|  |  |
| --- | --- |
| **ChainOfResponsibilityBank** | |
| **Line** | **Code** |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53**  **54**  **55**  **56**  **57**  **58**  **59**  **60**  **61**  **62**  **63**  **64**  **65**  **66**  **67**  **68**  **69**  **70**  **71**  **72**  **73**  **74**  **75**  **76**  **77**  **78**  **79**  **80**  **81**  **82**  **83**  **84**  **85**  **86**  **87**  **88**  **89**  **90**  **91**  **92** | **using** System;  **using** System.Collections.Generic;  **namespace** ChainOfResponsibilityBank{  **class** Program{  // Sets up the Handlers as level based structure  // Makes use of a user defined exception if the end of the chain is reached  **class** Handler{  Level level;  **int** id;  **public** Handler(**int** id, Level level){  **this**.id = id;  **this**.level = level;  }  **public string** HandleRequest(**int** data){  **if** (data < structure[level].Limit){  **return** "Request for " + data + " handled by " + level + " " + id;  }  **else**  **if** (level > First){  Level nextLevel = --level;  **int** which = choice.Next(structure[nextLevel].Positions);  **return** handlersAtLevel[nextLevel][which].HandleRequest(data);  }  **else**{  Exception chainException = **new** ChainException();  chainException.Data.Add("Limit", data);  **throw** chainException;  }  }  }  **public class** ChainException:Exception{  **public** ChainException(String msg) : **base**(msg) { }  **public** ChainException(){ }  }  **enum** Level { FrontDesk, Supervisor , Manager}  **static** Random choice = **new** Random();  **static** Level First{  **get** { return ((Level[])Enum.GetValues(typeof(Level)))[0]; }  }  **static** Dictionary<Level, Structure> structure =  **new** Dictionary<Level, Structure> {  {Level.Manager, new Structure {Limit = 9000, Positions = 1}},  {Level.Supervisor, new Structure {Limit = 4000, Positions = 3}},  {Level.FrontDesk, new Structure {Limit = 1000, Positions = 10}}};  **static** Dictionary<Level, List<Handler>> handlersAtLevel =  **new** Dictionary<Level, List<Handler>> {  {Level.Manager, new List <Handler>()},  {Level.Supervisor, new List <Handler>()},  {Level.FrontDesk, new List <Handler>()}};  **class** Structure{  **public int** Limit { get; set; }  **public int** Positions { get; set; }  }  **static void** Main(**string**[] args){  Console.WriteLine("Trusty Bank opens with");  **foreach** (Level level **in** Enum.GetValues(**typeof**(Level))){  **for** (**int** i = 0; i < structure[level].Positions; i++){  handlersAtLevel[level].Add(**new** Handler(i, level));  }  Console.WriteLine(structure[level].Positions + " " + level +  "(s) who deal up to a limit of " + structure[level].Limit);  }  Console.WriteLine();  **int**[] amounts = { 50, 2000, 1500, 10000, 175, 4500, 2000 };  **foreach** (**int** amount **in** amounts){  **try**{  **int** which = choice.Next(structure[Level.FrontDesk].Positions);  Console.Write("Approached FrontDesk " + which + ". ");  Console.WriteLine(  handlersAtLevel[Level.FrontDesk][which].HandleRequest(amount));  }  **catch** (ChainException e){  Console.WriteLine("\nNo facility to handle a request of " +  e.Data["Limit"]+".Try breaking it down into smaller requests");  }  }  Console.ReadKey();  }  }  } |