namespace lab5

{

partial class Bouquet

{

public List<Flower> plants = new List<Flower>();

public Flower bouquet

{

get;

set;

}

public static int count = 0;

public int AddToList(Flower obj)

{

plants.Add(obj);

count++;

return count;

}

public int RemoveFromList(Flower obj)

{

int reCount = 0;

foreach (Flower a in plants)

{

if (a == obj)

{

reCount++;

break;

}

}

if (reCount >= 1)

{

bool norm2 = plants.Remove(obj);

if (norm2 == true)

{

count--;

return count;

}

else return count;

}

else return count; }

public void PrintList()

{

foreach (Flower a in plants)

{

Console.WriteLine(a);

} }}}

namespace lab5

{

class FlowerException : Exception

{

public FlowerException(string message) : base(message)

{

Console.WriteLine("Вызвано исключение " + message);

}

}

class RoseException : FlowerException

{

public RoseException(string message) : base(message)

{

}}}

namespace lab5

{

partial class Flower : Plant, IWatering

{

public string colorOfFlower;

int ammountFlower;

Bouquet bouquet;

public Flower() : base()

{

colorOfFlower = "red";

ammountFlower = 1;

bouquet = new Bouquet();

}

public string ListikiForms

{

get { return listikiForms; }

set

{

if (int.TryParse(value, out int num) == true)

throw new FlowerException("FlowerException");

else

listikiForms = value;

}}

public override string ToString()

{

return base.ToString() + " " + "Color: " + colorOfFlower + " " + "Ammount: " + ammountFlower + " " + "Form: " + ListikiForms;

}

public override bool Equals(object obj)

{

if (obj == null) return false;

if (obj.GetType() != this.GetType()) return false;

Flower flower = (Flower)obj;

return (this.colorOfFlower == flower.colorOfFlower && this.ammountFlower == flower.ammountFlower);

}

}

}

static void Main(string[] args)

{

Flower floue = new Flower();

floue.Size = 5;

floue.colorOfFlower = "yellow";

floue.ListikiForms = "rounded";

Flower floue2 = new Flower();

floue2.Size = 8;

floue2.colorOfFlower = "green";

Rose rose = floue as Rose;

if (rose == null)

{

Console.WriteLine("Преобразование прошло неудачно");

}

else

{

Console.WriteLine(rose.barbed);

}

Flower floue3 = new Flower();

floue3.Size = 2;

if (floue3 is Cactus)

{

Cactus cactus = (Cactus)floue3;

Console.WriteLine(cactus.needle);

}

else

{Console.WriteLine("Преобразование не допустимо");}

Verse(Poem.first\_line);

Verse(Poem.second\_line);

Verse(Poem.third\_line);

Verse(Poem.fourth\_line);

Tree tree1;

tree1.sort = "oak";

tree1.age = 25;

tree1.DisplayInfo();

Bouquet bouqet = new Bouquet();

int answer;

answer = bouqet.AddToList(floue2);

try

{

floue.ListikiForms = "3";

}

catch (FlowerException ex)

{

Console.WriteLine($"Исключение #1 поймано", ex);

}

finally

{

Console.WriteLine("Блок 1 finally отработал");

}

try

{

rose1.Barbed = 101;

}

catch (RoseException ex)

{

Console.WriteLine($"Исключение #2 поймано", ex);

}

finally

{Console.WriteLine("Блок 2 finally отработал");}

try

{

int a;

a = tree1.age / 0;

}

catch(DivideByZeroException ex)

{

Console.WriteLine($"Исключение #3 DivideByZeroException поймано", ex );

}

finally

{Console.WriteLine("Блок 3 finally отработал");}

int[] aa = null;

Debug.Assert(aa != null, "Values array cannot be null");} }

namespace pac

{

interface IFigure

{

void Print();

}

[Serializable]

public class Rectangle : IFigure

{

public int x;

public int y;

public int h;

public int l;

public string color;

public int square;

public virtual void Print()

{

Console.WriteLine($"Прямоугольник с шириной {l} и высотой {h} цвета {color}");

}

public Rectangle()

{

x = 20;

y = 40;

h = 10;

l = 15;

color = "green";

}

public Rectangle(int x, int y, string color)

{

this.x = x;

this.y = y;

this.color = color;

}

public Rectangle(int x, int y, int h, int l, string color) : this(x, y, color)

{

this.h = h;

this.l = l;

}

public override string ToString()

{

return base.ToString() + " " + this.x + " " + this.y + " " + this.h + " " + this.l + " " + this.color;

}

public static Rectangle operator +(Rectangle rect, int i)

{

rect.h += i;

rect.l += i;

return rect;

}

public int Square(int h, int l, int square)

{

square = this.h \* this.l;

return square;

}

}

class Program

{

static void Main(string[] args)

{

List<Rectangle> rectangles = new List<Rectangle>(6);

rectangles.Add(new Rectangle() { x = 40, y = 45, l = 50, h = 60, color = "green" });

rectangles.Add(new Rectangle() { x = 50, y = 35, l = 60, h = 70, color = "blue" });

rectangles.Add(new Rectangle() { x = 60, y = 55, l = 70, h = 80, color = "pink" });

rectangles.Add(new Rectangle() { x = 20, y = 5, l = 30, h = 40, color = "orange" });

rectangles.Add(new Rectangle() { x = 10, y = 50, l = 20, h = 30, color = "red" });

rectangles.Add(new Rectangle() { x = 30, y = 25, l = 40, h = 50, color = "yellow" });

foreach (Rectangle p in rectangles)

{

p.Print();

p.h = p.h + 1;

p.l = p.l + 1;

p.Square(p.h, p.l, p.square);

Console.WriteLine($"{p.h}, {p.l}");

}

var selectbyx = from Rectangle in rectangles

orderby Rectangle.x

select Rectangle;

foreach (Rectangle x in selectbyx)

Console.WriteLine(x);

Console.WriteLine("\n");

var selectbyy = from Rectangle in rectangles

orderby Rectangle.y

select Rectangle;

foreach (Rectangle y in selectbyy)

Console.WriteLine(y);

Console.WriteLine("\n");

var selectbysquare = from Rectangle in rectangles

orderby Rectangle.square

select Rectangle;

foreach (Rectangle square in selectbysquare)

Console.WriteLine(square);

Console.WriteLine("\n");

Console.WriteLine(rectangles.First());

Console.WriteLine(rectangles.Last());}}}

namespace pac1

{

public class WebNet

{

public LinkedList <User> users = new LinkedList<User>();

public int count = 0;

public int AddToList(User obj)

{

users.AddFirst(obj);

count++;

return count;

}

public int RemoveFromList(User obj)

{

users.Remove(obj);

count--;

return count;

}

}

public class User: IComparable

{

private string email;

private string password;

private string status;

public User(string Email, string Password, string Status)

{

email = Email;

password = Password;

status = Status;

}

public override bool Equals(object obj)

{

return base.Equals(obj);

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public override string ToString()

{

return base.ToString();

}

public int CompareTo(object o)

{

User user = o as User;

if (user != null)

return this.email.CompareTo(user.email);

else

throw new Exception("Object is not a User");

}

}

class Program

{

static void Main(string[] args)

{

User user1 = new User("Katya", "12374", "signin");

User user2 = new User("Nastya", "43221", "signout");

User user3 = new User("Vlad", "14567", "signin");

User user4 = new User("Nastya", "98765", "signout");

User user5 = new User("Anya", "24575", "signin");

Console.WriteLine(user1.CompareTo(user2));

Console.WriteLine(user2.CompareTo(user4));

Console.WriteLine(user4.CompareTo(user5));

LinkedList<User> github = new LinkedList<User>();

github.AddFirst(user2);

Console.WriteLine(github);

var UserOrder = from vect in github

select vect;

}

}

}