作成者:村上　秀隆

５. 概念論述

1. オブジェクト指向とは何かを述べてください
a. 特徴 3 つ
①カプセル化
②継承
③多態性（ポリモーフィズム）
各用語説明
① カプセル化とは・・
関連のあるデータとその操作をひとまとめにして、外部から直接内部データを操作することを禁止する代わりに、外部に操作の仕様だけを見せること。カプセル化のメリットは大きく分けて 2 つあり、1 つめは、オブジェクトを利用する人はその操作の仕様だけを知っていれば、オブジェクト内の操作の実装やデータの内容を知る必要がない。2 つめは、関連する操作がひとまとめになっているので、理解しやすく変更の影響も局所化できること。カプセル化するひとまとまりの単位のことを「クラス」、クラスの中の操作を「メソッド」、データのことを「フィールド」と呼ぶ。

② 継承とは・・
クラスに階層を持たせ、上位クラスのデータやメソッドを下位クラスが引き継ぐこと。下位クラスでは継承した性質の一部を変更したり追加したりすることができる。上位クラスを「親クラス」と呼び、下位クラスを「子クラス」と呼ぶ。継承は、全体のほとんどは一緒だけど一部が異なるフィールドとメソッドを何種類も書かなければいけないときに役に立つ。親クラスに共通的なフィールドとメソッドを書いておき、親クラスを継承した子クラスには異なる部分だけを記述する。

③多態性（ポリモーフィズム）とは・・
オブジェクト指向における「実行される処理の実体が、コール（呼び出す）されたメッセージではなく、メッセージを受けたオブジェクトによって決定される性質」のこと。
また、この性質を使って「同一のメッセージを使って、オブジェクトごとに異なった処理を行わせること」を指して、「多態性」という言葉が使われる場合もある。
b. 具体例を含めてください
カプセル化の具体例・・・
カプセル化を日常生活で例えるなら、パソコンがいい例である。パソコンを使用している多くの人は、どのような構造で作られているかを知らないはずである。しかし普通に

使用するぶんには、そんなことはまったく知らなくても良いが、しかしいざ壊れてしまったときや、ウィルスに感染したときにどうしたらいいのか解らなくなくなってしまう。実は大したことのないアクシデントであっても、知らないがゆえに深刻に捕らえてしまう。カプセル化であるならば、使用する側が、まったくパソコンの構造を理解していなくても便利に使用できる。しかし、カプセル化によって不測の事態が起こったときに、実際簡単に直せるものでも本人一人では、どうすることもできないという点がある。

継承の具体例・・・github 参照
クラスの継承によって、開発者は既存のクラスで指定されている変数やメソッドといった要素を取り込み、同じ記述を繰り返すことなく新しいクラスを作成することができる。
【継承の書式】
class サブクラス名 extends スーパークラス名 {}
クラスの継承をする場合、継承元をスーパークラス（親クラス）と呼び、継承先をサブクラス（子クラス）と呼んでいます。継承後、サブクラスではスーパークラスのメソッドや変数を使えるようになる。
与えられた顧客の氏名、住所、ID を出力するプログラムとなっており、CustomerInfo クラスは Customer クラスを継承しているため、CustomerInfo クラスで再び顧客の氏名と住所について記述する必要がなくなる。

多態性（ポリモーフィズム）の具体例・・・
「モノ」が「そのモノ」らしく振舞うことで、例えば犬と猫とアヒルがいた時、 この 3 匹はそれぞれ、どのように鳴くのか？ 犬は「ワン」と鳴き、猫は「ニャー」と鳴き、アヒルは「クワッ」と鳴く。そうした時、プログラミングでは「中に入るものによって同じ関数でも違う処理を行える」というプログラミング言語自体の特徴を活かすことができる。これを多態性という。

2. Github flow とは何かを述べてください
GitHub Flow は、ブランチの有名な運用方法の中の一つです。
ブランチとは、履歴の流れを分岐して記録していくためのものです。
分岐したブランチは他のブランチの影響を受けないため、同じリポジトリ中で複数の変更を同時に進めていくことができる。リポジトリは2種類あり、リモートリポジトリとローカルリポジトリがある。リポジトリはデータの保管場所であり、リモートリポジトリはサーバー上に置かれ一つしか存在せず、ローカルリポジトリは各作業者の PC に存在する。そのため、リモートリポジトリが main リポジトリとなる。
また、分岐したブランチは合流させることもでき、それをマージと言い、
GitHub ではリポジトリ作成時に main というブランチが自動作成され、

GitHub Flow では main ブランチを常時デプロイが可能であるブランチとする。
常時デプロイが可能であるとはどのような状態かというと、
常に安定しているブランチで、リリース可能な状態であるということ。
機能の追加やバグ修正を行いたい場合に main ブランチをいじってしまうと、
「常時デプロイが可能である」というルールを破ってしまうことになるので、
作業用のブランチを作成してそちらで作業をおこないます。
作業終了後に main ブランチにマージしデプロイするという一連の流れ。

3. サーバーサイドエンジニア・フロントエンジニアとはどのような違いがあるかを述べてください。
フロントエンドエンジニア = ユーザーが見ている画面のデザインをする
サーバーサイドエンジニア = サーバーでプログラムの実行・管理をする
フロントエンジニアが使う主な言語は、HTML/CSS、JavaScript（jQuery）
サーバーサイドエンジニアが使う主な言語は Ruby、PHP、Python、Java
見た目の部分を構築する仕事か、内部の部分を構築するかの違いと使用言語による違いがある。

4. AWS とは何ですか。特徴を述べてください。
2006 年に、Amazon 社内のビジネス課題を解決するために生まれた IT インフラストラクチャのノウハウをもとに、アマゾン ウェブ サービス（AWS）という名称でサービスがスタートし、世界で最も利用されているクラウドインフラサービス。
AWS の利用者（主に管理するエンジニア）は、AWS に主にサーバーまわりの最新資源の調達やパフォーマンスチューニング、そしてセキュリティ対策を任せることができるため、利用者は自社サービスの開発や改善など、コアビジネスに集中することができる。 一口に AWS といっても、その内部には様々なサービスが存在します。ネットワークや仮想サーバー、ストレージ等の IaaS と呼ばれるサービスから、データベースやアプリケーションプラットフォームなどの PaaS、メールやキューイングサービス等の SaaS までカバーしている。 AWS のサービスの特徴の 1 つに、例外はあるものの多くが初期費用不要の時間課金で提供されている。また、仮想サーバーのようにユーザがリソースを専有するものより、SaaS のようなアプリケーションサービスのリソースを複数ユーザが共用できるものの方が価格が低いと言う特徴がある。

5. Docker とは具体的に何ができる技術ですか。また Docker を導入するメリットを述べてください。

Docker は、インフラ関係や DevOps 界隈で注目されている技術の一つで、Docker 社が開発している、コンテナ型の仮想環境を作成、配布、実行ができる技術。

Docker は、Linux のコンテナ技術を使ったもので、よく仮想マシンと比較される。

VirtualBox などの仮想マシンでは、ホストマシン上でハイパーバイザを利用しゲスト OS を動かし、その上でミドルウェアなどを動かします。それに対し、コンテナはホストマシンのカーネルを利用し、プロセスやユーザなどを隔離することで、あたかも別のマシンが動いているかのように動かすことができるため、軽量で高速に起動、停止などが可能。

主なメリットは

① コード化されたファイルを共有することで、どこでも誰でも同じ環境が作れる。

② 作成した環境を配布しやすい。

③ スクラップ＆ビルドが容易にできる。

例えば、開発環境(Windows 上)では動いていたけど Linux で動かなかった、といったケースも、開発工程から Docker を活用していくことで防ぎやすくなる。そして、開発工程の中で使っていた環境をそのまま本番環境に持っていくことも可能なため、環境差分が少なく、環境による問題を減らすことができる。

作成した Docker イメージを他の人にも渡して使ってもらうことで、各自の環境のバージョンずれ防止や、開発環境準備の短縮化にもつながる。

また、クラスタ構成を構築する場合も、Docker イメージがあれば、それを起動する名前(コンテナ名)などを変えるだけで、複数の環境(コンテナ)を起動できるので、一から手順に沿って環境を作る作業もなくなり、クラスタ構成を構築するのも容易になる。

引用、参照：https://business.ntt-east.co.jp/content/cloudsolution/column-37.html
https://www.docker.com/what-docker
https://aiacademy.jp/media/?p=131
https://webpia.jp/polymorphism/
https://xtech.nikkei.com/atcl/nxt/column/18/00208/031300003/