1. オブジェクト指向とは何か、述べてください。

変数や関数をひとまとめにして、名前をつけてテンプレ化し、再利用できるようにする事で、オブジェクト（モノ）を中心に考えたプログラミングスタイルを指します。
つまりプログラムを手順ではなく「モノ」の作成や操作として見る考え方です。
また、ここで言う「モノ」とは、必ずしも形があるモノではありません。
概念（ルール等）を含め、モノとして認識する為、形のないものがどのようなふるまいをするのか考える必要があります。
なお、何を「モノ」として扱うかは、開発者次第とされています。
オブジェクト指向プログラミングは、下記のようなモノで構成されています。
・作成するモノの設計書＝クラス
・オブジェクトの「状態」を表すモノ＝プロパティ
・オブジェクトの「操作（ふるまい）」を表すモノ＝メソッド
また、クラスをもとに作成されたオブジェクトの実体のことをインスタンスと呼び、インスタンスを生成することをインスタンス化と言います。

例としてゲームのキャラクターをイメージしてみましょう。
キャラクター毎に特性があり、攻撃タイプ、守備タイプ、魔法タイプがあったとします。
まず制作するのは、全てのキャラクターの骨組みとなる初期キャラクターの設計書（クラス）を作ります。
その設計書（クラス）の中には、構成要素（プロパティ）とふるまい（メソッド）を書くことが出来ます。
プロパティとして挙げられるとするならば、名前や年齢、HPの数、勝利した際の決め台詞等です。
メソッドとしては、攻撃する　避ける　決め台詞を言う等です。
そして、作成した初期キャラクターのクラスを引き継いで、それぞれのキャラクターを制作していきます。
つまり、初期クラスを引き継いで、新しいクラスを作成します。
このように、クラスの内容を引き継ぐ考え方を、継承と呼びます。
攻撃タイプキャラを作りたい場合、初期クラスを継承する為、初期クラスに設定されている機能は全て使えます。
そして、攻撃タイプキャラにあったプロパティやメソッドを形成していきます。
このように、オブジェクト指向プログラミングでは、設計書を作成して、それを継承して編集していくことで、汎用性が高くプログラミングを施すことが出来ます。

【インスタンスとオブジェクトの違い - ITを分かりやすく解説】
https://medium-company.com/%E3%82%A4%E3%83%B3%E3%82%B9%E3%82%BF%E3%83%B3%E3%82%B9-%E3%82%AA%E3%83%96%E3%82%B8%E3%82%A7%E3%82%AF%E3%83%88-%E9%81%95%E3%81%84/
【GitHub Flow ～GitHubを活用するブランチモデル～】
https://jsstudy.hatenablog.com/entry/2017/03/10/172049

2. Github flowとは何か述べてください。

GitHub flowとは、GitHubの開発で使用されているワークフローの事を指します。

まず、Githubとはコードやデータを保存/公開できるプラットフォームのことです。
Gitを軸にしたプロダクト開発を支援してくれる機能が多く備わっています。
ソースコードやデザインデータを用意に共有することが出来、管理がしやすく、開発の効率を上げることが出来ます。
その為、個人・チーム問わず、多くの開発現場で利用されています。
Github flowは、その中で利用されている開発フローを指していますが、その流れは以下の通りです。
まず、Gitでの管理は、リポジトリ（貯蔵庫）というものを軸に開発を進めていきます。
そして、Githubは、リモートリポジトリというものを提供してくれます。
リモートリポジトリとは、制作するもののファイル群を保管する貯蔵庫のイメージです。
それぞれが編集したデータを、こちらに集約して制作物を作成します。
また、リポジトリにはもう一つあり、それをローカルリポジトリと言います。
これは、それぞれのPC上に作られるリポジトリのことです。
チーム開発において、ローカルリポジトリで作業をした内容を、リモートリポジトリにアップしていくことで、それぞれがリモートでの同時作業を実現します。
また、Github上では複数人が一度に手分けして作業しても、効率を下げない為に、ブランチと言うものが採用されています。
ブランチとは、ソースコードの履歴を枝分かれさせることが出来る機能です。
各ページ毎や各修正内容毎に、ブランチを分けることで、それぞれの作業に影響を与えず、効率良く作業することが出来ます。
それらの中心となるブランチがmainブランチです。
こちらは、その名の通りmainとなるブランチのことを指します。
本番環境運用のためのフィールドや編集結果が集まる場所となります。
つまり、ユーザーにプロダクトとして提供するソースコードを管理する中核となります。
このように、適切なGithub flowを踏むことで、より効率よく制作を進めることが出来ます。

【GitHub Flow ～GitHubを活用するブランチモデル～】
https://tracpath.com/bootcamp/learning_git_github_flow.html
【Git-flowをざっと整理してみた】
https://dev.classmethod.jp/articles/introduce-git-flow/
【GitHubとは？わかりやすく使い方や機能、インストールまで徹底解説！】
https://www.brain-gate.net/content/column/system-program-github/

3. サーバーサイドエンジニア・フロントサイドエンジニアとはどのような違いがあるか述べてください。

フロントサイドエンジニアとは、画面の表示やサイト上での入力情報を、サーバーに伝える箇所を主に担うエンジニアです。
表示やデザインを作成して、ユーザーが分かりやすく使いやすいページを作成します。
ユーザーの操作により、サーバーに情報を提供して、必要な情報を要求する役割を担います。
つまり、ユーザーとサーバーとの仲介窓口と言えます。
一方、サーバーサイドエンジニアは、フロントサイドからの要求に対して、適切な情報を送信する役割を担うエンジニアです。

フロントサイドエンジニアのUIによって、リクエストされてきたユーザーの要求に対して、適切な情報を返す必要があります。
Webサイト閲覧の流れを含めたまとめは、以下の通りです。
一般的な、WebサイトやWebアプリケーションの動作の流れは、フロントエンドとサーバーサイドに分類されます。
これらを担うエンジニアを、上記でも説明した通り、それぞれフロントサイドエンジニア・サーバーサイドエンジニアと言います。
フロントサイドエンジニアとは、Webサービスやアプリケーションにおいて、直接ユーザーが接することが多い部分、つまり、ページの外観を担う役割を構築するエンジニアです。
フロントエンドとは、画面の表示やサイト上での入力情報を、サーバーに伝える役割を果たします。
私たちが目にする、Webページでの表示やデザインを的確に構築し、必要な情報をサーバーに要求します。
対して、受け取った要求を、的確に処理をして返す役割を担う部分を構築するのが、サーバーサイドエンジニアです。
Webページ上における、『〇〇ページを表示するボタンをクリック→〇〇ページを表示』という一見簡単そうな動作は、
『〇〇ページを表示するボタンをクリック→フロントエンドにより、「〇〇のページ」を表示するようにサーバーに要求→サーバーサイドにより、要求されたページを返す→〇〇ページを表示』といった複雑な作業が行われています。
プログラミングには、様々な言語があり、言語によって得意分野と不得意分野が異なります。
よって、フロントサイドエンジニア、サーバーサイドエンジニアが利用している言語が異なります。

【フロントサイドエンジニアとは？フロントサイドエンジニアに必要な6つのスキル】
https://www.acrovision.jp/career/?p=2826
【サーバーサイドってなに？サーバーサイドエンジニアの仕事内容と使う言語7つ】
https://www.acrovision.jp/career/?p=2804

4. AWSとは何ですか。特徴を述べてください。

〇AWSとは、Amazonが提供する、クラウドコンピューティングサービスです。
クラウドコンピューティングとは、コンピューティングサービスに必要な機能がインターネット上のサーバーに提供されており、それを利用する形態のことを言います。
つまり、AWSでは、PCやインターネット環境があれば、サーバーや大容量ストレージ・高速なデータベース等を、スピーディー且つ手軽にご利用することが出来ます。
メリットとしては、以下の4つが挙げられます。
1つ目はサービス内容です。役100以上のサービスを搭載し、それぞれを組み合わせることで、柔軟に様々なサービスを利用することが出来ます。
2つ目はコスト面です。初期費用はかからず、使わないサービスにコストが発生する等の心配はありません。また、サービスの追加や拡大が可能な為、汎用性高くご利用出来ます。
3つ目はセキュリティ面です。AWSは、Amazonが元々自社のサーバーとして運用

していたサービスの為、常に最新のセキュリティが施されています。
4つ目はパフォーマンス面です。世界中のデーターセンターで稼働しており、常に最新世代へとアップデートされています。その為、質は高いと言えます。
上記理由により、近年は多くの開発現場で使われています。
また、クラウド市場シェア率はトップといった特徴を持っており、信頼出来るクラウドコンピューティングサービスと言えるでしょう。

【AWS の クラウドが選ばれる 10 の理由 | AWS】
https://aws.amazon.com/jp/aws-ten-reasons/

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

Dockerとは、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です。
まず、仮想環境とは、自分のパソコンやサーバとは別に作り出した環境のことを指し、バージョンやOSの相違に柔軟に対応することが出来ます。
例えば、Mac環境の中にMacOSがある中で、仮想的にLinuxOSやWindowsOSを用意することが出来ます。
よって、仮想環境を構築することで、LinuxOSやWindowsOSのPCをそれぞれ用意する必要がなくなり、一つのPC上で一元管理することが出来ます。
なお、土台のOSのことを「ホストOS」、仮想環境上のOSのことを「ゲストOS」と言います。
そして、コンテナ型とは、あるアプリで必要な環境をOSレベルでパッケージ化してまとめた箱のようなイメージです。
基本的にホストOSの中核を利用しています。つまり、ホストOS上で、足りないものだけをコンテナとして補っていると言うことです。
そうすることで、メモリやCPUのリソースを抑えて運用することが出来ます。
このコンテナを作成する為には、設計書が必要です。これをDockerイメージと言います。
また、このDockerイメージを保存する場所を、Dockerレジストリと呼びます。
このDockerレジストリから、Dockerイメージを移動して、自作のDockerイメージへ編集し、これを起動することでコンテナが立ち上がります。
したがって、Dockerイメージがあれば、誰でも簡単に環境構築することが出来ます。
Dockerイメージを使い回しが容易なことから、汎用性が高く、様々な現場で利用されています。

【Dockerとは | OSSのデージーネット】
https://www.designet.co.jp/faq/term/?id=RG9ja2Vy