|  |  |
| --- | --- |
| **Curso:** Mobile Android | **Unidade e módulo: U01M01** |
| **Matéria da aula: Programação Orientada a Objetos 3** | **Duração da aula: 3h30min** |
| **Professor:** | **Professor:** |
| **Especialista:** | |
| **Assistente:** | **Assistente:** |
| **Link para o módulo no Github:** <https://github.com/digital-house/clasesMobile/tree/master/Unidad%2001%20-%20Programacion%20Orientada%20a%20Objetos/Modulo%2001_%20Diseno> | |

|  |
| --- |
| OBJETIVOS |
| * O aluno deve conseguir avaliar um design e justificar por que está correto. * O aluno deve conseguir elaborar suas próprias estratégias para transformar um diagrama de classes em código. |

|  |
| --- |
| CONTEÚDOS |
| * Revisão dos pilares da Programação Orientada a Objetos. * Contexto de design. * Diagramas de classes. |

# DESCRIÇÃO DA AULA:

|  |  |
| --- | --- |
| **Início da aula** | **Duração (5 – 10 minutos)** |
| **Como os objetivos serão apresentados:**  Apresentar os objetivos de maneira explícita. |  |
| **Atividade ou pergunta motivadora:**  Do que vocês se lembram sobre a Programação Orientada a Objetos? Quais são os pilares? O que significa cada um? | **5 minutos** |

|  |  |
| --- | --- |
| **Desenvolvimento da aula** | **Duração** |
| **Revisão sobre a Programação Orientada a Objetos** |  |
| **Matéria vista na aula anterior:**   * Programação em JAVA com objetos (Herança). |  |
| **Matéria:**   * O que é um objeto? * Quais são os pilares desse paradigma? * Encapsulamento. * Polimorfismo. * Abstração. * Herança. * Interfaces. * Coesão e acoplamento. | **20 minutos** |
| **Resumo da matéria apresentada:**  Onde começa e onde termina o problema a resolver. Como descrever esse problema de forma que o cliente possa entender. Revisar e reforçar os princípios básicos do paradigma. |  |
| **Perguntas para verificar a compreensão:**   * Pedir para os alunos darem um exemplo concreto em que seja possível observar cada um dos pilares (encapsulamento, herança, polimorfismo, abstração). |  |
| **Prática** |  |
| Apresentar o exercício do Facebook. Explicar o enunciado aos alunos e pedir para eles criarem o modelo UML.  <https://docs.google.com/a/digitalhouse.com/document/d/1exWOWLIT-GfMLH_Vn3_tsNL0PETQhmPKZSgdGse9xcQ/edit?usp=drive_web> | **30 minutos** |
| **Reflexão sobre o que foi trabalhado na prática curta:**  Pedir para várias equipes apresentarem suas soluções. Analisar as soluções e pedir para os alunos justificarem as decisões que tomaram. Pedir também para eles mostrarem onde seria possível aplicar o polimorfismo. |  |
| **Introdução a conteúdo novo** |  |
| **Matéria:**  Revisão rápida de herança em JAVA.   * O que utilizamos para indicar que uma classe é subclasse de outra? * Qual é o modificador de visibilidade utilizado para que os atributos possam ser herdados? * Os métodos do pai sempre são herdados? * O que é overriding? |  |
| **Resumo da matéria apresentada:**  Falar sobre os temas rapidamente. Se necessário, mostrar o código no IntelliJ. | **15 minutos** |

|  |  |
| --- | --- |
| **Prática** |  |
| Apresentar o próximo exercício. A ideia é que os alunos criem o design e, em seguida, desenvolvam o código.  <https://docs.google.com/document/d/1n0nEjZ1uDXXgEi0NC9k6YBlkHk-7CTSeG04aEE7-NEQ/edit>  Se necessário, apresentar os seguintes exercícios adicionais:  <https://docs.google.com/document/d/19-qfIvylChwyQD68jHFKK9VjoaGrqU9daMw1NFH3JgQ/edit> | **135 minutos** |
| **Reflexão sobre o que foi trabalhado na prática curta:**  Antes que os alunos comecem a implementar o código de cada exercício, conferir se o design realizado está correto.  Abrir uma discussão sobre cada exercício e pedir para várias equipes apresentarem suas soluções. Analisar as soluções e pedir para os alunos justificarem as decisões que tomaram. Pedir também para eles mostrarem onde seria possível aplicar o polimorfismo. |  |

|  |  |
| --- | --- |
| **Encerramento da aula** | **Duração** |
| **Atividade de encerramento:**  **Fazer uma recapitulação de todos os conceitos vistos.** | **5 minutos** |