5.概念論述

1. オブジェクト指向とは

ある概念を持ったモノごとにクラスを設計して、モノとモノとの関係性を定義していくシステム構成の考え方です。クラスとは先程記述した概念の構成要素や処理をまとめた設計書になります。例えば「車」というモノをクラスにする場合、「車」のメーカーや色、車種は構成要素となり、「走る」「止まる」「曲がる」は処理となります。「車」という一つの概念を構成するために必要な項目を設計していきます。また、モノは必ず実体化しているものを示すわけではなく、”ルール”や SNS で使用される”つぶやき”のような形のないモノも対象となります。
クラスを実際に使用する場合には「インスタンス化」をして実体化してあげる必要があります。

オブジェクト指向には 3 つの特徴があります。

一つ目はカプセル化です。
カプセル化とはクラス内の処理(メソッド)や変数に、外部から簡単にアクセスできないよう、アクセス修飾子(アクセス権限)を定義して保護することです。同じクラス内の処理(メソッド)を介すとアクセスが可能となります。
例えば、年齢という変数に対して現在の年齢を代入し、アクセス権限を持たせるとします。ただこの変数は処理(メソッド)を介すことで変更が可能です。思い出という処理(メソッド)を作成したとして、私が中学生のころの年齢を処理(メソッド)の引数として渡して年齢を変更し、戻り値を変更後の値とします。処理(メソッド)を介しては変更できますが、直接年齢という変数に対してはアクセスできないようにしています。処理(メソッド)を介することで、不正なデータ操作を防ぐことができます。エラーやバグを起こさない仕組みを構築できます。

二つ目は継承です。
新しくクラスを作成する際に、共通項目を記述しているクラスを親として引き継いで、新しいクラスで利用できるようにすることです。新しいクラスを作成する時に、毎度同じ処理を実装すると修正や改良が発生した時に、全てのプログラムに対して実装変更を行う必要があり、修正し忘れ等の手間がかかってしまいます。クラスを継承することで、追加機能を実装する際に、同様のプラグラムが記載している部分に追記できますので、拡張性のあるプログラムが実現できます。
例えば「車」というクラスを作成します。「車」の「走る」「止まる」は基本性能になりますので、「車」クラスの中に「走る」「止まる」それぞれを処理(メソッド)として記述しておきます。
「車」の種類にはトラック、乗用車、レーシングカーと色々ありますが、「車」というモノの動作は基本同じになります。その場合、「車」クラスをそれぞれが継承することで、どのような種類の「車」がきても「車」クラスの同じ処理(メソッド)で行うことができます。それぞれのクラスで「走る」「止まる」

という基本性能の処理(メソッド)を記述する必要がなくなり、プログラムコードが少なくなります。ただし、継承できるクラスは 1 つとなります。

三つ目はポリモーフィズムです。
異なる動作を同じ処理(メソッド)で実現することをいいます。
例えば「買い物」という処理(メソッド)を作成します。この処理(メソッド)には'商品を買う'という内容が記述されています。今回 5 名のグループで買い物を行いました。5 名ともそれぞれ買った商品は異なりますが、商品を買ったという動作は変わりありません。'誰が'、'商品'という情報を「買い物」という処理(メソッド)にまとめて渡してあげることで、全員分の商品を買う動作を行うことができます。このように共通した動作を一つの処理(メソッド)で実現することをポリモーフィズムといいます。メリットとして、同じ処理内容を記述することがなくなるため、プログラムコードが少なく済み、共通処理の変更が発生した場合でも容易に行えます。

参考文献

https://webpia.jp/polymorphism/
https://www.sejuku.net/blog/9598

2. Github flow とは

GitHub の開発で使用されているワークフローになります。個人開発/チーム開発両方で使用されています。
ワークフローの基本的な流れは下記の通りです。
GitHub 内にリポジトリというプロジェクトのソースコード保管場所を作成できます。
リポジトリ内ではブランチという概念があり、多くのプロジェクトでは main ブランチに安定板ソースコードを保管します。WEB 上に配置されたリポジトリはリモートリポジトリといい、常に最新の安定版ソースコードになります。
新規機能や既存ソースコードの修正を行う場合は、main ブランチをクローン(複製)して開発用の topic ブランチ(ローカルリポジトリ)を作成します。その後、作成した topic ブランチ内でソースコードを追加・編集していきます。追加・編集が完了したらリポジトリに対してコミット(記録)してデータをプッシュします。プッシュしたデータを確認してプルリクエストを行います。
プルリクエスト時にマスター管理者にコードレビューの依頼をかけ、承認がおりたら main のソースコードとマージを行うという流れになります。

参考文献

https://www.modis.co.jp/candidate/insight/column_30

3. サーバサイドエンジニアとフロントエンジニアの違い

フロントエンジニアは Web デザインを元に、スマートフォンや Web ブラウザ上に表示される Web サイトの構築やカスタマイズを行います。Web デザインを担当するケースもありますが、Web サイト制作スキルを持ったエンジニアを指します。主に HTML/CSS/javascript 言語を使用します。

サーバサイドエンジニアは Web サイト(フロントエンド)に関連したデータの管理(データベース等)、各機能を使用する際の処理プログラムの制作を行うエンジニアを指します。サーバサイド言語はたくさんありますが、代表的なところでは PHP/java/C#/perl 等があります。

4. AWS とは

AWS とは Amazon Web Services の略で、Amazon が提供しているクラウドコンピューティングサービスの総称です。クラウドコンピューティングサービスとは Web を介してサーバー・ストレージ・データベース・ソフトウェアといったコンピュータを使った様々なサービスを利用することを指します。従来の物理サーバーを構築することがないため、自分の PC とインターネット環境があれば簡単にサーバー、大容量のストレージを使用することができます。

AWS では豊富なサービス(仮想サーバーやデータベース等)を提供しており、初期費用なしで必要な機能だけを使用することができます。もともと Amazon 社が使用していたサービスのため、セキュリティも最新に保たれています。

たくさんのサービスが提供されているため、自分が構築したい環境に対してどのサービスが最適なのかを明確にして使用する必要があります。

参考文献

https://www.skyarch.net/column/whataws01/

5. Docker とは、導入するメリット

コンピュータ内にコンテナ型の仮想環境を構築できるオープンソースソフトウェアを指します。Virtualbox 等のハイパーバイザ型に比べ、仮想環境を構築するのに直接アプリを起動するため、起動が速く、パソコンの CPU やメモリなどのリソースを最低限しか利用しないというメリットがあります。

その他、Docker を導入するメリットは、簡単に開発環境を構築できることです。構築手順は、Docker エンジンをインストールして、利用したいアプリの Docker イメージを入手し、コンテナを作成することで簡単に環境構築ができます。また、チーム開発を行うときに、共通のアプリやツールをインストールするだけで、全員同じ環境を構築することができます。

参考文献

https://xtech.nikkei.com/it/atcl/watcher/14/334361/020900182/