**課 題**   
GitHubを⽤いて開発を⾏う際の初期設定について、GitHubの機能について整理して  
ドキュメントにまとめること。また、GitHubでのプルからコミットまでの流れを  
キャプチャを⽤いて説明すること。また、コンフリクトとは何か、なぜ起こるのか  
具体的に説明してドキュメントにまとめること。

**初期設定**1.アカウントを作成する  
2.インストールを行う  
3.ユーザ情報の設定  
4.パーソナルアクセストークンを作成する

**GitHubの基本機能**

|  |  |
| --- | --- |
| 機能名 | 説明 |
| Repository | プロジェクトのコードやファイルを 管理する場所 |
| Branch | 開発作業を分岐させることで 複数の作業を同時に進行可能 |
| Commit | 変更内容を記録する操作 |
| Push | ローカルの変更を リモートリポジトリに反映 |
| Pull Request | 変更内容をレビューしてもらい メインブランチに統合する提案 |
| Merge | ブランチの変更を他のブランチに統合 |
| Issue | バグ報告や機能提案など タスク管理に使用 |
| Fork | 他のユーザーのリポジトリをコピーして 独自の開発を行う |
| Star | お気に入りのリポジトリに付けることで 後で簡単にアクセス可能 |

**プルからコミットまでの基本的な流れ**

**リポジトリのクローン**  
リモートリポジトリをローカルに複製する

**新しいブランチの作成と切り替え**  
作業用のブランチを作成し、切り替える

**コードの編集**

**変更のステージング**  
変更をステージに追加する

**コミットの作成**  
変更をコミットする

**リモートリポジトリへのプッシュ**  
変更をリモートリポジトリに反映する

**プルリクエストの作成**  
GitHub上で、プルリクエストを作成し、レビューを依頼する

**レビューとマージ**  
レビュー後、問題がなければメインブランチにマージする

**ローカルブランチの更新**  
メインブランチに切り替え、最新の状態を取得する

**コンフリクトとは？**

gitはmergeする際、変更箇所を自動的に統合してくれるが同じファイルの同じ場所への変更が2つのブランチにそれぞれある場合どちらを適用すればいいかまではgitは判断できず手動で解決する必要がある。

**なぜ起こるのか？**

理由は大きくいうと、merge、cherry-pick、rebase、pull 時に  
「同じ場所を異なる方法で変更した」（以下例）ことが原因となる。  
・同じ行の変更を複数のブランチで行った  
・同じファイルが異なるブランチで変更され、その内容に依存関係があった  
・同じファイルの削除と変更が同時に発生した。  
・ファイルを追加した一方でそのファイルが削除された  
・同じファイルが異なるブランチで移動、名称変更が行われた  
・ブランチ間で変更内容の履歴にギャップが生まれた  
・ feature-branchを先にmainにマージしその後逆にmainをfeature-branchにマージした

**解決手順**1.コンフリクトが発生したことを確認する  
2.コンフリクトが発生しているファイルを確認する  
3.コンフリクト部分を手動で解消する  
※手作業なので消さなくていいところや、マーカー部分の削除漏れに注意が必要。  
4. コンフリクト解消後のファイルをステージする  
5. コンフリクト未解消のファイルがないか確認  
6. コンフリクト解消のコミットを作成する