Alunos:Daniel Costa - Lucas Santos - Lucas Cardoso

Nossa escolha: Singleton

|  |
| --- |
| O padrão do Singleton permite que e uma classe que permitira que  apenas uma única instancia seja criada. |
| Diagrama |
| Public sealed class Servidor {  private static Servidor instance = null; private Servidor(){ }  public static Servidor Instance{ get{  if(instance == null){  instance = new Servidor();  }  Return instance;  }  }  } |

FactoryMethod

|  |
| --- |
| O padrão do FacotryMethod transfere a responsabilidade de criação  para outra interface. |
| Diagrama  Diagrama  Descrição gerada automaticamente |
| public abstract class Hamburgueria {  abstract public Hamburguer fabricarHamburguer();  }  public class HamburgueriaIndustrialSadia extends Hamburgueria{ public Hamburguer fabricarHamburguer(){  return new HamburguerIndustrial();  }  }  public class HamburguerdoToninho extends Hamburgueria{ public Hamburguer fabricarHamburguer(){  return new HamburguerCaseiro();  }  } |

|  |
| --- |
| O padrão do adpter pode converter uma interface para outro como se  fosse um adaptador, permitindo o funcionamento da mesma. |
| Diagrama |

|  |
| --- |
| class AdaptadorComputadorPositivo implements Computador {  private ComputadorPositivo Computador;  AdaptadorComputadorPositivo (ComputadorPositivo computador) { this.computador = computador;  }  public void liga() { projetor.turnOn();  }  } |

|  |
| --- |
| O padrão strategy permite que o usuario encapsule os algoritimos  tornando os intercambiveis . |
| Diagrama |

|  |
| --- |
| Public Class Calculadora(funcao){ Public operação()  Somar(){  Return input 1 + 2 Subtrair(){  Return input 1 – 2 Divisao(){  Return input 1 / 2  }  }  Strategy calculadora (){  Printf(“(calculadora(funcao))’);  } |