Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**Иркутский национальный исследовательский технический университет**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

|  |
| --- |
| **Отчет** |
| по лабораторной работе №4 по дисциплине «Технологии программирования»  «Основы наследования» |
| наименование темы |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИСТб-22-3 |  |  |  | Д. А. Корепанов |
|  |  | шифр |  | подпись |  | И.О. Фамилия |
| Проверил доцент |  |  |  |  |  | З. А. Бахвалова |
|  |  |  |  | подпись |  | И.О. Фамилия |
|  | | | |  | | |

Иркутск 2024 г.

# Постановка задачи

**Требования к работе:**

**Оценка "Удовлетворительно":**

1. Придумать иерархию классов
2. Должен быть 1 базовый класс, и 3 класса наследника. У базового класса должно быть, как минимум одно свойство передающиеся по наследству остальным, у каждого из классов наследников должно быть как минимум два уникальных свойства.
3. Реализовать эмулятор торгового автомата

**Оценка "Хорошо":**

1. В эмуляторе реализовать отображение состояния очереди (чтобы было видно какие объекты в каком порядке сейчас находятся в автомате)

**Оценка "Отлично":**

1. Добавить картинки для каждого типа объектов, которые будут видны при выводе очереди

**Вариант 4**

Для раздачи транспортных средств (количество колес)

* Велосипеды (тип (горный, городской, детский), радиус колес)
* Автомобили (тип (автобус, грузовик, внедорожник, легковая), объем двигателя, количество дверей),
* Самолеты (тип двигателя, максимальная высота полета)

# Таблица спецификаций метода btnRefill\_Click

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| rnd | Случайное число | var | [0;2] |

# Таблица спецификаций метода ShowInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| bikeCount | Счётчик для велосипедов | int | [0; 232] |
| carCount | Счётчик для автомобилей | int | [0; 232] |
| planeCount | Счётчик для самолётов | int | [0; 232] |
| vehicle | Объект в списке vehiclesList | var | - |

# Таблица спецификаций метода btnGet\_Click

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| vehicle | Объект в списке vehiclesList | var | - |

# Таблица спецификаций метода AppdateAutomate

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| str | Объект в очереди | string | - |
| vehicle | Объект в списке vehiclesList | var | - |

# Таблица спецификаций класса Vehicle

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| rnd | Случайное число | Random | - |
| number\_of\_wheels | Количество колёс | int | [2;8] |

# Таблица спецификаций класса Bike

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| radius | Радиус колёс | int | [0;232] |
| type | Тип транспортного средства | TypeOfBike | - |

# Таблица спецификаций класса Car

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| engine\_capacity | Объём двигателя | int | [0;300] |
| number\_of\_doors | Количество дверей | int | [0;6] |
| type | Тип транспортного средства | TypeOfCar | - |

# Таблица спецификаций класса Plane

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | ОДЗ |
| maximum\_flight\_altitude | Максимальная высота полёта | int | [0;12000] |
| type | Тип двигателя | TypeOfEnginePlane | - |

# Таблица тестов

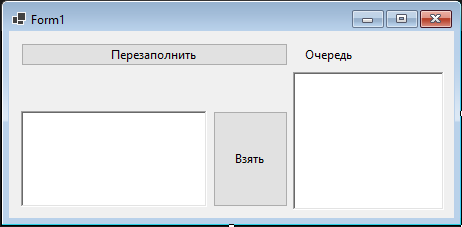
| Номер теста | Что проверяем | Выходные данные |
| --- | --- | --- |
| 1 | Работоспособность  (Список полон) |  |
| 2 | Работоспособность  (Список пуст) |  |

# Код программы

|  |
| --- |
| Form 1  namespace Laba\_4\_Inheritance  {  /\*Для раздачи транспортных средств (количество колес)  -Велосипеды (тип (горный, городской, детский), радиус колес)  -Автомобили (тип (автобус, грузовик, внедорожник, легковая), объем двигателя, количество дверей),  -Самолеты (тип двигателя, максимальная высота полета)\*/  public partial class Form1 : Form  {  List<Vehicle> vehiclesList = new List<Vehicle>();  public Form1()  {  InitializeComponent();  }  private void btnRefill\_Click(object sender, EventArgs e)  {  //vehiclesList.Clear();  //for (var i = 0; i < 10; ++i)  //{  // // классно да, список типа Fruit, а кладем Mandarin  // // вот она: "сила наследования"  // vehiclesList.Add(new Bike());  //}  vehiclesList.Clear();  var rnd = new Random();  for (var i = 0; i < 13; ++i)  {  switch (rnd.Next() % 3) // генерирую случайное число от 0 до 2 (ну остаток от деления на 3)  {  case 0: // если 0, то мандарин  vehiclesList.Add(Bike.Generate());  break;  case 1: // если 1 то виноград  vehiclesList.Add(Car.Generate());  break;  case 2: // если 2 то арбуз  vehiclesList.Add(Plane.Generate());  break;  // появление других чисел маловероятно  }  }  ShowInfo();  txtQueue.Text = AppdateAutomate();  }  private void ShowInfo()  {  // заведем счетчики под каждый тип  int bikeCount = 0;  int carCount = 0;  int planeCount = 0;  // пройдемся по всему списку  foreach (var vehicle in vehiclesList)  {  // помните, что в списки у нас лежат фрукты,  // то есть объекты типа Fruit  // поэтому чтобы проверить какой именно фрукт  // мы в данный момент обозреваем, мы используем ключевое слово is  if (vehicle is Bike) // читается почти как чистый инглиш, "если fruit есть Мандарин"  {  bikeCount += 1;  }  else if (vehicle is Car)  {  carCount += 1;  }  else if (vehicle is Plane)  {  planeCount += 1;  }  }  // а ну и вывести все это надо на форму  txtInfo.Text = "Велосипед\tАвтомобиль\tСамолёт"; // буквы экнмлю, чтобы влезло на форму  txtInfo.Text += "\n";  txtInfo.Text += String.Format("{0}\t\t{1}\t\t{2}", bikeCount, carCount, planeCount);  }  private void btnGet\_Click(object sender, EventArgs e)  {  if (vehiclesList.Count == 0)  {  txtOut.Text = "Пусто Q\_Q";  return;  }  var vehicle = vehiclesList[0];  vehiclesList.RemoveAt(0);  // ЗАМЕНИЛ НАШИ if`ы  txtOut.Text = vehicle.GetInfo();  // обновим информацию о количестве товара на форме  ShowInfo();  txtQueue.Text = AppdateAutomate();  }  private string AppdateAutomate()  {  string str = "";  foreach (var vehicle in vehiclesList)  {  switch (vehicle)  {  case Bike:  str += "Велосипед\n";  break;  case Car:  str += "Автомобиль\n";  break;  case Plane:  str += "Самолёт\n";  break;  }  }  return str;  }  }  } |

|  |
| --- |
| Vehicles  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  /\*Для раздачи транспортных средств (количество колес)  -Велосипеды (тип (горный, городской, детский), радиус колес)  -Автомобили (тип (автобус, грузовик, внедорожник, легковая), объем двигателя, количество дверей),  -Самолеты (тип двигателя, максимальная высота полета)\*/  namespace Laba\_4\_Inheritance  {  /\*  var bikes = new List<Bike>();  var cars = new List<Car>();  var planes = new List<Plane>();  \*/  public class Vehicle  {  public static Random rnd = new Random();  public int number\_of\_wheels = 0;  public virtual String GetInfo()  {  var str = String.Format("\nКоличество колёс: {0}", this.number\_of\_wheels);  str += String.Format("шт");  return str;  }  }  /\*Велосипеды\*/  public enum TypeOfBike {mountain, urban, children};  public class Bike : Vehicle  {  public TypeOfBike type = TypeOfBike.mountain;  public int radius = 0;  public override String GetInfo()  {  var str = "Вам достаётся велосипед!";  str += String.Format("\nТип: {0}", GetTypeString(this.type));  str += String.Format("\nРадиус: {0}", this.radius);  str += String.Format("″");  str += base.GetInfo();  return str;  }  private string GetTypeString(TypeOfBike type)  {  switch (type)  {  case TypeOfBike.mountain:  return "Горный";  case TypeOfBike.urban:  return "Городской";  case TypeOfBike.children:  return "Детский";  default:  return "Неизвестный тип";  }  }  public static Bike Generate()  {  //var rnd = new Random();  return new Bike  {  type = (TypeOfBike)rnd.Next(3),  radius = 5 + rnd.Next() % 25,  number\_of\_wheels = /\*2 + rnd.Next() % 9\*/(2 + rnd.Next() % 2 )  };  }  }  /\*Автомобили\*/  public enum TypeOfCar {bus, truck, offroad, passenger};  public class Car : Vehicle  {  public TypeOfCar type = TypeOfCar.bus;  public int engine\_capacity = 0;  public int number\_of\_doors = 0;  public override String GetInfo()  {  var str = "Вам достаётся автомобиль!";  str += String.Format("\nТип: {0}", GetTypeString(this.type));  str += String.Format("\nОбъем двигателя: {0}", this.engine\_capacity);  str += String.Format("л.с.");  str += String.Format("\nКоличество дверей: {0}", this.number\_of\_doors);  str += String.Format("шт");  str += base.GetInfo();  return str;  }  private string GetTypeString(TypeOfCar type)  {  switch (type)  {  case TypeOfCar.bus:  return "Автобус";  case TypeOfCar.truck:  return "Грузовик";  case TypeOfCar.offroad:  return "Внедорожник";  case TypeOfCar.passenger:  return "Легковой автомобиль";  default:  return "Неизвестный тип";  }  }  public static Car Generate()  {  //var rnd = new Random();  return new Car  {  type = (TypeOfCar)rnd.Next(4),  engine\_capacity = 64 + rnd.Next() % 301,  number\_of\_doors = 2 + rnd.Next() % 6,  number\_of\_wheels = /\*2 + rnd.Next() % 9\*/(1 + rnd.Next() % 4) \* 2  };  }  }  /\*Самолёты\*/  public enum TypeOfEnginePlane { piston\_engine, jet\_engine };  public class Plane : Vehicle  {  public TypeOfEnginePlane type = TypeOfEnginePlane.piston\_engine;  public int maximum\_flight\_altitude = 0;  public override String GetInfo()  {  var str = "Вам достаётся самолёт!";  str += String.Format("\nТип: {0}", GetTypeString(this.type));  str += String.Format("\nМаксимальная высота полета: {0}", this.maximum\_flight\_altitude);  str += String.Format("м");  str += base.GetInfo();  return str;  }  private string GetTypeString(TypeOfEnginePlane type)  {  switch (type)  {  case TypeOfEnginePlane.piston\_engine:  return "Поршневой двигатель (пропеллерный)";  case TypeOfEnginePlane.jet\_engine:  return "Реактивный двигатель";  default:  return "неизвестный тип";  }  }  public static Plane Generate()  {  //var rnd = new Random();  return new Plane  {  type = (TypeOfEnginePlane)rnd.Next(2),  maximum\_flight\_altitude =100 + rnd.Next() % 12001,  number\_of\_wheels = /\*2 + rnd.Next() % 9\*/(1 + rnd.Next() % 5) \* 2  };  }  }  } |

# Внешний вид формы

****

# Репозиторий с кодом

[DinoBombino/Laba\_4\_Inheritance (github.com)](https://github.com/DinoBombino/Laba_4_Inheritance)