2 Система методов исполнителя Робот

*(Фрагменты описания библиотеки классов «**библиотекаМетодовИсполнителяРобот**»)*

2.1 Введение

Всякому решению задачи с использованием исполнителя *Робот* (объектом класса ***инфоРобот***) должна предшествовать работа по созданию среды решаемой задачи[1].

Действия, связанные со вторым этапом решения задач – программной обработкой обстановки на поле исполнителя Робот, созданной каким-то из названных (см. [1]) средств, реализуются возможностями следующих классов библиотеки классов ***библиотекаМетодовИсполнителяРобот***:

инфоРобот.cs – центральный класс, методами которого реализуется решение задачи. Методы этого класса будут подробно описаны ниже.

инициаторСреды.cs – вспомогательный класс, участвующий в предварительной подготовке информации о поле *Робота*. Данная информация, используя конструктуры (их 4-е варианта), либо инициируется по ходу решения задачи (динамически), либо подчитывается из текстового файла ***Inlet.in***.

Клетка.cs – вспомогательный класс, методы которого предназначены для хранения информации о ситуации в конкретной клетке поля (наличие *Робота* в данной клетке, ее закрашенности, наличии стен на ее краях (верхнем, нижнем, левом и правом), наличии загрязненности (радиационной/температурной), наличии символьной пометки). Объекты данного класса, кроме того, призваны «выдавать» хранящуюся в нем (объекте) информацию о части хранящейся в нем информации описанных выше видов.

изоРобот.cs и *средаРобота.cs* – вспомогательные классы, предназначенные для визуализации результатов работы по конструированию среды, наравне, как и для визуализации результатов работы самого исполнителя Робот и запуска процесса визуализации

завершительРаботы.cs – вспомогательный класс, методы которого реализуют фиксацию работы как исполнителя РедакторСреды, так и исполнителя Робот в виде текстового файла определенной структуры. Имена создаваемых файлов соответственно ***Inlet.in*** и ***Pattern.out*** для *РедактораСреды* и ***Outlet.out*** для Робота. Кстати, это делает возможным автоматическое тестирование (а не только визульаный контроль) результатов работы с исполнителем *Робот*.

В данном документе мы познакомимся с совокупностью методов и свойств основного класса обработки созданных сред – ***инфоРобот***. Все множество средств разделим на отдельные группы, реализующих специфические действия.

2.2 Группы методов и свойств класса *инфоРобот*

2.2.1 Специальные методы

1. **public** инфоРобот(**ситуацияНаПоле** поле)

   Конструктор Робота, «подчитывающий среду» после ее создания объектом класса ***редакторСреды***

   где

[1] С основными вариантами такой работы можно познакомиться в ***«СК исполнителя РедакторСреды исполнителя Робот.pdf»***.

поле – переменная типа *ситуацияНаПоле* – структура, хранящая всю совокупность информации на поле Робота

2. **public** инфоРобот(**string** имяФайлаОбстановки)

   Конструктор Робота, вызывающий сформированную среду по «имениФайлаОбстановки», предопределенной условием задачи[2].

3. **public void** конечнаяТочкаРобота()

   Метод, используемый в обязательном порядке в случае, если по условию задачи Робота надо установить по окончании работы в строго определенную клетку. В этом случае, после перемещения Робота в указанную клетку поля, надо указать данную команду [3].

4. **public void** кончитьРаботу(**string** файлРезультата)

   Данным методом завершается решение всякой задачи с исполнителем Робот. Он автоматически формирует файл ответа, который и будет содержать «полный итог» решения задачи.

5. **public ситуацияНаПоле** результатРаботы

   Свойство, которым, аналогично предыдущему методу, следует завершать решение всякой задачи с исполнителем *Робот*, если есть необходимость визуализации результатов работы [4].

2.2.2 Методы движения и изменения среды

1. **public void** вверх()

   Метод перемещения Робота на одну клетку поля вверх, в случае отсутствия стены между клетками. В случае, если у данной клетки есть стена сверху, метод генерирует исключение с сообщением: «Не могу, сверху стена!»

2. **public void** вправо()

   Метод перемещения Робота на одну клетку поля вправо, в случае отсутствия стены между клетками. В случае, если у данной клетки есть стена справа, метод генерирует исключение с сообщением: «Не могу, справа стена!»

3. **public void** вниз()

   Метод перемещения Робота на одну клетку поля вниз, в случае отсутствия стены между клетками. В случае, если у данной клетки есть стена снизу, метод генерирует исключение с сообщением: «Не могу, снизу стена!»

---

[2]При автоматическом тестировании – ***Inlet.in***, а при ручном – удобнее пользоваться средствами визуализации, содержащими, как правило, «подсказывающую» информацию

[3]В противном случае решение задачи не будет считаться завершенным

[4]Последними операторами метода Main в этом случае должны быть команды:
поле = робот.результатРаботы;
Application.Run(new средаРобота(поле));

4. **public void** влево()

   Метод перемещения Робота на одну клетку поля влево, в случае отсутствия стены между клетками. В случае, если у данной клетки есть стена слева, метод генерирует исключение с сообщением: «Не могу, слева стена!»

5. **public void** закрасить()

   Метод закраски клетки поля текущего положения Робота, заметим, что допускается, но нежелательна, без особых на то указаний, многократная закраска клетки.

2.2.3 Свойства организации обратной связи

1. **public bool** сверхуСтена

   Свойство-определитель наличия стены сверху относительно текущего положения Робота

2. **public bool** справаСтена

   Свойство-определитель наличия стены справа относительно текущего положения Робота

3. **public bool** снизуСтена

   Свойство-определитель наличия стены снизу относительно текущего положения Робота

4. **public bool** слеваСтена

   Свойство-определитель наличия стены слева относительно текущего положения Робота

5. **public bool** сверхуСвободно

   Свойство-определитель возможности свободного (без генерации исключения) перемещения из данной клетки поля в верхнюю относительно текущего положения Робота

6. **public bool** справаСвободно

   Свойство-определитель возможности свободного (без генерации исключения) перемещения из данной клетки поля в правую относительно текущего положения Робота

7. **public bool** снизуСвободно

   Свойство-определитель возможности свободного (без генерации исключения) перемещения из данной клетки поля в нижнюю относительно текущего положения Робота

8. **public bool** слеваСвободно

   Свойство-определитель возможности свободного (без генерации исключения) перемещения из данной клетки поля в левую относительно текущего положения Робота

9. **public bool** клеткаЗакрашена

   Свойство-определитель наличия закраски клетки поля текущего положения Робота. При этом определяется сам факт закрашенности клетки, а не количество закрасок.

10. **public double** радиация

    Свойство-определитель величины радиационного загрязнения клетки поля текущего положения Робота

11. **public double** температура

    Свойство-определитель величины температурного загрязнения клетки поля текущего положения Робота

2.3 Особенности работы с исполнителем и визуализацией результатов работы

1. По входе в среду программирования *Visual Studio Начало работы* выбирается пункт меню «*Создать проект*».

2. На форме «*Создать проект*» в разделе «*Visual C# Windows*» выбирается (средняя часть формы) шаблон «*Приложение Windows Forms*», а в нижней части формы в окне «*Имя*» указывается имя проекта (*имя пространства имен программы*), а в окне «*Расположение*», используя возможности кнопки «*Обзор*», выбирается папка, в которой будет размещено решение задачи. После этого нажимается кнопка «*OK*». Через некоторое время (большее, чем при работе в консольном режиме) на экране разворачивается стандартное окно начальной установки, включая форму «*Forms1*».

3. В разделе формы «*Обозреватель решений*» нажатим правой клавиши мыши на пункте «*Forms1.cs*» добиваемся появления нового меню, в котором выбираем пункт «*Удалить*» и после запроса на подтвержение, удаляем форму «*Forms1*».

4. Дважды щелкаем левой клавишей мыши на пункте «*Program.cs*» того же раздела. В результата разворачивается окно вида:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace решениеЗадачи_00
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

5. В пункте меню «**Проект**» выбрать пункт «**Добавить ссылку**», а по открытии формы – подпункт «**Обзор**». Далее, найти файл «*библиотекаМетодовРоботаНаСтруктуреПоле.dll*»[5]

6. Раздел подключения библиотек классов дополнить строкой «**using** библиотекаМетодовРоботаНаСтруктуреПоле;» [6]

7. В блоке класса «**class** *Program*» вставить строки объявления объекта

   «**static** ***инфоРобот*** робот;» и

   «**static** ***ситуацияНаПоле*** поле;»

   В случае работы с готовым формирователем сред (см. ссылку 6) дополнительно указывается строка:

   «**static** ***построительСреды построитель*** = ***new*** построительСреды();»

8. Блок метода «**static void** Main(**string**[] args)» преобразовать следующим образом:

```
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
поле = построитель.построительСредыРобота("while", 14);
//Application.Run(new средаРобота(поле));

робот = new инфоРобот(поле);
решениеЗадачи14();
поле = робот.результатРаботы;
Application.Run(new средаРобота(поле));
```

где

– в строке *поле = построитель.построительСредыРобота("while 14);* у метода «построительСредыРобота» первый параметр – «while» – указывает класс решаемых задач, а второй – номер решаемой задачи;

– закоментированная строка *//Application.Run(new средаРобота(поле));*[7]

– последующие строки связаны с собственно решением задачи:

— строка *робот = new инфоРобот(поле);* «сообщает» исполнителю Робот информацию о исходной обстановке на поле;

— строка *решениеЗадачи14();* – передает поток управление методу, который описывает алгоритм решения задачи;

[5]Адрес файла следует узнать у преподавателя.

[6]В случае, если по ходу решения задачи предполагается использовать готовые средства библиотеки построения сред, пункты данного описания 5, 6 надо повторить, с целью подключения библиотеки «*библиотекаФормированияСред.dll*»

[7]Понятно, что для того, чтобы увидеть исходную обстановку решаемой задачи, комментарий надо убрать.

— строка *поле = робот.результатРаботы;* – сообщает переменной «поле» результат преобразования исходной обстановки предыдущим методом-решением задачи;

– строка *Application.Run(new средаРобота(поле));* – выполняет визуализацию выполненного решения задачи[8] .

[8]Комметрии на аналогичной строке визуализации исходной обстановки можно не ставить. В таком случае, после отображения исходной обстановки ее форму надо закрыть, а вместо нее появится новая форма с решением задачи.