1. ソース管理（バージョン管理）とは

ソースファイルに対して

「誰が」「いつ」「何を変更したか」というような情報を記録することで、

過去のある時点の状態を復元したり変更内容の差分を表示できるようにするシステム

バージョン管理システムは大きく 2 つに分けると、

「集中管理方式」「分散管理方式」がある。

(ア)集中管理型 - Subversion

(イ)分散管理型 – Git

導入する利点は以下の 3 つ。

・アジリティ（柔軟性）の向上

・開発速度の向上

・コードの肥大化を抑制

2. Git とは

分散型のバージョン管理システム

ローカル環境（自分のパソコンなど）に、全ての変更履歴を含む完全なリポジトリの複製が作成される

ローカル環境にもコードの変更履歴を保存（コミット）することができるので、リモートのサーバーに常に接続する必要がない。

(ア)Git と Subversion の違い

上で説明したように集中型か分散型かが一番の違い。

(イ)GitHub

Git の仕組みを利用して、世界中の人々が自分の作品(プログラムコードやデザインデータなど)を保存、公開することができるようにしたウェブサービス

(ウ)GitFlow

Git のプラグインツール

ブランチ運用ルールを効率的に取り入れられる

3. 用語解説

(ア)リポジトリ（ローカル・リモート）

(イ)コミット・インデックス・ワークツリー

(ウ)ブランチ（master, develop, feature, release, hotfix）運用ルール

(エ)基本コマンド（pull, push, fetch, commit, merge, rebase, stash）

(オ)タグ

(カ)プルリクエスト

4. 基本的な流れ

(ア)リモートリポジトリ作成

(イ)ローカルリポジトリ作成

(ウ)GitFlow 開始

(エ)Feature ブランチ作成

(オ)ソースの追加・修正

(カ)コミット

(キ)ローカルテスト

(ク)Feature 完了

(ケ)Release ブランチ作成

(コ)Release ブランチを Push

(サ)リリース前テスト

(シ)Release ブランチ完了

(ス)Master ブランチ Push

5. コンフリクト(衝突)の解消方法

多人数での開発時、同時に同じソースを改修することがよくある

先に A さんが feature を完了し、次に B さんが feature を完了すると Develop ブランチにてコンフリクト(衝突)が起こる

コンフリクトの解決には

1. Rebase を利用した方法

2. Merge する方法

の 2 種類が存在する

6. PullRequest 活用

(ア)PullRequest とは

開発者が修正した箇所を、他の開発者に通知する機能

開発担当とマージ担当者に分担しコードレビューを行う場を提供する

(イ)導入のメリット

レビュー・マージ作業をタスク化して管理し、やりとりを記録できる

レビューを促進できる

(ウ)運用手順

Feature ブランチ作成

ソースコード改修

Feature ブランチ コミット

PullRequest 作成　⇒　マージ担当者

コードレビュー

Develop ブランチへマージ

7. 演習 1 - リポジトリの作成〜リリース完了まで

    自分の GitHub 環境にリモートリポジトリの作成

    ローカル環境に Clone

    GitFlow の開始

    Feature ブランチの作成

    前回の Java ソースを追加

    Feature ブランチの完了

    Develop ブランチを Push

    Release ブランチ作成・Push

    Release ブランチ完了

    Master ブランチ Push

8. 演習 2 - 多人数で一つのリポジトリを操作〜コンフリクトの解消

    リポジトリ https://github.com/anmut/sample_201807_001

9. まとめ