НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ ім. Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №2

з дисципліни

«Об’єктно-орієнтоване програмування»

ТЕМА: «C# .Net. Розширені можливості реалізації ООП у мові C#. Події.»

Підготував: студент групи КП-51

Волощенко Олександр Євгенович

Перевірила:

Заболотня Тетяна Миколаївна

Київ 2016

**Мета роботи**

Ознайомитися з такими можливостями мови програмування C# як абстрактні

класи, інтерфейси, делегати. Вивчити механізми обробки подій у C#.

**Постановка задачі**

Для ієрархії класів, побудованої в лабораторній роботі No1, реалізувати:

1. Механізм інтерфейсів. При чому один з класів повинен реалізовувати щонайменше 2 інтерфейси.
2. Абстрактний клас. Забезпечити його наслідування.
3. Механізм «делегат – подія – обробник події».
4. Механізм створення та обробки власних помилок:

* створити новий клас виключної ситуації;b. створити новий клас аргументів для передачі їх до обробника виключної ситуації;
* забезпечити ініціювання створеної виключної ситуації та продемонструвати, як працює обробник даної помилки.

При виконанні завдань лабораторної роботи скористатися типом даних

Generic<T>

**Лістинг програми**

|  |
| --- |
| Program.cs |
| using System;  using System.Threading;  namespace Lab2  {  class MainClass  {  public static void Main()  {  var bird1 = new Bird("Chiki");  var dog1 = new Dog("Spike");  int age;  string name;  Watchman watchman;  Console.WriteLine("Input watchman name:");  name = Console.ReadLine();  Console.WriteLine("Input watchman age:");  age = Int32.Parse(Console.ReadLine());  Centaur centaur = new Centaur("Heran", 112);  watchman = new Watchman(name, age);  Zoo<IAnimal> zoo = Zoo<IAnimal>.getInstance("ZOO");  zoo.addGuardian(centaur);  try  {  zoo.addAnimal(bird1);  }  catch(ZooException ex)  {  exceptionMessage(ex);  }  while (true)  {  try  {  zoo.addWatchman(watchman);  break;  }  catch (WatchmanException ex)  {  exceptionMessage(ex);  }  finally  {  if (watchman.Name.Length < 2 || watchman.Name.Length > 10)  {  Console.WriteLine("Input watchman new name:");  watchman.Name = Console.ReadLine();  }  else if (watchman.Age < 18)  {  Console.WriteLine("Input watchman new age:");  watchman.Age = Int32.Parse(Console.ReadLine());  }  else if (watchman.IQ < 110)  {  Console.WriteLine("Hey, make BrainStorm");  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  while (watchman.IQ < 110)  {  watchman.BrainStorm();  Console.WriteLine("Now watchmans IQ is {0}!\n//Press any key to continue ", watchman.IQ);  Console.ReadKey();  }  Console.WriteLine("\*\*Now watcher is clever, mb...\*\*\*\*\*\*\*\*\n\n");  }  }  }  Horse horse1 = new Horse("Sara");  Horse horse2 = new Horse("Johny");  try  {  zoo.addAnimal(horse1);  zoo.addAnimal(horse2);  zoo.addAnimal(bird1);  zoo.addAnimal(dog1);  }  catch(ZooException ex)  {  exceptionMessage(ex);  }  zoo.Filter();  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*EVENT\*\*\*\*\*\*\*\*\*\*");  watchman.Feeding();  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");  zoo.staffBrain();  zoo.staffTrain();  }  static void exceptionMessage(Exception ex)  {  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.WriteLine("ERROR: {0}", ex.Message);  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");  }  }  } |

|  |
| --- |
| MyExceptions.cs |
| using System;  namespace Lab2  {  class AnimalException:Exception  {  public AnimalException(string message)  :base(message)  {}  }  class WatchmanException:Exception  {  public WatchmanException(string message)  :base(message){}  }  class ZooException:Exception  {  public ZooException(string message)  :base(message)  {}  }  } |

|  |
| --- |
| Zoo.cs |
| using System;  using System.Threading;  using System.Collections.Generic;  namespace Lab2  {  public class Zoo<T>  where T: IAnimal  {  static Zoo<T> instance = null;  Watchman watchman = null;  Centaur guardian = null;  readonly string name;  readonly LinkedList<T> animals;  Zoo(string name)  {  this.name = name;  animals = new LinkedList<T>();  }  public static Zoo<T> getInstance(string name)  {  if (instance == null)  instance = new Zoo<T>(name);  return instance;  }  public void addWatchman(Watchman w)  {  if (w.Name.Length < 2)  {  throw new WatchmanException("Watchman name is too short!");  }  else if (w.Name.Length > 10)  {  throw new WatchmanException("Watchman name is too long!");  }  else if (w.Age < 18)  {  throw new WatchmanException("Watchman is too young!");  }  else if (w.IQ < 110)  {  throw new WatchmanException("Stupid Watchman!");  }  else  {  watchman = w;  Console.WriteLine("In Zoo \"{0}\" new Watchman: {1}!", name, watchman.Name);  }  }  public void addGuardian(Centaur c)  {  guardian = c;  }  public string Name  {  get{ return name; }  }  public int Count  {  get{ return animals.Count; }  }  public bool addAnimal(T animal)  {  if (watchman == null)  throw new ZooException("Zoo don't have watchman!");  else  {  watchman.WatchmanFeedingEvent += animal.Feed;  animals.AddLast(animal);  return true;  }  }  public bool removeAnimal(T animal)  {  if (watchman == null)  throw new ZooException("Zoo don't have watchman!");  else  {  watchman.WatchmanFeedingEvent -= animal.Feed;  return animals.Contains(animal) && animals.Remove(animal);  }  }  public void Filter()  {  int bird, cat, dog, fox, horse;  bird = cat = dog = fox = horse = 0;  foreach (T an in animals)  {  string type = an.GetType().ToString().Remove(0, 5);  switch(type)  {  case "Bird":  bird++;  break;  case "Cat":  cat++;  break;  case "Dog":  dog++;  break;  case "Fox":  fox++;  break;  case "Horse":  horse++;  break;  }  }  Console.WriteLine("In Zoo \""+this.name+"\" now live\n"+  +bird+" birds, "+cat+" cats, "+dog+" dogs, "+fox+" foxes and "+horse+" horses!");  }    }  } |

**Діаграма класів**

