|  |  |  |
| --- | --- | --- |
| **OOP – Object Oriented Programming 1/2** | | |
| Koncept | Formål | Beskrivelse |
| 1. Grundlæggende klasser og objekter | Definere skabeloner for objekter og deres adfærd. | Klasser er skabeloner, der definerer, hvordan objekter skal oprettes og opføre sig. Objekter er forekomster af klasser. |
| 1. Constructors og tilstand | Initialisere objekter og sætte deres tilstand. | Constructors er metoder, der kaldes ved objektets oprettelse. De initialiserer objektets felter og tilstand. |
| 1. Mutation og tilstand | Ændre objekters tilstand efter oprettelsen. | Mutation refererer til ændringer i objektets tilstand, f.eks. ved at ændre værdierne af dets felter. |
| 1. Objekter har data og “behavior” | Objekter indeholder både data (felter) og metoder (adfærd). | Objekter repræsenterer en entitet og indeholder information (data) samt funktionalitet (metoder). |
| 1. Princippet om indkapsling | Skjule objektets interne detaljer og give adgang via metoder og egenskaber. | Indkapsling øger sikkerheden og gør koden mere vedligeholdelig ved at begrænse direkte adgang til objektets felter. |
| 1. Princippet om komposition (aggregation) | Oprette komplekse objekter ved at kombinere mindre objekter. | Komposition er en form for relation mellem klasser, hvor et objekt indeholder andre objekter. |

|  |  |  |
| --- | --- | --- |
| **OOP – Object Oriented Programming 2/2** | | |
| 1. Princippet om arv (nedarving) | Arve egenskaber og metoder fra en anden klasse. | Arv tillader genbrug af kode og oprettelse af hierarkier. En underklasse (barn) arver fra en overklasse (forælder). |
| 1. Princippet om polymorfi | Objekter af forskellige klasser kan behandles ens, hvis de deler en fælles baseklasse. | Polymorfi giver mulighed for at arbejde med objekter af forskellige typer på en ensartet måde. |
| 1. Abstrakte typer (klasser og interfaces) | Definere kontrakter, som andre klasser kan implementere. | Abstrakte klasser og interfaces definerer metodesignaturer uden implementering. Abstrakte klasser kan have fælles funktionalitet, mens interfaces er rene kontrakter. |
| 1. Generics <T> | Oprette generiske klasser og metoder, der kan arbejde med forskellige datatyper. | Generics tillader fleksibilitet ved at skabe genbrugelig kode, der kan tilpasses forskellige datatyper. |
| 1. Virtual og override | Tillade metoder at blive overskrevet i arvshierarkier. | Virtual markerer en metode, der kan overskrives i en underklasse. Override implementerer metoden i den specifikke underklasse. |
| 1. Constructor chaining (base()) | Kalde en constructor fra en anden constructor i samme klasse eller i en baseklasse. | Constructor chaining bruges til at initialisere objekter ved at kalde forskellige constructors i en bestemt rækkefølge. |

|  |  |  |
| --- | --- | --- |
| **Systemudvikling** | | |
| Koncept | Formål | Beskrivelse |
| 1. Seperation of concerns (OOP princippet om abstraktion og modularitet) | Opdele koden i logisk adskilte dele for bedre vedligeholdelse og modularitet. | Separation of concerns handler om at opdele koden i forskellige ansvarsområder, f.eks. dataadgang, forretningslogik og brugergrænseflade. |
| 1. Selve processen (Inception, Elaboration, Construction og Testing) | Hele processen fra kravspecifikation til test og implementering af software. | Systemudvikling dækker hele livscyklussen for softwareudvikling, herunder krav, design, implementering og test. |
| 1. Versionstyring med git/github | Spore ændringer i kildekoden og samarbejde om projekter. | Git og GitHub bruges til at spore ændringer i kildekoden og muliggøre samarbejde mellem udviklere. |
| 1. Designmønster: Generic Repository med EF | Abstrahere dataadgang i applikationer ved hjælp af Entity Framework. | Generic Repository er et designmønster, der giver en ensartet måde at arbejde med dataadgang på. Det bruges ofte sammen med Entity Framework (EF) til at håndtere CRUD-operationer på databaser. |

|  |  |  |
| --- | --- | --- |
| **UI/UX med .NET MAUI** | | |
| Koncept | Formål | Beskrivelse |
| 1. UI/UX | Forbedre brugeroplevelsen ved at designe intuitive og æstetisk tiltalende grænseflader. | UI/UX står for brugergrænseflade og brugeroplevelse. Det handler om at skabe en god interaktion mellem brugeren og softwaren. |
| 1. .NET MAUI | Skabe platformsovergribende apps med .NET. | .NET MAUI (Multi-platform App UI) er et framework til at bygge apps, der fungerer på tværs af iOS, Android og Windows ved hjælp af .NET. |
| 1. MVVM | Adskille præsentation, logik og data i applikationen. | MVVM (Model-View-ViewModel) er en arkitekturmodel, der hjælper med at organisere kode i WPF, Xamarin og .NET-applikationer. |
| 1. XAML | Deklarativt sprog til at designe brugergrænseflader. | XAML (Extensible Application Markup Language) bruges til at beskrive brugergrænseflader i .NET-applikationer. |