Тема 15

Изпитна тема: Приложения с графичен потребителски интерфейс

1. Дефинира понятията: интегрирана среда за разработка (IDE), графичен интерфейс, събитие, обработка, източник. Различава основни контроли на графичния интерфейс и посочва техните свойства.

Средата за програмиране(Integrated Development Environment - **IDE**, интегрирана среда за разработка) е съвкупност от традиционни и нструменти за разработване на софтуерни приложения.

В средата за разработка пишем код, компилираме и изпълняваме програмите.

Средите за разработка интегрират в себе си **текстов редактор** за писане на кода, **език за програмиране, компилатор или интерпретатор** и **среда за изпълнение** за изпълнение на програмите, **дебъгер** за проследяване на програмата и търсене на грешки, **инструменти за дизайн на потребителски интерфейс** и други инструменти и добавки.

Графичен интерфейс е разновидност на потребителски интерфейс, в който елементите, предоставени на потребителя за управление, са изпълнени във вид на графични изображения (менюта, бутони, списъци и др.).

Събитие – действие, което протича в реално време и като отговор предизвиква едни или други действия, които се извършват като отговор на дадено събитие.

- събития свързани с клавиатурата
- събития свързани с мишката
- програмни събития

Събитията възникват при действие от страна на потребителя или системата.

Обработчик на събитие – част от програмния код, която съответства на конкретно събитие, наричаме обработчик на събитие (манипулатор на събитие). Като правило всички обекти могат да реагират на събития – едни на повече, други на по-малко.

Източник на събитие – (генератор) на едно събитие може да бъде:

- извършване на определено действие от страна на лицето, което използва (изпълнява) програмата – натискане на клавиш, придвижване или щракване на мишката и др.
- ОС – оповестяване, че текущият сеанс на работа с компютъра завършва и др.
- езиковия процесор – изтичане на зададен период от време.
- програмният код – изпълнение на специфични оператори или прилагане на определени методи към даден екземпляр на обект.

Контроли на графичния интерфейс - Контролите са обекти, които дават възможност за визуализиране на информацията на екрана и за взаимодействието с приложението чрез мишка, клавиатури др. Базов клас за всички контроли е класът System.Windows.Forms.Control

Общи свойства за всички компоненти:

Name – уникално име на компонентата

- клас Button – компоненти button1, button2

Visible – видимост на компонентата - True, False

Enabled – определя достъпността на компонентата -True, False

Size – размер в пиксели

- ширина – width; височина – height;

Location – координати X и Y в пиксели

Екранна форма от клас Form – свойства

StartPosition - място на формата върху екрана

Text – заглавие на прозореца от тип string

BackColor – цвят на вътрешността на формата

ForeColor – цвят за изписване на текстове във всички обекти

ControlBox – активира бутоните за затваряне, минимизиране и оразмеряване

Етикет – клас Label

Предназначение – поставяне на надписи

Основното предназначение на компонентите от класа Label е да се поставят надписи в основния прозорец на програмата и останалите контейнери. В етикета се поставят различни заглавия на прозорците и поясняващи надписи за предназначението на осталите компоненти. В етикета могат да се извеждат и стойности, които трябва само да се покажат в прозореца, но не бива да се променят от потребителя – например, резултати от извършените от програмата пресмятания. Текстът, който ще се изпише в етикета се задава в свойството Text. При поставяне на етикет в екранната форма Text = label1. Исканият от нас текст се въвежда в полето за редактиране на свойството. Друго важно свойство е Font – съставно свойство, в което се задават параметрите на използвания шрифт, с много подсвойства. Най-важните от тях са Name и Size, задаващи вида и размера на шрифта, както и определящите стила - Bold, Italic и Underline.

Свойства – Text, Font, Name, Size

Текстова кутия – клас TextBox

Компонентите на класа TextBox (текстова кутия) са предназначени за въвеждане на данни от клавиатурата по време на изпълнение на програмата. Това е и основното различие между текстовата кутия и етикета. Останалите свойства са почти идентични. Компонентата позволява да се въвеждат данни на един или на много редове, което се управлява от свойството Multiline. След като потребителят е въвел данните в текстовата кутия, те стават достъпни в програмата като съдържание на свойството Text. Като съдържание на това свойство по време на проектиране на формата или програмно може да се постави текст, който ще се покаже при

отваряне на прозореца и може да бъде съдържание на кутията по премълчаване или подсказка за потребителя какво да въведе в полето. Компонентите, в които потребителят може да въвежда данни, се наричат активни. Важно събитие за текстовата кутия е TextChanged (текстът е променен). Когато това събитие се случи, програмата би трябвало да го обработи, като съхрани въведеното в кутията в съответна променлива.

Предназначена за въвеждане на данни от клавиатурата

Свойства

- Multiline – въвеждане на повече от един ред
- Text – приема въведените стойности или визуализиран текст при зареждане на формата
- TextChanged – активна компонента

Бутон – клас Button

Компонентите на класа Button са предназначени за подаване на команди от страна на потребителя към изпълняваната програма. Основно свойство на командния бутон е надписът му, което е стойност на свойството Text и е добре да подсказва командата, която ще се стартира, когато потребителят щракне върху него с мишката. За тази компонента е характерно събитието Click, което се генерира при натискане на бутона с левия бутон на мишката. Средата автоматично генерира тяло на метода, с който това събитие да бъде обработено, когато щракнем с бутона. В тази функция програмистът изписва програмния код, който изпълнява свързаната с бутона команда.

Предназначение - за подаване на команди; командни бутони

Свойства-Text(задава текст върху бутона), Click(активира се при натискане)

Всички контроли от Windows Forms дефинират събития, които програмистът може да прихваща. Например контролата Button дефинира събитието Click, което се активира при натискане на бутона. Събитията в Windows Forms управляват взаимодействието между програмата и контролите и между самите контроли помежду им.

2. Дава пример за конструкции за контрол на изпълнението. Демонстрира употребата на конструкции за прихващане и обработка на изключения.

Конструкции за контрол на изпълнението - if / else , вложени if конструкции, **switch-case**

```
if (булев израз)
{
 тяло на условната конструкция;
}
else
{
 тяло на else-конструкция;
}
```

Тук тази конструкция работи по следния начин: в зависимост от резултата на израза в скобите (булевият израз) са възможни два пътя по, които да продължи потока от изчисленията. Ако булевият израз е true, се изпълнява тялото на условната конструкция, а else се пропуска като операторите в него не се изпълняват. В обратния случай се изпълнява else-конструкцията, а се пропуска основното тяло и операторите в него не се изпълняват.

```
switch (селектор)
{
 case стойност-1: код за изпълнение; break;
 case стойност-2: код за изпълнение; break;
 case стойност-3: код за изпълнение; break;
 case стойност-4: код за изпълнение; break;
 // ...
 default: код за изпълнение; break;
}
```

Конструкцията switch-case избира измежду части от програмен код на базата на изчислената стойност на определен израз. Този израз най-често е целочислен, но може да бъде и от тип string или char. Стойността на селектора трябва задължително да бъде изчислена преди да се сравнява със стойностите вътре в switch конструкцията. Етикетите (case) не трябва да имат една и съща стойност. При намиране на съвпадение на селектора с някоя от case стойностите, switch-case конструкцията изпълнява кода след съответния case. При липса на съвпадение, се изпълнява default конструкцията, когато такава съществува. Всеки case етикет, както и етикетът по подразбиране (default), трябва да завършват с ключовата дума break, която приключва работата на switch-case конструкцията, след като е намерено съвпадение и е изпълнен съответния код.

Пример:

```
x = int.Parse(textBox1Number1.Text);
y = int.Parse(textBox2Number2.Text);
switch (label1Sight.Text) {
case "+": rezultat = x + y; break;
case "-": rezultat = x - y; break;
case "*": rezultat = x * y; break;
case "/": rezultat = x / y; break;
default: { labelBug.Font = new Font(labelBug.Font, FontStyle.Bold);
MessageBox.Show(labelBug.Text); break; }
 } textBox3Sum.Text = rezultat.ToString();
```

В езика C# с оператора try … catch … finally може да се обработват изключения (exception), т.е. ситуации, които ако не бъдат обработени, ще предизвикат аварийно спиране на програмата.
Синтаксис:

```
try { < оператори, при изпълнение на които се очаква изключение > }
catch ( < вид на изключението >)
{ < оператори, които ще се изпълнят при възникне на изключение > }
```

finally { < оператори, които ще се изпълнят винаги > }
Когато програмата достигне този оператор, се опитва да изпълни операторите в блока try. Ако изключението се случи, се изпълнява кодът след catch (хващам, улавям), а след това кодът след finally (накрая). Ако очакваното изключение не се случи, се изпълнява само кодът след finally. Частите catch и finally не са задължителни, може да липсват.

Пример:

```
try {
    a = int.Parse(textBoxA.Text);
 }
catch (FormatException)
{
   a = 0;
   textBoxA.Text = "0";
}
```

3. Демонстрира работа с класове, обекти и свързаните с тях събития.

- Библиотеката Windows Forms дефинира:
  - съвкупност от базови класове за контролите и контейнер-контролите
  - множество графични контроли
- Основни базови класове:
  - Component – .NET компонент
  - Control – графична контрола (компонента с графичен образ)
  - ScrollableControl – контрола, която поддържа скролиране на съдържанието си
  - ContainerControl – контрола, която съдържа други контроли и управлява поведението на фокуса
- Класът System.Windows.Forms.Control е основа на всички графични Windows Forms контроли
- Неговите свойства са типични за всички Windows Forms контроли
- По-важните свойства на класа Control:
  - Anchor, Dock – задават по какъв начин контролата се "закотвя" за контейнера си
  - Bounds – задава размера и позицията на контролата в нейния контейнер
  - BackColor – задава цвета на фона
  - ContextMenu – задава контекстно меню (popup menu) за контролата
- По-важните събития на класа Control:
  - Click – настъпва при щракване с мишката върху контролата
  - Enter, Leave – настъпват при активиране и деактивиране на контролата
  - KeyDown, KeyUp – настъпват при натискане и отпускане на клавиш (или комбинация)
  - KeyPress – при натискане на нефункционален клавиш
  - MouseDown, MouseUp, MouseHover, MouseEnter, MouseLeave, MouseMove, MouseWheel – настъпват при събития от мишката, настъпили върху контролата

Поставяне на контроли:

Поставянето на контроли във формата става чрез Controls.Add:

```
Form form = new Form();
Button button = new Button();
button.Text = "Close";
form.Controls.Add(button);
```

Управление на събитията:

Прихващането на събития става по следния начин:

```
Form form = new Form();
Button button = new Button();
button.Click += new EventHandler(
   this.button_Click);
...
private void button_Click(
   object sender, EventArgs e)
{
   // Handle the "click" event
}
```

4. Обяснява моделите бази данни. Разработва модел бази данни, така че да реши поставената задача.

- Модели на базите от данни
  - йерархичен (дървовиден)
  - мрежови
  - релационен (табличен)
  - обектно-релационен
- Релационните бази от данни
  - Представляват съвкупности от таблици и връзки между тях (релации)
  - Ползват здрава математическа основа: релационната алгебра
- RDBMS(**Relational Database Management System**) системите се наричат още
  - сървъри за управление на бази от данни
  - или просто "Database сървъри"

ADO.NET - Набор от класове за работа с данни

- Набор от класове, интерфейси, структури и други типове за достъп до данни през изцяло .NET базирана реализация
- Програмен модел за работа с данни
- Осигурява възможност за работа в несвързана среда
- Осигурява връзка с XML
- Наследник на ADO (Windows технология за достъп до бази от данни)

Data Provider-и в ADO.NET