In unserem Code werden wir an allen nötigen Stellen ein Exception-Handling implementieren und somit ein gutes Feedback über den auftretenden Fehler erhalten. Damit wir in unserem Programm sehr spezifisch nach Bugs suchen können werden wir einen mehrstufigen Server-Logger verwenden**.** Die folgenden Log-Stufen wurden in unserem Code eingebaut: log.info(«msg»); printed Informationen, log.warn(«msg»); Warnungen, die Verlauf des Programms nicht beeinflussen, log.error(«msg»); Fehler die nicht auftreten sollten, jedoch das Programm nicht stoppen und log.fatal(«msg»); fataler Fehler bei dem das Programm nicht weiterlaufen darf.
5. Zwecks eines möglichst gut lesbaren Codes werden wir regelmässig in Zweiergruppen Code schreiben, um so auch voneinander profitieren und lernen zu können. Ebenfalls im Quality Management Ordner werden die in einem File «ProtocolPairprogramming.txt» protokolliert.
6. Ausgiebige Test-Routinen werden mit Hilfe der JUnit library vorgängig implementiert und unser Code anhand der Testvorgaben fortlaufend programmieren**.**

Gruppe 14-while (true)do nothing;

Um die Einhaltung unserer Produktqualität zu gewährleisten wurden Verantwortlichkeiten und Deadlines im Projektplan definiert. Damit die Code Qualität für die Milestones erneut überprüft werden kann, wurde zudem eine Checkliste «Checklist.txt» erstellt. Diese wird vor der Milestone-Abgabe abgearbeitet und vom Quality-Manager mit einem «Checklist approved» commitet.

Der Quality-Manager

Tobias