Ehindistudy.com

Visual Basic in Hindi – विजुअल बेसिक क्या है?

Visual Basic एक प्रोग्रामिंग लैंग्वेज है जिसे 1991 में माइक्रोसॉफ्ट ने विकसित किया था। विजुअल बेसिक का इस्तेमाल विंडोज ऑपरेटिंग सिस्टम की एप्लीकेशन को विकसित करने के लिए किया जाता है।

विजुअल बेसिक को windows ऑपरेटिंग सिस्टम के लिए विकसित किया गया है। पहले visual basic को BASIC के नाम से जाना जाता था।
Visual Basic को शार्ट फॉर्म में VB कहा जाता है। यह एक प्रोग्रामिंग भाषा है, जिसे माइक्रोसॉफ्ट के द्वारा उनके ऑपरेटिंग सिस्टम के लिए develop (विकसित) किया गया है। इस भाषा का उपयोग एप्लीकेशन और सॉफ्टवेयर को develop करने के लिए किया जाता है।

यह भाषा प्रोग्रामर को GUI (Graphical User Interface) करती है। इस भाषा को कंप्यूटर की third generation के दौरान विकसित किया गया था। कंप्यूटर की third generation के दौरान बहुत सी प्रोग्रामिंग भाषाओ को विकसित किया गया था, जिनमे COBOL, FORTRAN जैसी भाषाए शामिल थी।
लेकिन इन भाषाओ को सिखना और समझना किसी भी यूजर के लिए मुश्किल होता था, इसलिए वर्ष 1964 मे Thomas Kurtz और Denis Ritchi के द्वारा एक प्रोग्रामिंग भाषा को विकसित किया गया जिसका नाम BASIC (Beginners All Purpose Symbolic Instruction Code) रखा गया।

जिस समय माइक्रोसॉफ्ट ने विंडोज ऑपरेटिंग सिस्टम को लांच किया था, ठीक उसी समय Windows Programming के लिए BASIC भाषा के नए वर्शन को भी लांच कर दिया गया था , जिसका नाम Visual Basic रखा गया।

इस भाषा का syntax काफी सरल है जिसकी वजह से इस भाषा को सिखना और समझना काफी आसान है और इस भाषा में यूजर या developer अपनी इच्छा के अनुसार एप्लीकेशन और सॉफ्टवेयर को create कर सकता है।

Features of Visual Basic in Hindi – विजुअल बेसिक की विशेषताएं

इसकी विशेषताएं निम्नलिखित होती हैं:-

1:- यह एक ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग लैंग्वेज है जो OOPS के सभी Features को Support करती है।
2:- यह भाषा user interface डिज़ाइन करने में मदद करती है।

3:- इस भाषा में यूजर को application का निर्माण करने के लिया ज्यादा कोड नहीं लिखना पड़ता।

4:- यह भाषा visual studio का उपयोग करके developer को integrated development environment प्रदान करती है।

5:- विजुअल बेसिक एक object पर आधारित प्रोग्रामिंग लैंग्वेज है जिसमें यूजर tools का इस्तेमाल करके एप्लीकेशन को डिज़ाइन कर सकता है।

6:- इस लैंग्वेज में एक शक्तिशाली data access टूल होता है जिसके द्वारा यूजर front end database का निर्माण कर सकता है।

7:- इस भाषा में यूजर को auto complete का फीचर मिलता है जिसकी मदद से यूजर को पूरा कोड लिखने की ज़रूरत नहीं पड़ती , कोड अपने-आप complete (पूरा) हो जाता है।

8:- इसका प्रयोग Graphical User Interface के लिये भी किया जाता है।

9:- इस लैंग्वेज में एक ऐसा फीचर होता है जो Programs को अपने-आप Compile कर देता है।

10:- इसमें एक कोड एडिटर होता है जो खुद ही Syntax Error को check करके error (गलतियों) को दिखा देता है।

11:- यह भाषा developer और user की मदद ऑफलाइन और ऑनलाइन दोनों तरीको से करती है।

Ehindistudy.com

All Versions of Visual Basic – विजुअल बेसिक के सभी वर्शन

1- Visual Basic 1.0 (Window)

यह visual basic का पहला वर्शन है जिसे माइक्रोसॉफ्ट के द्वारा वर्ष 1991 में लॉन्च किया गया था। इस वर्शन में एप्लीकेशन को डिज़ाइन करने के लिए पहले एक यूजर इंटरफ़ेस बनाया जाता था उसके बाद कोड को add किया जाता था।

इस वर्शन का उपयोग ज्यादातर business application बनाने के लिए किया जाता था। इस वर्शन को जॉर्जिया के Atlanta मे (Comdex/Windows World Trade Show) के दौरान लांच किया गया था।

2– Visual Basic 1.0 (DOS)

इस वर्शन को माइक्रोसॉफ्ट के द्वारा वर्ष 1992 में लॉन्च किया गया था। माइक्रोसॉफ्ट के इस वर्शन का इस्तेमाल GUI (Graphical User Interface) के Appearance को Simulate करने के लिए किया जाता था।

यह वर्शन यूजर इंटरफ़ेस को कण्ट्रोल करने में मदद करता है। इस वर्शन में Button, Text Box, Checkbox, Combo Box, List जैसे कॉम्पोनेन्ट शामिल है।

3– Visual Basic 2.0 (windows)

इस वर्शन को नवंबर 1992 में लांच किया गया था। यह विजुअल बेसिकका तीसरा वर्शन था। यह वर्शन Visual Basic 1.0 (DOS) और Visual Basic 1.0 (window) की तुलना में काफी सरल और तेज था।

4– Visual Basic 3.0 (windows)

यह एक Standard और Professional Version था। इस वर्शन को माइक्रोसॉफ्ट ने वर्ष 1993 में लांच किया था। इस Version मे Microsoft ने Jet Database Engine को add किया था। Visual Basic 3.0 वर्शन exe files को Create करने में सक्षम नहीं था।

5– Visual Basic 4.0 (window)

इस वर्शन को वर्ष 1995 में लॉच किया गया था। इस वर्शन का उपयोग 32-bit Application और 16-bit के Windows Program को Develop करने के लिए किया जाता था। इसके अलावा Visual Basic 4.0 का उपयोग non-GUI Classes को लिखने के लिए किया जाता था।

6– Visual Basic 5.0 (Windows)

विजुअल बेसिकके इस वर्शन को windows के 32-bit Version के लिए launch किया गया था। Visual Basic 5.0 को वर्ष 1997 में लांच किया गया था।

7– Visual Basic 6.0 (windows)

इस वर्शन को माइक्रोसॉफ्ट ने वर्ष 1998 में लांच किया था। इस वर्शन में कुछ नए फीचर्स को add किया गया था, जिनकी मदद से यूजर इस वर्शन में Web Based Application को डिज़ाइन कर सकता था।

Ehindistudy.com

IMPLEMENTING NAVIGATIONAL DESIGN

Implementing Navigational Design

Introduction

एक Visual Basic Application की Functionality को Improve करने के लिए Menu Bars बहुत ही सरल तरीका प्रदान करते हैं। Menu Bars का प्रयोग करके हम अपने Program को ज्यादा बेहतर तरीके से Control कर सकते हैं। Menu Bar हमेंशा ज्यादातर Applications में Top पर होता है।

Visual Basic IDE हमें एक Built-In Menu Editor प्रदान करता है, जिसका प्रयोग करके हम काफी आसानी से अपने Application का Menu Create कर सकते हैं। एक बार Menu Bar को Create कर देने के बाद हम Menu के किसी भी Option को जिस तरह से चाहें उस तरह से Program करके Use कर सकते हैं।

इस Top Level Menu Bar के अलावा एक और प्रकार का Menu Window Based Applications में Create किया जाता है, जिसे Popup Menu कहते हैं। ये Menu किसी Particular Control के लिए Define किए जाते हैं, इसलिए ये Control Specific Menu भी कहलाते हैं। इस प्रकार के Menus को अक्सर Context-Sensitive Menu भी कहा जाता है। User को Customized Menu Options प्रदान करने के लिए हम Menus को Program के Runtime में भी Create कर सकते हैं।

Understanding Menu Basics

Applications Menu Bars का प्रयोग Commands के एक Organized Collection Represent करने के लिए Define किए जाते हैं। ये Menu Bars User को सारांश के रूप में Application की Capabilities की Information देते हैं। यदि किसी Program में उचित रूप से Organize किया गया Menu Bar हो, तो User काफी आसानी से Application व उसके विभिन्न Commands को समझ सकता है। User इन Menu Bar के Options के साथ Assign की गई Shortcut Keys को भी आसानी से समझ सकता है और उपयोग में ले सकता है।

चूंकि एक Menu Bar का Application के User के लिए काफी महत्व होता है, इसलिए एक Programmer को काफी सावधानी व सतर्कता के साथ Menu को Design करना होता है, ताकि वह User को Application के सम्बंध में ज्यादा से ज्यादा जानकारी प्रदान कर सके।

Programmers Visual Basic 6 में दो तरीकों से Menus को Create कर सकते हैं। पहले तरीके में एक Programmer Visual Basic 6 के IDE के साथ आने वाले Built-In Menu Editor का प्रयोग करके Menus Develop कर सकता है। इस तरीके को Use करके काफी तेजी से Menu को Design व Maintain किया जा सकता है। एक बार Menu को Design कर लेने के बाद Menu के हर Item की Properties को Programmatically Set किया जा सकता है।

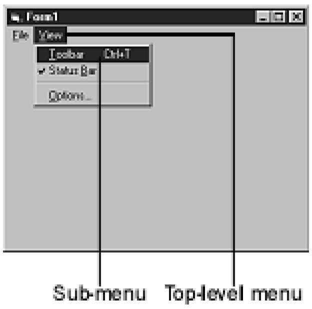
दूसरे तरीके से Menu को Create करने के लिए Programmer को Windows के Win-32 API Functions का प्रयोग करना होता है। Win-32 API Functions Low Level Functions होते हैं, जिन्हें Operating System प्रदान करता हैं और Visual Basic द्वारा इन्हें Use किया जाता है।

Win32 API Functions के समूह के Specialized Functions को Use करके एक Programmer, Menu Items Create कर सकता है, Items की Property को Set कर सकता है और Menu के Structure को Change कर सकता है। API में जो Functionality होती हैं,

उनमें से ज्यादातर Functionality Menu Editor में भी Available होती हैं। हालांकि ये API हमें Menu की क्षमताओं को बढाने की सुविधा प्रदान करती हैं, लेकिन इन्हें Use करना काफी जटिल प्रक्रिया होती है।

Menu Creation With Menu-Editor

Menu Editor का प्रयोग करके Menu Create करने के लिए हमें सबसे पहले Visual Basic IDE में Standard EXE Project को Select करना होता है। वैसे हम किसी भी प्रकार के Project में Menu को Add कर सकते हैं, लेकिन हम हमारे ज्यादातर उदाहरणों में Standard EXE Project को ही Use करेंगे, क्योंकि ये काफी Flexible होता है और हम इसे विभिन्न प्रकार के अन्य Project के रूप में Define करके Use कर सकते हैं।

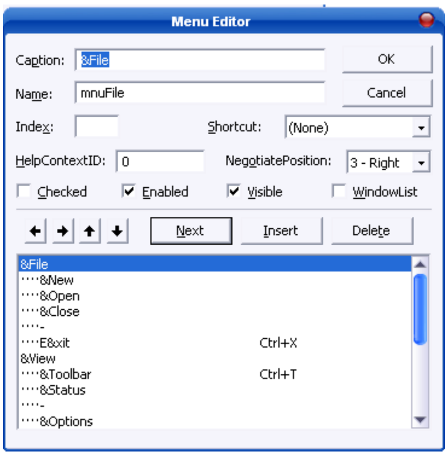


इस चित्र में हम देख सकते हैं कि Form के सबसे Top पर दिखाई देने वाले Menu को Top-Level Menu कहा जाता है, जबकि इस Menu से Related Menu को Sub-Menu कहा जाता है। हम हमारी जरूरत के आधार पर जितने चाहें, उतने Sub-Level तक Sub-Menus Create कर सकते हैं।

Sub-Menu तब Appear होता है, जब हम Top-Level Menu को Open करते हैं। Sub-Menu विभिन्न प्रकार के Commands को Implement करता है, जो कि Logically Top-Level Menu से सम्बंधित होते हैं। चित्र में दिए गए उदाहरण में हमने File Menu Create किया है, जो कि ज्यादातर Applications में होता है। ये एक Top-Level Menu है और इस Menu में सामान्यतया File से Related Options जैसे कि New, Open, Close, Save, Save As आदि होते हैं। ये Options उन Commands को Execute करते हैं, जो कि Currently Opened File को प्रभावित करते हैं।

Using The Menu Editor

Standard Windows Menus हमेंशा Title Bar से Just नीचे Located होता है। Menu-Editor का प्रयोग करते हुए Menu Create करने के लिए हमें Form पर Right Click करना होता है। जब हम Form पर Right Click करते हैं, तब एक Popup Menu Display होता है। इस Popup Menu से हमें "**Menu Editor…**" Option को Select करना होता है। जब हम इस Option को Select करते हैं, तब हमें अगले चित्र के अनुसार Menu Editor Dialog Box दिखाई देता है। Visual Basic में Menu-Editor हमेंशा Design Time में ही उपलब्ध होता है, जहां हम Menu को अपनी जरूरत के आधार पर Design कर सकते हैं।



Menu Item Create करते समय सबसे पहले हमें Menu के Caption को Set करना होता है। Menu Caption वह नाम होता है, जो हमें Menu Bar पर दिखाई देता है। menu Caption में हम जो नाम लिखते हैं, उसके जिस Character को हम Access Key बनाना चाहते हैं, उस Character से पहले Prefix के रूप में '&' Ampersand Character को Use करना होता है। Access Key वह Character होता है, जो Keyboard के Alt Key को Press करने से Highlight हो जाता है।

Menu Bar के किसी Item का Caption Set करने के बाद हमें उस Menu Item का वह नाम Specify करना लिए होता है, जिसे Use करके हम उस Menu Item Object को Program के Code में Use करने में सक्षम हो पाते हैं। Visual Basic में किसी भी Object का नाम Specify करते समय हम सामान्यतया **Hungarian Notation** का प्रयोग करते हैं। इस Notation का प्रयोग करके हम ये तय करते हैं कि हम किस तरह के Object का नाम Define कर रहे हैं।

चूंकि हम यहां पर Menu Item का नाम Define कर रहे हैं, इसलिए नाम Specify करते समय हमें Name Text Box में Menu Item के नाम के साथ "**mnu**" Prefix का प्रयोग करना होता है, जो कि Menu Item के लिए Hungarian Notation है। सामान्यतया Hungarian Notation में किसी भी Object के नाम के Short Form के तीन Characters का प्रयोग किया जाता है, लेकिन हम हमारी सुविधानुसार इससे कम या ज्यादा Characters का प्रयोग कर सकते हैं।

यदि हम किसी Menu Item को कोई Shortcut Key Assign करना चाहें, तो ये काम हम Shortcut नाम के Combo Box द्वारा कर सकते हैं। इस Combo Box से हम जिस Shortcut Key को Select करते हैं, Currently Define किए जा रहे Menu Item के साथ वह Shortcut Key Set हो जाती है। Menu Editor पर निम्नानुसार एक Button Set होता है:

Next Button

इस Button पर Click करके हम अगले Menu Item पर Move हो सकते हैं।

Insert Button

इस Button का प्रयोग करके हम किसी नए Menu Item को पहले से Exist दो Menu Items के बीच में नया Menu Item Insert कर सकते हैं।

Delete Button

इस Button पर Click करके हम किसी Exist Menu Item को Menu से Delete कर सकते हैं।

इन Buttons के अलावा Menu Editor पर चार Arrow Buttons होते हैं। Up Arrow व Down Arrow Button का प्रयोग करके हम Menu में दिखाई देने वाले Menu Item की Position को Up या Down Direction में Shift कर सकते हैं। जबकि Right Arrow Button का प्रयोग करके हम किसी Menu Item को Main Menu का Sub Menu बनाने के लिए करते हैं और Left Arrow Button को Use करके हम किसी Sub Menu Item को Main Menu Item के रूप में Convert कर सकते हैं।

इनके अलावा Menu Editor पर Checked, Enabled, Visible व WindowList नाम के चार Check Box भी होते हैं, जिनका प्रयोग करके हम किसी Menu Item की Check, Enable व Visible Property को Set कर सकते हैं। WindowList नाम का Check Box MDI Form पर Window पर Effect दिखाता है।

Menu Bar में Separator का प्रयोग करने के लिए हमें किसी भी Menu Item के Caption में एक Single Dash का प्रयोग किया जाता है। जब हम किसी Menu Item के Caption में Dash का प्रयोग करते हैं, तो Program के Run Time में ये एक Separator की तरह दिखाई देता है।

Attaching Code With Menu Items

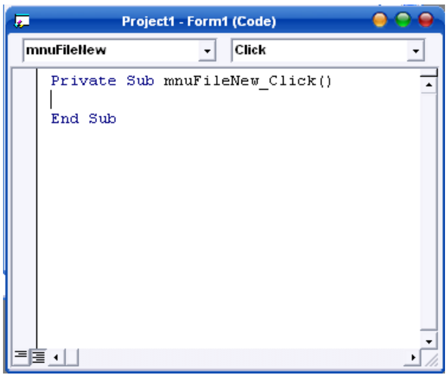
Visual Basic एक Event Driven Programming Language है। हम जब भी Computer के साथ किसी Action को Use करते हैं, तो कोई ना कोई Event Generate होता है। उदाहरण के लिए जब हम Mouse से Click करते हैं, तब Click Event Generate होता है।

इसी तरह से जब हम Keyboard से किसी Key को Press करते हैं, तब Keypress Event Generate होता है। Visual Basic में Develop किए जाने वाले Application में हम विभिन्न प्रकार के कामों को पूरा करने के लिए इन्हीं Events को उपयोग में लेते हैं।

उदाहरण के लिए जब हम Menu के किसी Item पर Click करते हैं, तब Click Event Generate होता है। इस Event के Response में हमें जो भी काम करने होते हैं, उन कामों को हम एक Procedure में लिख लेते हैं। ये Procedure **Event Procedure** कहलाता है।

Visual Basic में जितने भी Controls होते हैं, उन Controls के नाम व हमारे जिस Action के Response में हमें कोई काम करवाना होता है, उस Event के नाम दोनों नामों को एक Underscore के Symbol द्वारा आपस में Connect करके एक Procedure Create किया जाता है। ये Procedure ही Event Procedure होता है।

जैसे यदि हम File Menu के New Option पर Click करते हैं, तो इस Generate होने वाले Click Event को Handle करने के लिए हमें जो भी Code लिखने होते हैं, वे Codes हमें mnuFileNew_Click() नाम के Procedure में लिखने होते हैं। ये Event Procedure File Menu के New Option पर Click करने पर Execute होता है, जिससे इसमें लिखे गए Codes Execute हो जाते हैं।



Visual Basic में हमें किसी भी Event Procedure को Create करने की जरूरत नहीं होती है। हम जब भी किसी Menu Item को Click करते हैं अथवा जब भी किसी Control को Double Click करते हैं, तो उस Control व उससे Associated Default Event के Combination के Event Procedure का Header Automatically Create हो जाता है और हम Visual Basic के Code Editor Window में पहुंच जाते हैं, जहां हम Visual Basic Application के विभिन्न प्रकार के Runtime के Executable Codes विभिन्न प्रकार के Event Procedures में लिखते हैं।

Visual Basic के Code Window में लिखे जाने वाले विभिन्न प्रकार के Codes किसी ना किसी Event Procedure में ही लिखे जाते हैं। इस Code Window में दो Combo Box होते हैं। पहले Combo Box में Form पर स्थित विभिन्न प्रकार के Controls के नाम होते हैं, तथा दूसरे Combo Box में उस Control के साथ Use किए जा सकने वाले विभिन्न प्रकार के Events की जानकारी होती है। हम जिस Control के साथ जिस Event को Use करना चाहते हैं, उस Control व उससे Associated Event को Select करते ही Visual Basic स्वयं ही एक नया Event Procedure Create कर देता है, जिसे हम Code Window में देख सकते हैं।

Visual Basic में Menu Items के साथ केवल एक Click Event ही Associate हो सकता है, इसलिए विभिन्न Menu Items केवल Click Event को ही Response करते हैं और हमें जिस

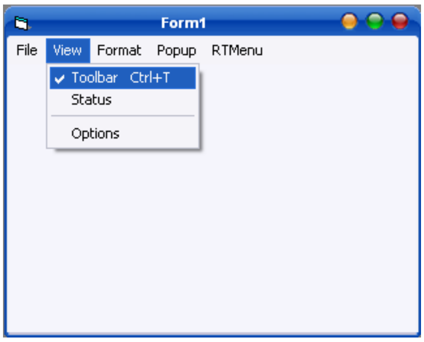
Menu Item पर Click करने पर जो काम करवाना होता है, उस काम को हम उस Menu Item से Associated Event Procedure में ही Define करते हैं।

हमें जिस Menu Item के लिए Click Event को Handle करने वाला Event Handler Define करना होता है, उस Menu Item पर Single Click करते ही हम उस Menu Item के Click Event Handler में पहुंच जाते हैं, जहां हम उस Menu Item से सम्बंधित वे सारे काम Specify कर सकते हैं, जिन्हें हम Menu Item के Click होने पर पूरा करवाना चाहते हैं।

Dynamic Modification Of Menu Items

Visual Basic एक Object Based Programming Language है। इसमें विभिन्न कामों को करने के लिए हमें विभिन्न प्रकार के Visible अथवा Non-Visible Objects को Use करना होता है, जिन्हें हम Controls भी कहते हैं। जिस तरह दुनिया की हर वस्तु की कुछ ना कुछ विशेषता होती है, उसी तरह से Visual Basic के हर Control अथवा Object की भी अपनी कुछ विशेषताएं होती हैं, जिन्हें Object या Control की **Property** कहा जाता है।

ये Properties Object के Appearance से सम्बंधित होती हैं। Visual Basic के विभिन्न प्रकार के Controls की विभिन्न प्रकार की Properties होती हैं। हर Object की कुछ ऐसी Properties होती हैं, जिन्हें केवल Program के Design Time में ही Control किया जा सकता है, तो कुछ Properties ऐसी होती हैं, जिन्हें केवल Program के Run Time में ही Control किया जा सकता है। कुछ Properties ऐसी भी होती हैं, जो Program के Design Time व Run Time दोनों स्थानों पर Changeable होती हैं।



जब हम किसी Menu Item को Program के Design Time में Create करते हैं, तब हम Menu Item की कुछ Property Settings जैसे कि Checked, Enabled, Visible आदि को Design Time में ही Define कर लेते हैं।

एक बार इन Properties को Set कर दिए जाने के बाद हम इन Properties को Program के Run Time में भी Control कर सकते हैं। जब हम किसी Control की किसी Property को Program के Run Time में Control करते हैं, तब इस प्रक्रिया को Dynamic Modification कहा जाता है।

यदि हम हमारे पिछले बनाए गए Menu System के View Menu के Check Box को Control करना चाहें, तो हम ये काम निम्नानुसार Code द्वारा कर सकते हैं। इस Code में हम View Menu के Toolbar Option के आगे Check Box को Display कर रहे हैं।

```
Private Sub mnuViewToolbar_Click()
   mnuViewToolbar.Checked = True
End Sub
```

जब हम View Menu के Toolbar Option पर Click करते हैं, तब उपरोक्तानुसार mnuViewToolbar_Click() नाम का Event Procedure Create होता है। इस Event Procedure में हमने mnuViewToolbar Object की Checked Property को True कर दिया है। ऐसा करने के कारण जब हम इस Option पर Click करते हैं, तब ये Procedure Execute हो

जाता है, और Form के View Menu के Toolbar नाम के Option के साथ हमें एक Check दिखाई देने लगता है।

यदि इस एक Line के Code में हम थोडा सा परिवर्तन करके Code को निम्नानुसार Change कर दें, और फिर इस Menu Item पर Click करें, तो यदि Toolbar के साथ Check लगा होता है, तो वह Check हट जाता है और यदि इस Option के साथ Check लगा हुआ नहीं होता है, तो Check लग जाता है। ये Code निम्नानुसार है:

```
Private Sub mnuViewToolbar_Click()
  mnuViewToolbar.Checked = True = Not mnuViewToolbar.Checked = True
End Sub
```

चूंकि Menu Item की Checked Property Default रूप से True होती है, इसलिए उपरोक्त Define किए अनुसार Code को हम निम्नानुसार भी लिख कर सकते हैं:

```
Private Sub mnuViewToolbar_Click()
  mnuViewToolbar.Checked = Not mnuViewToolbar.Checked
End Sub
```

जब ये Code Execute होता है, तब यदि mnuViewToolbar Object की Checked Property में True मान होता है, तो Not Operator उस मान का Opposite मान यानी False मान mnuViewToolbar के Checked Property में Assign कर देता है, जिससे Toolbar Menu Item के साथ दिखाई देने वाला Check Symbol Hide हो जाता है।

लेकिन यदि Checked Property में False मान होता है, तो Not Operator की वजह से इस Property में True Store हो जाता है और Toolbar Option के साथ हमें एक Check Symbol दिखाई देने लगता है। इसी तरह से हम कुछ अन्य Menu Items की Properties को Set करने के लिए भी निम्नानुसार Codes लिख सकते हैं:

```
Private Sub Form_Load()
  mnuFileOpen.Enabled = False
  mnuViewStatus.Checked = True
  mnuFormatBold.Checked = True
  mnuPopup.Visible = False
End Sub
```

इन Code Lines द्वारा हम समझ सकते हैं कि हम Menu Design करते समय जिस तरह के Format का प्रयोग करके Menu के विभिन्न Items का नाम Specify करते हैं, उन नामों से Program को Code करते समय हमें काफी सुविधा हो जाती है।

उदाहरण के लिए mnuFormatBold नाम का Menu Item Format Menu का एक Sub Menu Item है। इसी तरह से mnuFileOpen Menu Item Open Menu का एक Sub Menu Item है।

Adding Pop-Up Menu With Application

Popup Menu और एक Top Level Menu, दोनों प्रकार के Menu Create करने में किसी प्रकार का कोई अन्तर नहीं होता है। अन्तर केवल इतना होता है, कि Top Level Menu Items Click Event से Effect में आते हैं, जबकि Popup Menu Mouse Up Event से Effect में आते हैं।

Popup Menu को भी सामान्य Menu की तरह ही Menu Editor में Create किया जाता है, लेकिन इस Popup होने वाले Main Menu की Visible Property को False कर दिया जाता है, जिससे Popup का Main Menu Display नहीं होता है और चूंकि Main Popup Menu ही Display नहीं होता है, इसलिए इसके Sub Menu Items भी Application Window के Top में Display नहीं होते हैं।

लेकिन इस Main Menu के सभी Sub Menu Items Normal होते हैं, इसलिए जब हम Parent Form के PopupMenu Method Call करते हैं और उसमें Argument के रूप में किसी Popup Menu का नाम Pass करते हैं, तो Argument के रूप में Pass किए गए Popup Menu के सभी Sub Menu Items Form पर उस स्थान पर दिखाई देने लगते हैं, जहां हमने Mouse के Right Button को Click किया है।

हम जिस Form पर Menu को Menu Editor द्वारा Design करते हैं, उसी Form पर हम किसी Popup Menu को Show करवा सकते हैं। इसलिए यदि हमने Form1 नाम के Form पर कोई Popup Menu Design किया हो, तो हमें निम्नानुसार Statement लिखकर Popup Menu को Display करवाना होगा:

```
Form1.PopupMenu mnuPopup
```

ये Statement Form1 नाम के Form पर बनाए गए mnuPopup नाम के Popup Menu को Show करने का काम करता है। सामान्यतया Popup Menu Application के किसी Specific Control अथवा किसी Specific Area से Related होता है। हम विभिन्न प्रकार के Popup Menus को Create करके उन्हें विभिन्न प्रकार के Application Area अथवा विभिन्न प्रकार के Controls के साथ Associate कर सकते हैं। Popup Menus को सामान्यतया Mouse के Right Button को Click करने पर Display करवाया जाता है।

MouseUp Event and MouseUp Event Procedure

चूंकि सामान्यतया Mouse के Right Button को Click करने पर ही Popup Menu Display होता है, इसलिए Popup Menu को Display करवाने के लिए हमें Mouse के MouseUp Event को Use करना होता है, क्योंकि सामान्यतया जब हम Mouse के Right Button को Press करके Release करते हैं, तो Button के Release होते समय यानी Mouse के Right Button के Up होते समय ही Popup Menu Display होता है। मानलो कि हम Form पर ही Mouse के Right Click से Activate होने वाले Popup Menu को Display करवाना चाहते हैं, तो हमारा MouseUp Event Procedure Form के साथ Associate हो जाएगा और ये Event Handler कुछ निम्नानुसार दिखाई देगा:

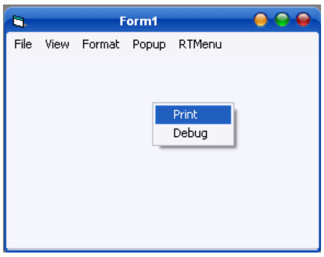
```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

End Sub
```

इस Procedure में यदि हम उपरोक्त Code Type कर दें, तो हमें निम्नानुसार Procedure प्राप्त होगा:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Form1.PopupMenu mnuPopup
End Sub
```

ये Procedure Create करने के बाद जब हम Application को Run करते हैं और Form पर Left Click करते हैं, तब हमें एक Popup Menu दिखाई देता है। लेकिन वास्तव में हमें ये Popup Menu Mouse के Right Button के Press करने पर ही दिखाई देना चाहिए। इस स्थिति में हमें Mouse के Right Button को Identify करना होगा।



MouseUp Event Procedure – The Button Argument

जब हम MouseUp Event के साथ किसी जरूरत को पूरा करना चाहते हैं, तब MouseUp Event Procedure में हमेंशा चार Arguments दिखाई देते हैं। इन चारों Arguments में एक Button नाम का Argument होता है। इस Argument में उन Mouse Button की Information होती है, जिससे Event से Associated Control पर Click किया गया है।

1 यदि हम Mouse के Left Button से Click करते हैं, तो इस Argument में vbLeftButton नाम का Visual Basic Constant मान होता है, जो कि Visual Basic का Default Mouse Button है।

2 यदि हम Mouse के Right Left Button से Click करते हैं, तो इस Argument में vbRightButton नाम का Visual Basic Constant मान होता है। और

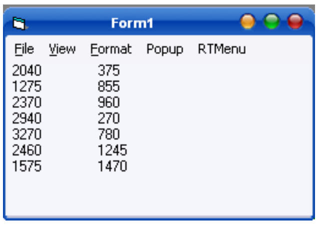
3 यदि हम Mouse के Middle Button से Click करते हैं, तो इस Argument में vbMiddleButton नाम का Visual Basic Constant मान होता है।

हम MouseUp Event Procedure के इस Button Argument में Stored मान को Check करके ये पता लगा सकते हैं कि Mouse के किस Button से Click किया गया है। ये काम करने के लिए हमें If Control Statement का प्रयोग करना होता है। If Control Statement का प्रयोग यदि हम पिछले Procedure में करें, तो हम पिछले Procedure को निम्नानुसार Modify कर सकते हैं:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

  If Button = vbRightButton Then
    Form1.PopupMenu mnuPopup
  End If

End Sub
```



इस MouseUp Procedure में हम Mouse द्वारा Associated Control पर किस Location पर Click कर रहे हैं, उस Location की X व Y Position की जानकारी MouseUp Event Procedure के X व Y नाम के Argument में होती है। यदि हम चाहें, तो Mouse के Click होने की Position की जानकारी को भी Print Method का प्रयोग करके Form पर Display कर सकते हैं, जिसके लिए हमें पिछले Procedure को निम्नानुसार Modify करना होगा:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

  If Button = vbRightButton Then
    Form1.PopupMenu mnuPopup
  End If
  Print(X, Y)
End Sub
```

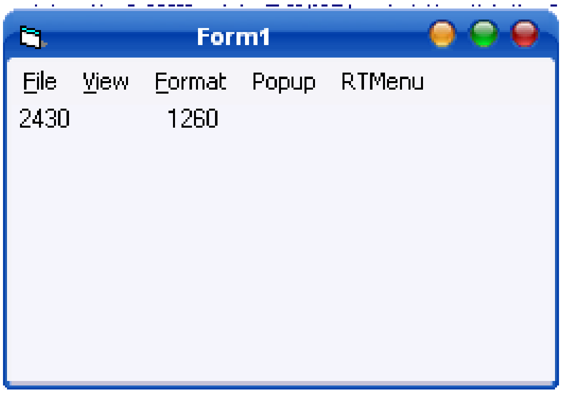
MouseMove Event and MouseMove Event Procedure

जब हम Mouse को Move करते हैं, तब MouseMove Event Trigger होता है और इस Event को Handle करने के लिए हम जिस Procedure को Create करते हैं, उसे MouseMove Event Procedure कहते हैं।

जब हम Mouse के MouseMove Event को Use करते हैं, तब भी Create होने वाले Procedure में ये ही चारों Arguments होते हैं। इसलिए यदि हम चाहें तो Mouse Pointer की Runtime Current Position को भी Form पर Display करवा सकते हैं। ये काम करने के लिए हमें Form के साथ MouseMove Event को Select करना होता है, जिससे निम्नानुसार Event Procedure Create होता है:

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

End Sub
```



अब Mouse के Pointer की Current Position को Display करने के लिए हम Form के Print Method को Use कर सकते हैं, लेकिन इस Method को Use करने पर हमें Form पर Mouse Position का पिछला मान भी दिखाई देता रहता है, इसलिए Mouse के पिछली Position के मान को Form पर से Clear करने के लिए हम Form के Cls Method को Use कर सकते हैं।

इस तरह से हम इस Procedure को निम्नानुसार Define कर सकते हैं, जो Mouse Pointer की Current Position को Form पर Display करता है:

```
Private Sub Form_MouseMove()
    (Button As Integer, Shift As Integer, X As Single, Y As Single)
  Form1.Cls()
  Print(X, Y)
End Sub
```

MouseUp Event Procedure – The Shift Argument

MouseUp Event Procedure में Shift नाम का एक Argument और होता है। MouseUp Event Generate होते समय यदि हम Keyboard के Ctrl, Alt अथवा Shift अथवा इनके Combination को Press करके रखते हैं, तो Shift Argument में उस Special Key अथवा Key Combination की जानकारी होती है। इसका पता भी हम If Statement का प्रयोग करके लगा सकते हैं कि **MouseUp** Event के समय Keyboard के किस **Shift Key** को Press किया गया था।

1. यदि MouseUp Event के समय Shift Button को Press किया गया हो, तो Event Procedure के Shift नाम के Argument में Visual Basic का vbShiftMask Constant मान Stored होता है। हम इस मान के स्थान पर Integer मान 1 भी Use कर सकते हैं।

2. यदि MouseUp Event के समय Shift Button को Press किया गया हो, तो Event Procedure के Shift नाम के Argument में Visual Basic का vbCtrlMask Constant मान Stored होता है। हम इस मान के स्थान पर Integer मान 2 भी Use कर सकते हैं।

3. यदि MouseUp Event के समय Alt Button को Press किया गया हो, तो Event Procedure के Shift नाम के Argument में Visual Basic का vbAltMask Constant मान Stored होता है। हम इस मान के स्थान पर Integer मान 4 भी Use कर सकते हैं।

जब MouseUp Event के समय किसी Shift Button को Press किया गया होता है, तो वह Button True Return करता है। इस स्थिति में हम If Statement द्वारा ये Check करवा सकते हैं कि किसी Shift Button को Press किया गया है अथवा नहीं। इस प्रक्रिया को हम निम्न MouseUp Procedure द्वारा समझ सकते हैं:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  If Shift = vbCtrlMask Then
    MsgBox "Control Key was pressed"
  End If

  If Shift = vbAltMask Then
    MsgBox "Alt Key was pressed"
  End If

  If Shift = vbShiftMask Then
    MsgBox "Shift Key was pressed"
  End If

End Sub
```

इस Procedure को Create करने के बाद Form को Run करते समय हम जिस किसी भी Shift Key को Press करके रखते हैं, उस Shift Key से सम्बंधित Message Box Display हो जाता है। यदि हम एक से ज्यादा Shift Key Combination को Identify करना चाहें, तो हमें इस Trick के आधार पर काम करना होता है कि जब भी हम MouseUp Event के समय किसी Shift Key को Press करके रखते हैं, तो वह Key कोई ना कोई Non-Zero मान Return करता है, जो कि True को Represent करता है।

```
Private Sub Form_MouseUp _
  (Button As Integer, Shift As Integer, X As Single, Y As Single)

  If Shift = 0 Then
    MsgBox "No Shift Key was pressed. Key Value : 0"
  End If

  If Shift = vbCtrlMask Then
    MsgBox "Control Key was pressed. Key Value : " & vbCtrlMask
  End If

  If Shift = vbAltMask Then
   MsgBox "Alt Key was pressed. Key Value : " & vbAltMask "
  End If

  If Shift = vbShiftMask Then
   MsgBox "Shift Key was pressed. Key Value : " & vbShiftMask "
  End If
'-----------------------------------------------------------------
--
Dim blnIsAlt As Boolean
Dim blnIsCtrl As Boolean
Dim blnIsShift As Boolean

  blnIsAlt = Shift And vbAltMask
  blnIsCtrl = Shift And vbCtrlMask
  blnIsShift = Shift And vbShiftMask

  If blnIsCtrl And blnIsShift And blnIsAlt Then
   MsgBox "Ctrl and Shift Button is Pressed"
  Exit Sub
  End If

  If blnIsAlt And blnIsShift Then
   MsgBox "Alt and Shift Button is Pressed"
  End If

  If blnIsAlt And blnIsCtrl Then
      MsgBox "Alt and Ctrl Button is Pressed"
  End If

  If blnIsCtrl And blnIsShift Then
   MsgBox "Ctrl and Shift Button is Pressed"
  End If

 End Sub
```

जब हम कोई Shift Key Press नहीं करते हैं, तब Shift Argument में Zero होता है, जो कि False को Represent करता है, लेकिन जब हम किसी Shift Key को Press करते हुए Click करते हैं, तब MouseUp Event के Shift Argument में उस Shift Key का कोई Integer Number होता है।

Visual Basic में 0 के अलावा सभी मान True को Represent करते हैं, इसलिए जब हमें दो Shift Keys को एक साथ Check करना होता है, तब एक Boolean Type के Variable में निम्नानुसार मानों को Check करके True या False के रूप में Keys की Information को Hold किया जाता है:

```
blnIsAlt = Shift And vbAltMask
blnIsCtrl = Shift And vbCtrlMask
blnIsShift = Shift And vbShiftMask
```

पहले Statement में हमने Shift व vbAltMask Constant के बीच AND Masking किया है। AND Masking में दोनों ही Identifiers के मान यदि True हों, तो Resultant मान भी True होता है, जिसका मतलब ये है कि Mouse के Button को Press करते समय Alt Button को Press करके रखा गया था।

लेकिन यदि Mouse का Button Press करते समय Keyboard के Alt Button को Press करके नहीं रखा गया हो, तो Shift Argument में vbAltMask के मान के स्थान पर False यानी Zero होगा अथवा किसी अन्य Shift Button का Integer मान होगा, जिससे AND Masking के दौरान Shift Argument व vbAltMask दोनों के मानों में Difference होने के कारण False Return होगा और blnIsAlt नाम के Boolean Type के Variable में False Store हो जाएगा।

इसी तरह से blnIsCtrl व blnIsShift नाम के Boolean Type के Variable में भी उसी स्थिति में True Store होता है, जब Keyboard से क्रमशः Control Key अथवा Shift Key को Press करके रखते हुए Mouse से Click किया गया हो।

जब हम एक से ज्यादा Shift Keys को Press करके रखते हुए Mouse के Button को Press करते हैं, तब एक से ज्यादा Boolean Variables में AND Masking के दौरान True Store हो जाता है और If Statement में हम फिर से इन Boolean Variables की Values की AND Masking करके ये पता लगा लेते हैं, कि एक से ज्यादा Shift Keys Press की गई थी अथवा नहीं, क्योंकि यदि Mouse Clicking के समय एक से ज्यादा Shift Keys को Press किया गया हो, तो एक से अधिक Boolean Variables में True होगा और जिन दो Boolean Variables में True Stored होगा, उन्हीं से सम्बंधित If Statement True Return करेगा और Execute होगा।

Popup Method

Form Object के Popup Method में भी कुछ Arguments होते हैं, जिनका प्रयोग करके हम Display होने वाले Popup Menu को Control कर सकते हैं। एक Popup Method के विभिन्न Optional Arguments का Description निम्नानुसार है:

PopupMenu(***Menu As Object***, [*Flags*], [*X*], [*Y*], [*DefaultMenu*])

Menu As Object

इस Argument के स्थान पर हमें उस Popup Menu का नाम देना होता है, जिसे हम किसी Specified Control के लिए Display करना चाहते हैं। इस Argument के अलावा सभी अन्य Arguments Optional होते हैं।

हम हमारी जरूरत के अनुसार अन्य Arguments को छोड सकते हैं, लेकिन इस Argument को Specify करना जरूरी होता है। चूंकि हमने हमारे उदाहरण में अपने Popup Menu का नाम mnuPopup दिया है, इसलिए हमें इस स्थान पर यही नाम Specify करना होता ळें

Flags

इस Argument के स्थान पर हमें Popup Menu के Popup होते समय की Location का Flag Set करना होता है, जो निम्न में से कोई भी हो सकता है:

vbPopupMenuLeftAlign (Constant)

ये Default मान होता है। ये Constant Set करने पर Popup होने वाला Menu Mouse Pointer के Click होने की X Location के Left Side में Popup होता है।

vbPopupMenuRightAlign (Constant)

ये Constant Set करने पर Popup होने वाला Menu Mouse Pointer के Click होने की X Location के Right Side में Popup होता है।

vbPopupMenuCenterAlign (Constant)

ये Constant Set करने पर Popup होने वाले Popup Menu के बीच में Mouse Pointer का वह X Location होता है, जहां Mouse Click किया गया होता है।

vbPopupMenuLeftButton (Behavior)

ये Behavior Set करने पर Popup Menu उस समय Trigger होता है, जब User Mouse के केवल Left Button से Popup होने वाले Menu के किसी Menu Item को Click करता है।

vbPopupMenuRightButton (Behavior)

ये Behavior Set करने पर Popup Menu उस समय Trigger होता है, जब User Mouse के किसी भी Button से Popup होने वाले Menu के किसी भी Menu Item को Click करता है।

जब हम Constant व Behavior दोनों को एक ही Argument में Set करना चाहते हैं, तब हमें दोनों मानों के बीच OR Operator का प्रयोग करना होता है। जैसे निम्न Statement Flags Argument में Popup Menu को Center में Align करेगा और Popup Menu Right व Left दोनों Button से Activate होगा:

```
Form1.PopupMenu mnuPopup, vbPopupMenuCenterAlign Or _
vbPopupMenuRightButton , , mnuPopupPrint
```

X, Y

इन Argument के स्थान पर हम जो भी मान Specify कर देते हैं, हमारा Popup Menu हमेंशा उसी Location पर Display होता है, फिर चाहे हम Control पर किसी भी स्थान पर Click क्यों ना करें।

चूंकि सामान्यतया हम Menu को उसी स्थान पर Display करवाना चाहते हैं, जहां हमने Right Click किया होता है, इसलिए इन Arguments को Fill नहीं किया जाता है। जब हम इन Arguments को Fill नहीं करते हैं, तब इनमें Mouse के Pointer की Current Position का Coordinate Automatically Set हो जाता है।

DefaultMenu

हम Popup Menu के जिस Item को Default रूप से Selected रखना चाहते हैं, उस Menu Item का नाम हमें इस Argument में Specify करना होता है।

जब हम किसी Method को Call करते हैं और उसमें बहुत सारे Arguments को Pass किए जाने की व्यवस्था होती है, तब हम विभिन्न प्रकार के Arguments को क्रम से Specify कर सकते हैं। लेकिन यदि हमें कुछ Arguments को छोड कर कुछ ही Arguments को Specify करना हो, तो हमें एक विशेष तरीके को Use करना जरूरी होता है।

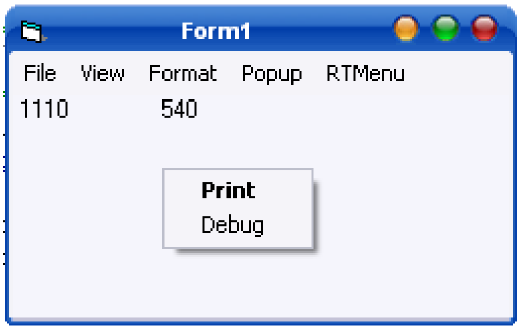
मानलो कि हम PopupMenu में केवल प्रथम व अन्तिम Argument को ही Set करना चाहते हैं, इस स्थिति में इस PopupMenu Method को Use करते समय हमें इसमें Arguments को निम्नानुसार Pass करना होगा।

```
Form1.PopupMenu mnuPopup, , , , mnuPopupPrint
```

किसी भी Method में हम प्रथम Argument को सामान्य तरीके से Specify कर सकते हैं, लेकिन यदि हम पिछले तरीके को Use करना ना चाहें, तो हम एक और तरीका Use करके किसी Procedure के विभिन्न Optional Arguments के मानों को बिना क्रम का ध्यान रखते हुए Pass कर सकते हैं, लेकिन इस तरीके में हमें Argument के नाम का पता होना जरूरी होता है।

उदाहरण के लिए यदि हम पिछले Statement को ही इस तरीके से Specify करें, तो हमें निम्नानुसार Argument Passing Code लिखना होगाः

```
Form1.PopupMenu mnuPopup, DefaultMenu := mnuPopupPrint
```



First Argument को Pass करने के लिए हमें किसी तरीके की जरूरत नहीं है, क्योंकि सामान्यतया सबसे पहले उन्हीं Arguments में मान Pass करने होते हैं, जिन्हें Pass करना जरूरी होता है और ये Compulsory Arguments एक Series में होने की वजह से इनके नामों को Specify करने की जरूरत नहीं होती है ना ही हम Compulsory Arguments के नामों को Specify कर सकते हैं, क्योंकि Compulsory Arguments के नामों में Visual Basic में Space का प्रयोग किया गया होता है और हम किसी Space युक्त Variable को सामान्य तरीके से Specify नहीं कर सकते हैं।

लेकिन Optional Arguments एक Single String Name होते हैं, इसलिए इन्हें Directly Access किया जा सकता है। इसी वजह से हमने PopupMenu Method के अन्तिम Option Argument के नाम DefaultMenu में **:=** Assignment Operator का प्रयोग करके Directly

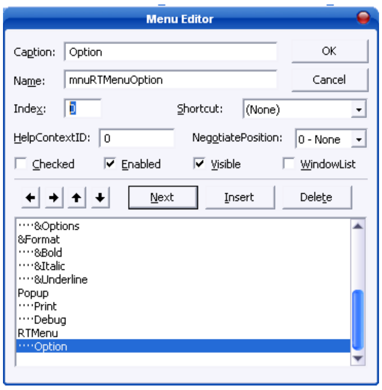
Default Menu Item का नाम Specify कर दिया है। Default Item को चित्र में Bold Face के रूप में देखा जा सकता है।

Adding Deleting Menu Items At Runtime

Runtime Menus Application के Runtime में उस समय Create होते हैं, जब उनकी जरूरत होती है। Runtime Menu Items Create करने के लिए हमें Menu का Control Array Create करना पडता है। Application के Runtime में Load Statement नया Menu Item Create करता है और Unload Statement किसी Menu Item को Delete करता है।

Creating Runtime Menu Items

Runtime Menu Item Create करने के लिए हमें तीन Steps को Follow करना पडता है। ये Steps निम्नानुसार हैं:



1 सबसे पहले हमें Menu Editor में उस Menu को Create करना होता है, जिसमें नए Sub Items को Application के Run Time में Add करना होता है। हमें जिस Sub Menu में नए Run Time Items को Add करना होता है, उसमें एक Sub Menu Item Create किया जाता है और उसे Control Array बना दिया जाता है। Control Array बनाने के लिए उस Sub Menu Item की Index Property में मान 0 लिख दिया जाता है, जिससे बनने वाला Item एक Control Array बन जाता है। Control Array एक ऐसाArray होता है, जो कि Visual Basic के किसी Control का Array होता है।

2 दूसरे Step में हमें Load Statement का प्रयोग करके Menu के Items को Application के Run Time में Load करना होता है। इस काम को हम निम्नानुसार Syntax द्वारा कर सकते हैं:

```
Load mnuRTMenuOption(1)
```

Menu Item का नाम Design Time में दिए गए उसी नाम पर ही आधारित होता है, जिसकी Index Property को हमने Set किया होता है। नया Menu Item Create करने के लिए हमें केवल उस नाम व एक Unique Index Number को Use करना होता है, जिसे Control Array में पहले Use ना किया गया हो।

3 जब Menu Item Load हो जाता है, उसके बाद हम उस Loaded नए Item को Access करने के लिए Menu के नाम व Index Value का प्रयोग कर सकते हैं। Menu Item के Load हो जाने के बाद हम Menu Item के साथ विभिन्न प्रकार की अन्य Run Time Properties को Set कर सकते हैं।

4 जब Menu Item को Program के Run Time में Create कर लिया जाता है, तब वह Menu Item Control Array का एक हिस्सा बन जाता है। जब हमें Runtime में Generate किए गए

Menu के लिए Code लिखने होते हैं, तब Design Time में Create किए गए Menu के First Index Number को आधार के रूप में Use करना होता है।

जो Menu Item हमने Design Time में Create किया था, उसमें Click Event नाम का एक Extra Argument होता है। **Index** Argument किसी Control को Identify करने के लिए Use किया जाता है।

Dynamic Menu Items के लिए हम निम्नानुसार Code लिख सकते हैं, जिसमें Code लिखने के लिए हमें Design Time में Create किए गए Menu Item के Click Event को Choose करना होता है। जब हम ऐसा करते हैं, तब हमें Code Window में निम्नानुसार एक Procedure Header प्राप्त हो जाता है:

```
Private Sub mnuRTMenuOption_Click(Index As Integer)

End Sub
```

इस Procedure में Index नाम का एक Argument होता है। इस Argument में उस Menu Item के Index का Number होता है, जिसे Click किया जाता है। चूंकि हमारे Menu Items Runtime में Create हो रहे हैं, फिर भी Runtime में भी Create होने वाले हर Menu Item का एक Index Number होता है, इसलिए यदि हम Runtime में Create होने वाले किसी Menu Item पर भी Click करते हैं, तब भी उस Runtime में Created Menu Item का Index Number इस Index Argument में आता है।

इसलिए हम Runtime में Create होने वाले Menu Item को भी इसी Procedure द्वारा Handle कर सकते हैं। इस Procedure में हम निम्नानुसार Code लिख सकते हैं, जो केवल एक Message Display करने का काम करता है:

```
Private Sub mnuRTMenuOption_Click(Index As Integer)

  Select Case Index
    Case 0
      MsgBox "You have clicked the first menu item."
    Case 1
      MsgBox "You have clicked the first Runtime menu item."
    Case 2
      MsgBox "You have clicked the second Runtime menu item."
  End Select
End Sub
```

Program के Run Time में दो नए Menu Items Create करने के लिए हम Form_Load Procedure में निम्नानुसार दो नए Menu Items को Load कर सकते हैं:

```
Private Sub Form_Load()

  Load mnuRTMenuOption(1)
  Load mnuRTMenuOption(2)

  mnuRTMenuOption(0).Caption = "First Menu Item Of Design Time"
  mnuRTMenuOption(1).Caption = "First Menu Item Or Runtime"
```

```
    mnuRTMenuOption(2).Caption = "Second Menu Item Or Runtime"
End Sub
```

केवल उन्हीं Menu Item को Runtime में Remove किया जा सकता है, जिन्हें Run Time में Create किया गया होता है, Design Time में Create किए गए Menu Items को Run Time में Remove नहीं किया जा सकता है। Runtime Menu Items को Remove करने के लिए हम दो तरीकों का प्रयोग कर सकते हैं। पहले तरीके में Menu Item की Visible Property को False करके Item को Invisible कर दिया जाता है। इस तरह का Item Invisible होने के बावजूद भी Memory में रहता है और उसकी Properties को Access किया जा सकता है। इस पहले तरीके को हम निम्नानुसार Use कर सकते हैं:

```
mnuRTMenuOption(2).Visible = False
```

लेकिन यदि हम किसी Menu Item को Permanently Memory से Remove करना चाहते हैं, तो हमें उस Menu Item को Unload करना होता है। Menu Item को Unload करने के लिए हम निम्नानुसार Syntax का प्रयोग कर सकते हैं:

```
Unload mnuRTMenuOption(2)
```

Unload Statement में केवल Runtime Control Names व Elements को ही Pass किया जा सकता है। यदि हम किसी Design Time Item को Unload Statement में Pass करते हैं, तो हमारे Application में Error Generate होता है।

ADDING CONTROLS TO FORM

Adding Controls To Form

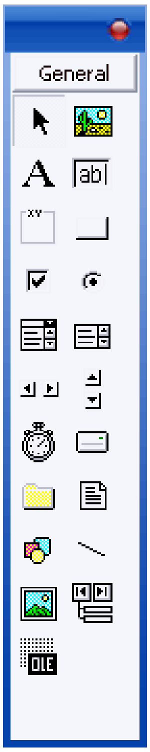
Introduction

Visual Basic में विभिन्न प्रकार के Built-In Controls हैं, जिन्हें हम Form अथवा किसी भी अन्य Container पर Visually Place कर सकते हैं। हम जिन Controls को Form पर Place करना चाहते हैं, Visual Basic IDE में दिखाई देने वाले Toolbox में उस Control को Click करते हैं और Mouse के Pointer को Form पर Drag N Drop करके एक Rectangle बनाते हैं। ऐसा करने पर नया Control Form पर Draw हो जाता है।

दूसरे तरीके में हमें जिस Control को Form पर Place करना होता है, उसे Toolbox पर Double Click करते हैं। Toolbox पर हम जिस Control को Double Click करते हैं, वह Control अपनी Default Properties के साथ Form में Center में Place हो जाता है, जिसे हम हमारी जरूरत के अनुसार सही Properties से Set कर सकते हैं।

हमें Toolbox पर स्थित सभी Tools का नाम ध्यान रखने की भी जरूरत नहीं होती है। हम जिस किसी भी Tool पर Mouse के Pointer को Place करते हैं, उस Control का नाम एक **ToolTip** Message के रूप में हमें दिखाई देने लगता है।

Container Controls



Visual Basic में कुछ Controls ऐसे होते हैं, जो स्वयं Active रूप से कोई काम नहीं करते हैं, बल्कि ये Controls किसी अन्य Control के लिए एक Container की तरह काम करते हैं। ये Controls Container Controls कहलाते हैं। Visual Basic में Form सबसे मुख्य Container होता है, जिस पर अन्य सभी प्रकार के सामान्य Controls अथवा Container Controls को Place किया जा सकता है। इसके अलावा Frame Control व PictureBox Control भी Container की तरह काम करते हैं।

PictureBox Control व Image Control दोनों के ज्यादातर Features एक समान होते हैं, लेकिन दोनों Controls में एक मुख्य अन्तर ये होता है कि PictureBox Control Container की तरह काम कर सकता है, जबकि Image Control Container की तरह काम नहीं करता है। हम किसी भी Container में विभिन्न प्रकार के Controls को Add करने के लिए विभिन्न प्रकार के तरीकों का प्रयोग कर सकते हैं।

जब हम Form के अलावा किसी अन्य Container में किसी Control को Place करना चाहते हैं, तब हमें हमेंशा उस Control को Container में Draw करना पडता है। यदि हम Control को Double Click करके Container में लाना चाहें, तो Controls हमेंशा Form Container पर ही Place होते हैं।

यदि हमें पहले से बने हुए Controls को किसी Container में Place करना हो, तो Place किए जाने वाले सभी Controls को Select करके Cut करना होता है और फिर जिस Container में इन Controls को Place करना है, उस Container को Select करके उसमें Paste करना होता है।

Setting Properties

विभिन्न प्रकार के Controls को Draw करने के बाद हमें उन Controls की Properties को अपनी जरूरत के अनुसार Set करना होता है। सामान्यतया सबसे ज्यादा काम में आने वाले Controls Visual Basic IDE के Toolbox पर Place किया गया है। लेकिन फिर भी CommandButton, TextBox व Labels Toolbox पर स्थित सभी Controls में से भी सबसे ज्यादा उपयोग में आने वाले Controls होते हैं, इसलिए सबसे पहले हम इन्हीं Controls की Properties के बारे में बात करेंगे।

हम किसी भी Control की Property को Design Time अथवा Run Time में Set कर सकते हैं। जब हम किसी Controls को Select करते हैं, तो उस Controls की विभिन्न Properties एक Properties Window नाम के Window में दिखाई देने लगती है, जिसे हम हमारी जरूरत के अनुसार Set कर सकते हैं। यदि Properties Window दिखाई ना दे रहा हो, तो Function Key F4 को Press करके इस Window को Display किया जा सकता है।

विभिन्न प्रकार के Controls की विभिन्न प्रकार की Properties होती है। हर Property को किसी ना किसी प्रकार की Numerical अथवा String Type की Value से Set किया जा सकता है।

जो Properties Boolean Type की Value से Set होती हैं, उन्हें Set करने के लिए Mouse से उन Properties पर Double Click करके उन्हें Toggle किया जा सकता है। उदाहरण के लिए CommandButton Control की Default व Cancel Properties Boolean Properties होती हैं, जिन्हें Double Click करके Toggle किया जा सकता है।

कुछ Properties ऐसी होती हैं, जिनमें कुछ Named Enumerated मानों में से किसी मान को Set किया जा सकता है। इस प्रकार की Properties एक Drop-Down List के रूप में दिखाई देती हैं। उदाहरण के लिए MousePointer Property एक Drop-Down List प्रदान करता है, जिसमें से वांछित Enumerated Named मान को Select किया जा सकता है।

कुछ Properties में Dialog Boxes के रूप में Sub Components होते हैं, जो कुछ Special Type के Properties के एक Group को Represent करते हैं। इस प्रकार की Property पर Double Click करने पर एक नया Property Page Display होता है, जिसके विभिन्न मानों को अपनी जरूरत के अनुसार Set किया जा सकता है।

जब हम किसी Control की Property को Program के Run Time में Set करना चाहते हैं, तब हमें Control के नाम के साथ Dot Operator का प्रयोग करना होता है और Dot के बाद Property को Specify करके Equal To Symbol को Use करना होता है और नए मान को इस Equal To Symbol के बाद Specify करना होता है। जैसे यदि हम CommandButton cmdClose के Caption को Change करना चाहें, तो ये काम हम निम्नानुसार कर सकते हैं:

```
cmdClose.Caption = “E&xit”
```

Visual Basic में जितने भी Objects हैं उन सभी Objects की Properties, Methods, Visual Basic में Use किए जाने वाले विभिन्न Constants आदि की जानकारी हम Object Browser से प्राप्त कर सकते हैं। Object Browser को Visual Basic के IDE में Display करने के लिए हमें Function Key F2 को Press करना होता है।

जब हम किसी Complex Object की Property को Access करना चाहते हैं, तब हमें Dots का प्रयोग एक से ज्यादा बार करना पडता है। उदाहरण के लिए यदि हम किसी TextBox में दिखाई दे रहे Text के Font की Style को Run Time में Change करना चाहें, तो ये काम हमें निम्नानुसार करना होगा:

```
txtBox.Font.Bold = True
txtBox.Font.Name = "Times New Romans"
```

Visual Basic में हर Object की कोई ना कोई एक Default Property होती है, जिसे बिना Specify किए हुए ही Access अथवा Set किया जा सकता है।

1 **CommandButton** की Default Property **Value** Property होती है।
2 **Label** की Default Property **Caption** Property होती है और
3 **TextBox** की Default Property **Text** Property होती है।

किसी भी Control की Default Property को Access करने के लिए हम केवल Control के नाम को Use कर सकते हैं। हमें Control के नाम के साथ Property को Specify करने की जरूरत नहीं होती है। जैसे निम्न Statements को देखिए:

```
cmdOK = True
txtName = "Manmohan"
lblName = "Krishna"
```

हम देख सकते हैं कि इन तीनों ही Controls के साथ हमने किसी Property को Specify नहीं किया है। इस स्थिति में ये तीनों ही Controls अपनी Default Property को Access करते हैं।

Common Properties

Visual Basic में कुछ ऐसी Properties होती हैं, जिन्हें ज्यादातर Objects द्वारा Common रूप से Common कामों के लिए Share किया जाता है। इनमें से कुछ Most Common व सबसे ज्यादा Use की जाने वाली Properties निम्नानुसार हैं:

Name Property

हर Object का अपना एक नाम होता है, जिसे Name Property द्वारा Assign किया जाता है। ये नाम किसी Object को उस समय Refer करता है, जिस समय हम Code लिखते समय उस Control की Properties व Methods को Access करना चाहते हैं।

उदाहरण के लिए यदि हम Command Button को cmdExit नाम देते हैं, तो इस Command Button को Refer करने के लिए हमें निम्नानुसार Code द्वारा Refer कर सकते हैं:

```
cmdExit.Caption = "Close"
```

Name Property Design Time में ही Set की जा सकती है। Run Time में ये Property Read-Only हो जाती है, इसलिए हम Program के Run Time में इस Property को Change नहीं कर सकते हैं।

Enabled Property

ये एक True/False Property होती है। जब ये Property False होती है, तब इससे Associated Control Grayed हो जाता है और किसी भी Event पर Response नहीं करता है, यहां तक कि Grayed Control पर Focus भी Move नहीं होता है।

चूंकि Label Control कभी भी Focus प्राप्त नहीं करता है, इसलिए इसकी Enabled Property का Application Program पर कोई प्रभाव नहीं पडता है।

हालांकि यदि Label Control की Enabled Property True हो, तो वह उस समय Click व DblClick जैसे Events को Response करता है, जब Mouse का Pointer Label पर होता है। इस Property को Design Time व Run Time दोनों समयों पर Set किया जा सकता है।

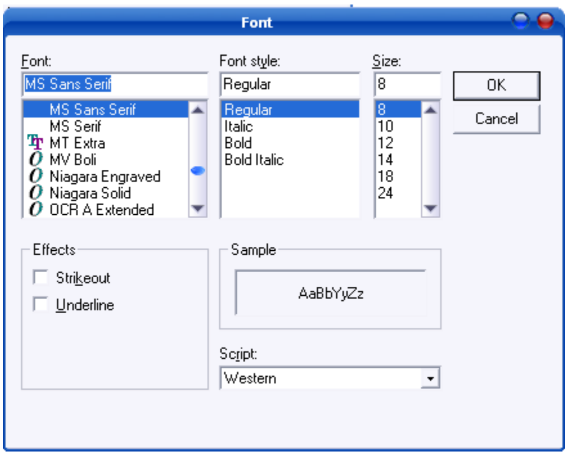
Timer Control की Enabled Property को यदि False कर दिया जाए, तो Timer Control अन्य Controls की तुलना में अलग तरीके से React करता है। जब Timer Control Enabled होता है, तब Specified Time के Duration में ये अपने Procedure के Codes को Execute करता रहता है, लेकिन जब इसे Disable कर दिया जाता है, तब Specified Time Duration में इसका Code Procedure Execute नहीं होता है। सामान्यतया यदि Timer Control के Interval Property में मान 0 Specify कर दिया जाए, तो ये Control Disable Control की तरह काम करने लगता है।

Visible Property

ये Property सभी Controls के लिए By Default True होता है। इस Property को False करने पर Associated Control Form पर दिखाई नहीं देता है। यदि Design Time में इस Property को False कर दिया जाए, तो भी Programmer को Design Time में सम्बंधित Control दिखाई देता है, लेकिन Run Time में Control दिखाई नहीं देता। हम इस Property को Design Time व Run Time दोनों स्थितियों में Access कर सकते हैं।

Font Property

Font Property वास्तव में एक Object होता है, जिसमें इसकी स्वयं की Properties होती हैं। हम Font Object की Properties को Design Time में एक Dialog Box द्वारा Manipulate कर सकते हैं। इस Object को Call करने के दो तरीके होते हैं।



पहले तरीके में हम Control की Font Property को Double Click करते हैं और दूसरे तरीके में हम Font Property के साथ में Associated Ellipsis Button (…) को Click करते हैं। दोनों ही स्थितियों में हमें एक Font Dialog Box दिखाई देता है।

यदि हम किसी Control के लिए इस Font Object की Property को Runtime में Access करना चाहें, तो हमें Double Dotted Syntax का प्रयोग करना पडता है।

उदाहरण के लिए यदि हम चाहते हैं कि Label का नाम Bold Face में दिखाई दे, तो इस काम को हम निम्न Code द्वारा कर सकते हैं:

```
Dim blnOrigBold As Boolean
blnOrigBold = lblName.Font.Bold
lblName.Font.Bold = True
```

यदि हम एक ही Object के लिए उसकी कई Properties को एक साथ Set करना चाहें, तो हम With … End With Block का प्रयोग कर सकते हैं। उदाहरण के लिए यदि हम Label के लिए Font की तीन Properties को एक साथ Set करना चाहें, तो ये काम हम निम्नानुसार कर सकते हैं:

```
With lblLable.Font
   .Bold = True
   .Underline = True
   .Italic = True
   .Name = "Times New Romans"
End With
```

Size And Position Properties

Object की Height व Width Properties Object की Size को Specify करते हैं, जबकि Top व Left Properties Object की Container पर Position को Specify करते हैं।

Size व Position को जिस Unit में Measure किया जाता है, उस Unit को हम Container की ScaleMode Property द्वारा Set कर सकते हैं। Container का Default Unit Twip होता है, जो कि एक Point का बीसवां हिस्सा यानी 1/20 Point के बराबर होता है।

हम इन सभी Properties को Run Time में भी Change कर सकते हैं और हम Control के Move Method का प्रयोग करके Control की Size व Position को Runtime में Change कर सकते हैं।

TabIndex and TabStop Properties

TabIndex Property द्वारा ये तय किया जाता है कि जब User Tab Key को Press करेगा, तब कौनसा Control कब Focus प्राप्त करेगा। Label Control की भी TabIndex Property होती है, लेकिन एक Label कभी भी Focus प्राप्त नहीं करता है।

TabStop Property हमें ये Specify करने की सुविधा प्रदान करता है कि जब User Tab Key को Press करेगा, तब किस Control पर Tab Navigation के दौरान Move नहीं होगा। इस Property का Default मान True होता है।

यदि हम किसी Control की TabStop Property में False मान Store कर देते हैं, तो User Navigation के दौरान उस Control पर Focus को Tab Key द्वारा Transfer करने में सक्षम नहीं हो पाता है। TabStop Property केवल उसी समय अपना Effect दिखाता है, जब User Tab Key

Press करके Navigation करता है। TabStop Property को False Set करने के बावजूद हम किसी भी अन्य तरीके से सम्बंधित Control पर Focus को Set कर सकते हैं।

Assigning Access Key To TextBox Control

हम किसी Text Box पर किसी Access Key को Focus के लिए Set नहीं कर सकते हैं। लेकिन फिर भी यदि हम किसी Text Box को Access Key द्वारा Control करना चाहें, तो हम TextBox Control के साथ Associate किए जाने वाले Label Control पर उस Key को Highlight कर सकते हैं, जिसे Use करके हम Text Box को Control करना चाहते हैं, क्योंकि Label Control कभी भी Focus प्राप्त नहीं करता है।

Access Key Set करने के लिए हम Label Control के Caption Property में किसी Key को '&' का प्रयोग करके Access Key Define कर सकते हैं। लेकिन चूंकि एक Label Control कभी भी Focus प्राप्त नहीं करता है, इसलिए जब हम Label Control के Access Key को Use करते हैं, तो Visual Basic Automatically उस Control पर Focus को Set कर देता है, जो Label के TabIndex Property के अगले क्रम पर आता है और सामान्यतया ये Label से Associated Text Box होता है, इसलिए Control Text Box पर Transfer हो जाता है। किसी Text Box पर Access Key Set करने के लिए हमें निम्न क्रम को Follow करना होता हैः

1. सबसे पहले Text Box के Left में एक Label Control को Place करें।

2. सामान्यतया सभी Controls की UserMnemonic Property True होती है, लेकिन यदि Label Control की ये Property False हो, तो सबसे पहले इसे True Set करें। जब ये Property True होती है और हम Caption Property में Specify किए जाने वाले नाम के किसी Character के आगे & का Symbol लगाते हैं, तो वह Character Access Key बन जाता है। लेकिन यदि ये Property False हो, तो Caption में Use किया जाने वाला & भी Caption के रूप में दिखाई देने लगता है।

3. अब Label Control को उसके Right में Placed Text Box के TabIndex के मान से एक कम मान Label Control को Set करना होता है। ऐसा करने का कारण ये है कि जब हम Access Key को Use करेंगे, तब Focus Label Control पर जाएगा, लेकिन चूंकि Label Control Focus Receive नहीं कर सकता है, इसलिए Focus अलग TabIndex पर Move होगा, जो कि Text Box का TabIndex Number होगा। इस तरह से Focus Text Box पर पहुंच जाएगा।

CommandButton Control – Important Properties

Window Application में TextBox व Label Controls के बाद यदि कोई Control सबसे ज्यादा Use किया जाता है, तो वह CommandButton Control है। Command Button का मुख्य काम उस समय Code Procedure को Run करना होता है, जब User Command Button पर Click करता है। जब हम Command Button को Program कर रहे होते हैं, तब हमें Command Button की निम्न मुख्य Properties का ध्यान रखना होता हैः

Caption Property

हमें CommandButton के Caption Property को हमेंशा Valid नाम Set करना होता है, क्योंकि Caption Property वह Property होती है, जो Run Time में Object पर दिखाई देती है। इस Property को Design Time व Run Time दोनों स्थितियों में Access किया जा सकता है।

Cancel Property

Command Button की Cancel Property एक Boolean Property होती है। ये Property Form पर स्थित किसी एक Command Button के Click Event Procedure Codes को Esc Key द्वारा Access करने की सुविधा देता है। हम जिस Button पर Cancel Property को True Set करते हैं, उस Button के Click Event के लिए लिखे गए Codes Keyboard पर स्थित Esc Key को Press करके Execute किया जा सकता है।

ध्यान रखें कि Esc Button को Press करने पर Click Event Fire नहीं होता है, केवल Click Event Handle करने वाला Procedure ही Run होता है। सबसे ज्यादा Notice करने वाली बात ये है कि Esc Key के Press करने पर Click Event Procedure के Execute होने के बावजूद Focus की Position उस Command Button पर Change नहीं होती है, जिसका Click Event Procedure Run हो रहा है।

Default Property

Default Property भी Cancel Property की तरह ही एक Boolean Property होती है और Form पर स्थित केवल एक ही Control के लिए Default Property को True Set किया जा सकता है। ये Property भी ठीक Cancel Property की तरह ही काम करता है, अन्तर केवल इतना है कि जिस Button पर Default Property को Set किया जाता है, उसका Click Event Handler Procedure उस समय Execute हो जाता है, जब User Keyboard से Enter Key को Press करता है।

हम एक Form पर केवल एक ही Button के लिए Cancel व Default Property को True Set कर सकते हैं, Form पर स्थित शेष सभी Commands के लिए Automatically ये Properties False Set हो जाती हैं। यदि हम चाहें तो Form पर स्थित किसी एक ही Button के लिए ये दोनों Properties True Set कर सकते हैं, इस स्थिति में Esc व Enter दोनों ही Keys को Press करने पर हर बार उसी Command Button का Click Event Procedure Execute होता है।

BackColor Style and Picture Property

Command Button की एक BackColor Property भी होती है, लेकिन ये Property उसी स्थिति में Effect दिखाती है, जब हम Command Button की Style Property को "1 – vbGraphical" Set करते हैं। जब हम ये Style Set करते हैं, तो उसके बाद हम Command Button की Picture Property को Set करके Command Button पर किसी Graphical Image को भी Display कर सकते हैं।

Value Property

Value Property Command Button की Default Property होती है और ये केवल Run Time में ही Available होती है। यदि हम इसकी Value Property को True Set कर देते हैं, तो Command Button का Click Event Procedure Execute हो जाता है। Command Button

के Click Event Procedure को Execute करने के लिए हमें Command Button की इस Default Property को True Set करना होता है, जो कि निम्नानुसार दो तरीकों से किया जा सकता है:

```
cmdOK = True          OR          cmdOK.Value = True
```

TextBox Control – Important Properties

TextBox Windows Application में काफी ज्यादा Use होने वाला Control है। इसका प्रयोग सामान्यतया User से Free Form Input प्राप्त करने के लिए किया जाता है। हम TextBox Control में किए जाने वाले Changes व TextBox Control के Contents को कई Properties का प्रयोग करके Manipulate कर सकते हैं। TextBox Control की कुछ Most Useful Properties निम्नानुसार हैं:

HideSelection Property

TextBox की ये Property By Default True होती है और ये इस बात को Indicate करती है कि User ने TextBox में जिन Texts को Highlight किया है, वे उस समय Highlight नहीं रहेंगी, जब User Focus को अन्य Control पर Move करेगा। यदि हम चाहें कि Focus किसी भी Control पर हो, लेकिन TextBox का Selected Text Highlighted ही रहे, तो हमें इस Property को False Set करना होता है।

MaxLength Property

TextBox की ये Property इस बात को तय करती है कि User Text Box में कितने Characters Input कर सकता है। यदि हम इस Property में 0 Set करते हैं, तो User अधिकतम 32 KB तक के Characters एक TextBox में Input करता है।

Locked Property

TextBox की इस Property को True करने का मतलब ये होता है कि User इस Text Box पर Focus तो Set कर सकेगा, लेकिन Text Box में किसी भी तरह का Change नहीं कर सकता है। हालांकि User उस Text Box के Contents को Scroll कर सकता है। हम इसे ऐसी Enabled Property के रूप में समझ सकते हैं, जो Focus को प्राप्त करने में सक्षम है।

MultiLine Property

TextBox की ये Property केवल Design Time में ही उपलब्ध होती है। सामान्यतया सभी Text Box Default रूप से केवल एक ही Line के Text को Hold करने में सक्षम होते हैं, जिसमें अधिकतम 256 Characters Hold हो सकते हैं। लेकिन जब हम इस Property को True Set करते हैं, तब हम एक Text Box में एक से ज्यादा Lines में Texts Enter कर सकते हैं और Text Box पर Automatically Word-Wrapping की Property Apply हो जाती है, इसलिए TextBox में Input किए जा रहे Texts जैसे ही TextBox की Boundary को Touch करते हैं, Automatically Next Line में Transfer हो जाते हैं।

PasswordChar Property

TextBox की इस Property में हम जिस Character को Set कर देते हैं, Run time में उस Text Box में Input किए जाने वाले Actual Texts दिखाई नहीं देते हैं, बल्कि वह Text दिखाई देता है, जिसे हमने इस Property में Set किया होता है। Password Input करते समय हम जिस Text Box में Password Input कर रहे होते हैं, उस Text Box की इस Property में सामान्यतया "*" Set किया गया होता है, इसलिए Password का हर Character हमें एक "*" के रूप में ही दिखाई देता है।

ScollBars Property

TextBox की इस Property को Set करके हम Text Box के साथ Scroll Bars को Set कर सकते हैं। लेकिन ये Scrollbars Text Box पर तभी दिखाई देते हैं, जब हम MultiLine Property को True Set करते हैं। हम मुख्यतः तीन तरह के Scroll Bars को TextBox पर Set कर सकते हैं, जिनकी जानकारी Scrollbars Property के List Box में दिखाई देती है।

SelText Property

TextBox में हम जिस Text को Select करते हैं या जिस Content को Return करवाना चाहते हैं, उसे इस Property में Specify करते हैं। हम इस Property में जो भी String Type करते हैं, Currently Highlighted Text इस Text से Replace हो जाता है।

SelStart Property

इस Property में एक Integer मान होता है, जो TextBox के First Highlighted Character की Position को Represent करता है। ये Zero Based होता है इसलिए यदि इसमें Zero हो, तो इसका मतलब है कि हमने Text Box के First Character पर Cursor को Place कर रखा है। यदि Text Box में Currently किसी Text को Select ना किया गया हो, तो ये Property TextBox में Cursor की Current Position को Represent करता है।

यदि हम इस Property का मान Change करते हैं, तो हमने TextBox में जो कुछ भी Select किया होता है, वह सबकुछ Deselect हो जाता है और हमारा Cursor उस Position पर पहुंच जाता है, जिसे हमने SelStart Property में Set किया है।

SelLength Property

TextBox की इस Property में भी एक Integer मान होता है, जो कि TextBox में Select किए गए Characters की संख्या को Represent करता है। हम इस Property को Run Time में Change करके Selected Texts के Characters की संख्या को Change कर सकते हैं। यदि हम इसमें 0 Set कर दें, तो Currently Selected सभी Characters Deselect हो जाते हैं।

Text Property

Text Box की Caption Property नहीं होती है, बल्कि Text Property होती है, जो कि TextBox की Default Property होती है। हम इस Property को Design Time व Run Time दोनों स्थितियों में इस Property को Access कर सकते हैं। Text Property Text Box के Currently Visible व Editable Text को Represent करता है। चूंकि ये TextBox की Default Property

होती है, इसलिए हम इसे Specify किए बिना भी Text Box के Texts को Set अथवा Return करवा सकते हैं। जैसे :

```
txtName = “Rahul”        OR        txtName.Text = “Rahul”
```

Label Control – Important Properties

Label Control की कुछ मुख्य Properties निम्नानुसार हैं:

Alignment Property

Label Control की String को इस Property द्वारा Left, Center अथवा Right में Align किया जाता है।

Appearance, BackStyle and BorderStyle Properties

Label Control की ये तीनों Properties Label की General Appearance से सम्बंधित Properties हैं।

उदाहरण के लिए Label की Default Appearance 3D, Default BackColor vbWhite व Default BorderStyle FixedSingle होता है, जिसे हम जरूरत के आधार पर Change कर सकते हैं। हम एक Label को TextBox की तरह का Look दे सकते हैं। हम इसकी BackStyle जो कि Normally Opaque होती है, उसे Transparent बना सकते हैं

AutoSize and WordWrap Property

Label Control की AutoSize Property को True करने पर Label अपने Caption के अनुसार स्वयं ही Adjust हो जाता है। जबकि WordWrap Property को True करने पर इसमें Type किया जाने वाला Caption Automatically Next Line में Shift हो जाता है।

WordWrap Property उसी स्थिति में Vertically Text को Shift करता है, जब उसकी AutoSize Property को True Set किया गया होता है। यदि हम केवल AutoSize Property को True Set करें, तो Caption केवल Horizontally ही Increase होता है।

Caption Property

Label Control की Caption Property Label की Default Property होती है। हम Caption को Run Time में भी Change कर सकते हैं। हम इस Property को बिना Specify किए हुए Directly Access कर सकते हैं। जैसे:

```
lblName = “Rahul”        OR        lblName.Caption = “Rahul”
```

Events and Event Procedures

विभिन्न प्रकार के Controls उसी स्थिति में Response करते हैं, तब हम उन Controls द्वारा Generate की जाने वाली Events को Handle करने के लिए Event Procedures लिखते हैं। Event कोई भी ऐसा User अथवा System Action हो सकता है, जिसके Reaction के रूप में किसी काम को पूरा किया जा सके।

उदाहरण के लिए Command Button पर Click करना अथवा TextBox के Text को Change करना आदि विभिन्न प्रकार के Events के उदाहरण हैं और इन Events को Handle करते हुए Application में किसी काम को पूरा करवाने के लिए ऐसे Procedures लिखना, जो कि इन Events के Response में Execute हो जाएं, Event Procedures या Event Handlers के उदाहरण हैं।

Visual Basic Programming मुख्यतः इस Concept पर आधारित है कि "किस Control के साथ किस Action के Response में क्या Reaction होना चाहिए।" जैसे यदि CommandButton cmdOK Control पर Click Event Generate हो, तो जो–जो काम होने चाहिएं, उन्हें cmdOK_Click() नाम के Procedure में Code कर लिया जाता है। ये Procedure ही Event Handler या Event Procedure कहलाता है, जो उस स्थिति में Automatically Execute हो जाता है, जब हम cmdOK नाम के Button पर Mouse से Click करते हैं।

किसी भी Event Procedure का Header Create करने के लिए हमें Design Time में केवल उस Control पर Double Click करना होता है, जिसके लिए हम Event Handler Create करना चाहते हैं। जब हम ऐसा करते हैं, तब उस Control व उसके Default Event के साथ Automatically एक Event Procedure Header बन जाता है।

Command Button के लिए Click Event, TextBox Control के लिए Change Event व Label Control के लिए Click Event Default Events होते हैं। यदि हम Default Event के स्थान पर किसी अन्य Event को Use करना चाहें, तो Code Window में दिखाई देने वाले Events के Combo Box में से Required Event को Select करके वांछित Control के साथ Associated नया Event Handler Header Create कर सकते हैं।

The Click Event

Click Event तब Generate होता है, जब User किसी Enabled Control पर Mouse Button को Press करके Release करता है। यदि Mouse Pointer किसी Disabled Control पर हो अथवा यदि Mouse Pointer Form के किसी Blank Area पर हो, तो उस स्थिति में Form ही Click Event Receive करता है, वह Control नहीं जो कि Disabled होता है।

Click Event केवल उसी स्थिति में Generate होता है, जब User Same Control पर Mouse का Pointer Place रखते हुए Mouse के Button को Press व Release करता है। यदि User Mouse के Pointer को एक Control पर रखते हुए Mouse के Button को Press करता है और Mouser के Button को किसी दूसरे Control पर Move करते हुए Release करता है, तो Click Event Generate नहीं होता है। Click Event उस स्थिति में भी Fire होता है, जब User Control की Access Key को Press करता है।

CommandButton

इस Control के लिए Mouse का केवल Left Button ही Click Event Fire करता है। जब किसी Command Button पर Focus होता है, तब Spacebar अथवा Enter Key को Press करने पर भी Click Event Fire हो जाता है। Programmatically इसके Value Property को True Set करके भी Click Event Procedure को Run कर देता है, लेकिन इस प्रक्रिया से Click Event Fire नहीं होता है।

Label

इस Control के लिए Mouse के दोनों Buttons Click Event Fire करते हैं।

TextBox

इस Control के लिए Mouse के दोनों Buttons Click Event Fire करते हैं।

The DblClick Event

ये Event तब Fire होता है, जब Control Enabled होता है और User तेजी से दो बार Click करता है। Operating System दोनों Clicks के बीच के Timing के आधार पर ये तय करता है कि User ने दो बार Single Click किया है अथवा एक बार Double Click किया है। Command Button के लिए इस Event को Visual Basic में Specify नहीं किया गया है, लेकिन हम Label व Textbox Control के लिए इस Event को Use कर सकते हैं।

MouseUp and MouseDown Events

जब User किसी Control पर Mouse के Button को Press करता है, तब MouseDown Event Fire होता है और जब User Control पर Press किए गए Mouse के Button को Release करता है, तब MouseUp Event Trigger होता है। यदि User किसी Control पर Mouse के Button को Press करता है, लेकिन Mouse को Move करके वह किसी अन्य Control पर Mouse के Button को Release करता है, तो इस स्थिति में भी वही Control MouseUp Event प्राप्त करता है, जिसने MouseDown Event प्राप्त किया था।

MouseDown व MouseUp Event के दौरान किस Shift Key को Press किया गया था अथवा Mouse के किस Button से MouseUp व MouseDown Event को Generate किया गया था, इस बात की जानकारी हमें क्रमशः Shift व Button Arguments से प्राप्त हो सकती है, जो कि MouseUp व MouseDown Event Procedure के Parameters होते हैं।

Mouse Move के अलावा सभी अन्य Mouse Events Mouse Buttons के Action से Directly Related होते हैं। जब User किसी Label अथवा TextBox पर Click करता है, तब निम्न क्रम में विभिन्न Events Fire होते हैं:

```
MouseDown
MouseUp
Click
```

जब User Command Button पर Click करता है, तब Trigger होने वाले विभिन्न Events निम्न क्रमानुसार होते हैं:

```
MouseDown
Click
MouseUp
```

जब User किसी TextBox अथवा Label पर Double Click करता है, तब विभिन्न Events निम्न क्रम में Fire होते हैं:

```
MouseDown
MouseUp
Click
DblClick
MouseUp
```

इसी तरह से जब User Command Button पर Double Click करता है, तब निम्न क्रम में विभिन्न Events Generate होते हैं, क्योंकि Command Button Double Click Event को Support नहीं करता है:

```
MouseDown
Click
MouseUp
MouseDown
Click
MouseUp
```

The MouseMove Event

जब भी Mouse को किसी Control पर Move किया जाता है, तब MouseMove Event Fire होता है। ये Events काफी तेजी से Execute होता है। MouseMove Event में भी वही सब Parameters होते हैं, जो MouseDown व MouseUp Event में होते हैं।

The Change Event

ये Event TextBox Control का Default Event होता है। जब भी हम किसी Text Box के Text Property को Change करते हैं, ये Event Trigger हो जाता है। User Input के समय भी ये Event Trigger होता है, क्योंकि Input के समय User इसकी Text Property को ही Change कर रहा होता है।

Label Control भी उस स्थिति में Change Event Trigger करता है, जब Run Time में इसके Caption Property को Change किया जाता है। हालांकि User का Input Label के Change Event को कभी भी Fire करने में सक्षम नहीं होता है। ये Event केवल तभी Fire होता है, जब User Run Time में Label के Caption को Change करता है।

The KeyPress Event
जब User Keyboard से किसी Key को Press करके Release करता है, तब ये Event Trigger होता है।

The KeyUp Event
जब User Keyboard से किसी Key को Press करके Release करता है, तब ये Event Trigger होता है।

The KeyDown Event
जब User Keyboard से किसी Key को Press करता है, तब ये Event Trigger होता है।

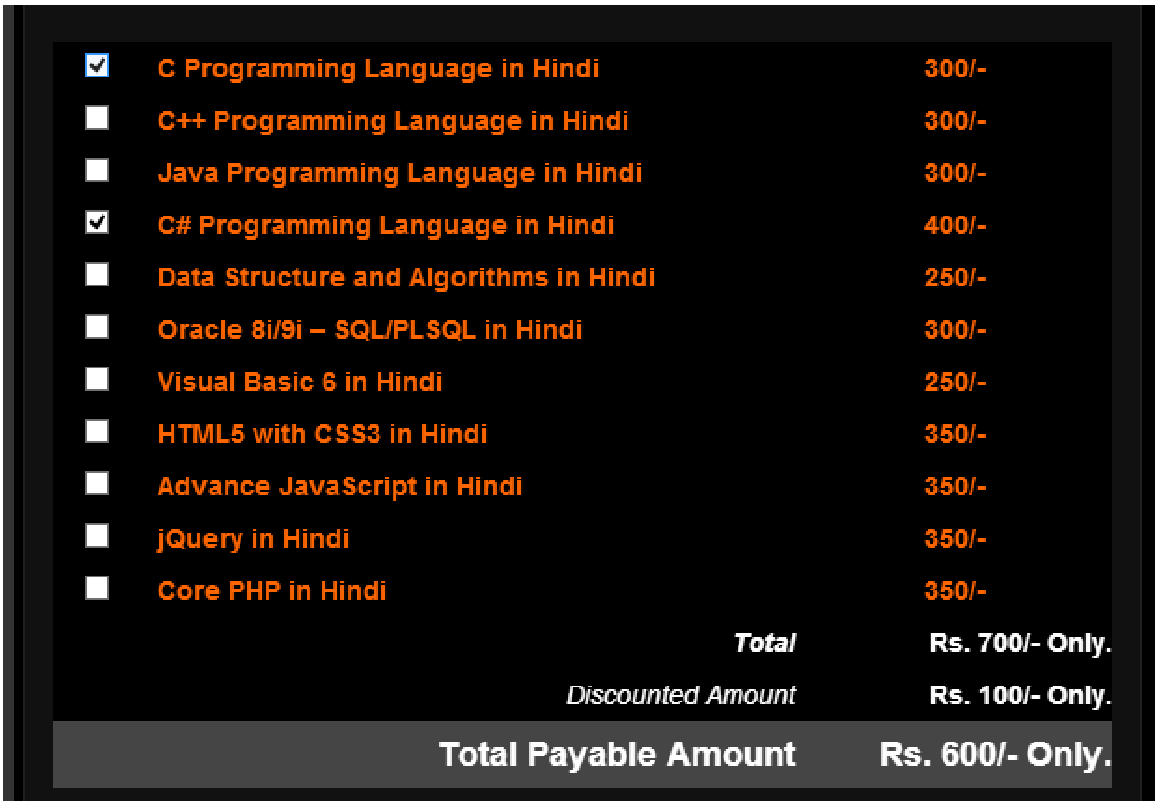
The GotFocus and LostFocus Event
जब किसी Control का Focus Lost होकर किस अन्य Control पर Transfer होता है, तब जिस Control का Focus Lost होता है, उसका LostFocus Event Trigger होता है और जिस Control को Focus Transfer होता है, उसका GotFocus Event Trigger होता है।

How to Get this Ebook in PDF Format

ये पुस्तक केवल **PDF Format Ebook** के रूप में ही Available है और आप इस पुस्तक को केवल हमारी **Official Website** (http://www.bccfalna.com/) से ही खरीद सकते हैं। इसलिए यदि आपको ये पुस्तक पसन्द आ रही है और आप इसे PDF Format Ebook के रूप में खरीदना चाहते हों, तो आप इस पुस्तक को Online खरीदने के लिए निम्नानुसार दिए गए **3 Simple Steps** Follow कर सकते हैं:

Select Purchasing EBooks

सबसे पहले **http://www.bccfalna.com/how-to-pay/** Link पर Click कीजिए। जैसे ही आप इस Link पर Click करेंगे, आप हमारी Website के निम्नानुसार **Order Page** पर पहुंच जाऐंगे :



इस Page पर आपको उन पुस्तकों को Select करना है, जिन्हें आप खरीदना चाहते हैं। आप जैसे–जैसे पुस्तकें Select करते जाऐंगे, आपको उनका **Total Amount, Discount** व **Total Payable Amount** उपरोक्त चित्रानुसार दिखाई देने लगेगा, जहां Total Payable Amount ही वह Amount है, जो आपको अपनी Selected EBooks को खरीदने के लिए **Pay** करना होगा।

पुस्तकें Select करने के बाद इसी Page पर दिखाई देने वाले "**Order Details**" Form में आपको निम्न चित्रानुसार अपना *Name, Email Address* व *Mobile Number* Specify करके "**Order Now**" पर Click करते हुए उपरोक्त Selected EBooks का Order Place करना होगा:

चूंकि ये सारी पुस्तकें Physical Books नहीं बल्कि **PDF Format Ebooks** हैं। इसलिए ये पुस्तकें आपको आपके Email पर ही भेजी जाऐंगी, जिन्हें आप अपने Email के माध्यम से अपने Computer पर Download करके अपने PDF Supported *Computer*, *Mobile*, *Smart Phone*, *Tablet PC*, *Net-Book*, *Notebook* या *Laptop* जैसी किसी भी Device के माध्यम से पढ सकते हैं अथवा यदि आप चाहें, तो अपने Printer द्वारा इन पुस्तकों का Hard Copy Printout निकाल सकते हैं।

इसलिए जरूरी है कि उपरोक्त "**Order Details**" Form पर आप जो **Email Address** व **Mobile Number** Specify करते हैं, वह Working और एकदम सही हो। क्योंकि किसी भी तरह की परेशानी की स्थिति में हम आपको आपके **Mobile Number** पर ही Contact करते हैं।

Pay "Total Payable Amount"

जैसे ही आप "**Order Now**" Button पर Click करेंगे, आपको एक Email मिलेगा, जिसमें आप द्वारा Order की गई EBooks की Details होगी। Selected पुस्तकों का Order Place करने के बाद अब आपको "**Total Payable Amount**" का Payment करना होगा।

यदि आपके पास **Net-Banking** या **Mobile-Banking** की सुविधा है, तो आप Payment करने के लिए अपने Account में Login करके निम्न में से किसी भी Bank A/c में Payment Deposit कर सकते हैं:

भारतीय स्टेट बैंक
State Bank of India
With you - all the way

SBI Bank A/c no.	:	**31154882587**
Account Name	:	**Namita Mishra**
Branch Name	:	**Falna**
Address	:	**Near Railway Crossing, Falna Station – 306116**
IFSC Code	:	**SBIN0007868**

बैंक ऑफ़ बड़ौदा
Bank of Baroda
India's International Bank

BOB Bank A/c no.	:	**35260100003212**
Account Name	:	**Namita Sharma**
Branch Name	:	**Falna**
Address	:	**Sanderao Road, Falna, Dist. Pali (Raj.)- Pin-306116**
IFSC Code	:	**BARB0FALNAX**

स्टेट बैंक ऑफ बीकानेर एण्ड जयपुर
State Bank of Bikaner and Jaipur
The Bank with a vision

SBBJ Bank A/c no.	:	**61089986732**
Account Name	:	**Kuldeep Chand Mishra**
Branch Name	:	**Bali**
Address	:	**Sr. Secondary School Road, Bali- 306701**
IFSC Code	:	**SBBJ0010193**

punjab national bank
...The name you can BANK upon!

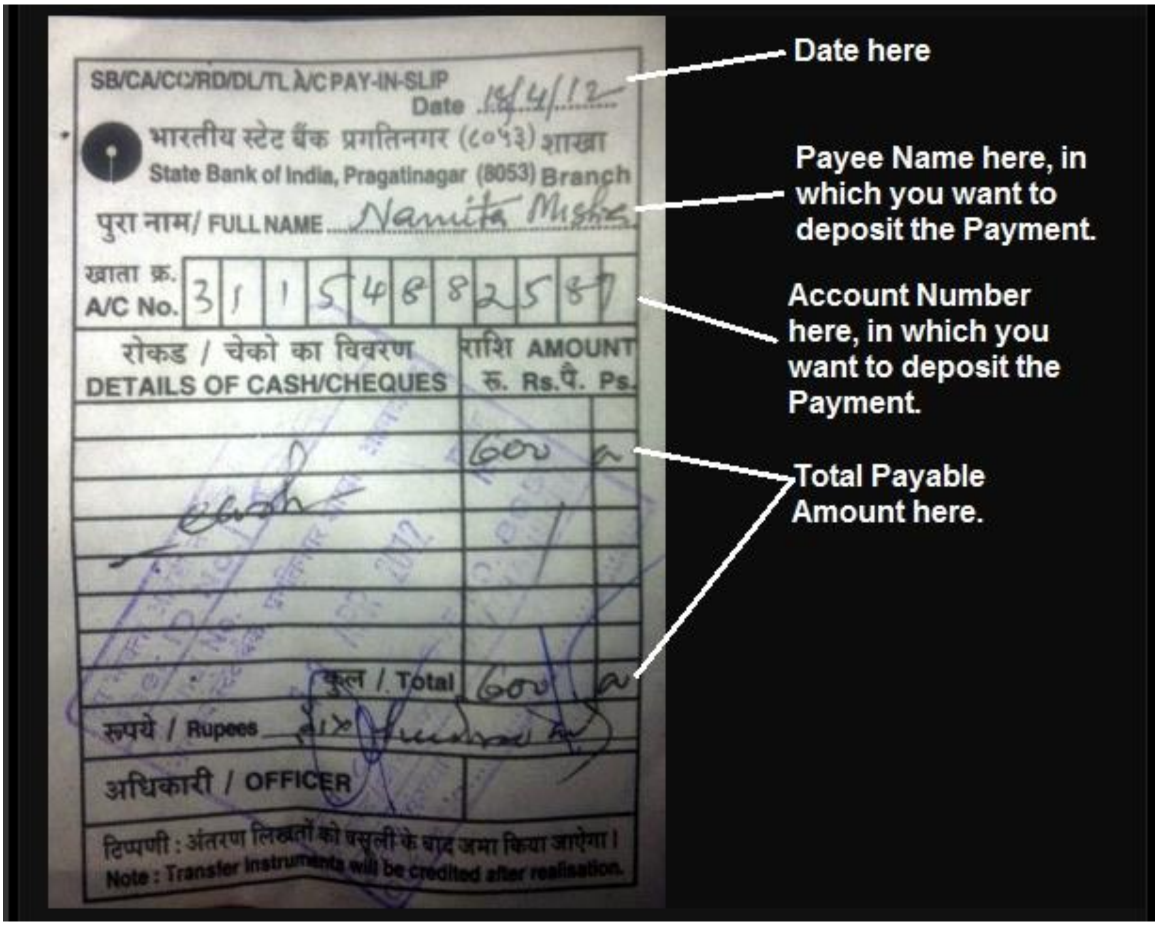
PNB Bank A/c no.	:	4445000100034960
Account Name	:	Kuldeep Chand
Branch Name	:	Falna
Address	:	College Road, Falna Station – 306116
IFSC Code	:	PUNB0444500

जब आप Net-Banking के माध्यम से Payment करना चाहते हैं, तो आपको लगभग **8 से 24 घण्टे पहले** हमारे उस Account को **Beneficiary** के रूप में अपने Bank A/c से Link करना पडता है, जिसमें आप Payment Deposit करना चाहते हैं।

यदि आपके पास Net-Banking या Mobile-Banking की सुविधा नहीं है, तो आप हमारे किसी भी Bank A/c में *Total Payable Amount*, **Direct Deposit** भी कर सकते हैं।

जब आप **Direct Deposit** करना चाहते हैं, तब आपको आपके किसी भी नजदीकी Bank Branch में जाकर एक *Payment Deposit Slip* Fill-Up करना होता है, जिसमें आपको हमारे किसी भी Bank A/c की Information को Fill करना होता है, **जबकि Payment Deposit करवाने के लिए उसी Bank में आपका स्वयं का Account होना जरूरी नहीं है।**

उदाहरण के लिए यदि आप हमारे SBI Bank A/c में अपनी Selected पुस्तकों का *Total Payable Amount* Pay करने के लिए Bank में जाकर Direct Deposit करना चाहते हैं, तो आप जो **Payment Deposit Slip** Fill-Up करेंगे, वह निम्न चित्रानुसार करना होता है:



इस चित्र द्वारा आप समझ सकते हैं कि Payment, Direct Deposit करने के लिए आपको हमारे किसी Bank A/c की Information को *Payment Deposit Slip* में Specify करना होता है, इसलिए उस Bank में आपका स्वयं का Bank A/c होना जरूरी नहीं होता।

Net-Banking, Mobile-Banking व **Direct Deposit** के अलावा किसी अन्य माध्यम से भी आप Payment कर सकते हैं। उदाहरण के लिए कुछ Banks अपनी ATM Machine द्वारा Direct Payment Transfer करने की सुविधा Provide करते हैं।

यदि आपके Bank का ATM Machine इस तरह से Payment Transfer करने की सुविधा देता है, तो आपको Bank में जाकर Payment Deposit Slip के माध्यम से Payment करने की जरूरत नहीं होती, बल्कि आप Bank के ATM Machine से भी Directly हमारे किसी भी Bank A/c में **Total Payable Amount** Transfer कर सकते हैं। इसी तरह से यदि आप चाहें, तो हमारे किसी भी Bank A/c में Check द्वारा भी Amount Direct Deposit कर सकते हैं।

यानी आप किसी भी तरीके से हमारे किसी भी Bank A/c में *Total Payable Amount* Deposit कर सकते हैं। लेकिन हम **Money-Order, Demand-Draft** या **Check** जैसे Manual माध्यमों से Payment Accept नहीं करते, क्योंकि इस तरह का Payment Clear होने में बहुत समय लगता है। जबकि Direct Deposit या Mobile अथवा Net-Banking के माध्यम से तुरन्त Payment Transfer हो जाता है, जिससे हम आपको आपकी Purchased EBooks **20** से **30 Minute** के दरम्यान आपके Order में Specified **Email** पर Send कर देते हैं।

Confirm the Payment

जब आप अपनी Selected पुस्तकों को खरीदने के लिए उपरोक्तानुसार किसी भी तरीके से *"Total Payable Amount"* हमारे किसी भी Bank A/c में Deposit कर देते हैं, तो Payment Deposit करते ही आपको हमें उसी Mobile Number से एक **Call/Miss Call/SMS** करना होता है, जिसे आपने Order Place करते समय "**Order Form**" में Specify किया था।

इसी Mobile Number के माध्यम से हमें पता चलता है कि आपने किन पुस्तकों के लिए Order किया है और उनका *Total Payable Amount* कितना है। साथ ही हमें ये भी पता चल जाता है कि आप द्वारा Purchase की जा रही पुस्तकें किस Email Address पर Send करनी है।

आपके *Total Payable Amount* को हम Net-Banking के माध्यम से अपने Bank A/c में Check करते हैं और यदि आपका *Total Payable Amount* हमारे किसी भी Bank A/c में Deposit हुआ होता है, तो हम आपको **30 Minute** के दरम्यान आपकी Ordered EBooks आपके Email पर Send कर देते हैं, जिसे आप अगले दिन 12AM तक Download कर सकते हैं।

यदि अभी भी आपको कोई बात ठीक से समझ में न आ रही हो या किसी भी तरह का Confusion हो, तो आप **097994-55505** पर **Call/Miss Call/SMS** कर सकते हैं। यथा सम्भव तुरन्त आपको Callback किया जाएगा और आपकी समस्या या Confusion का Best Possible Solution करने की कोशिश की जाएगी।

उम्मीद है, इस पुस्तक के Sample Chapters का Demo भी आपको पसन्द आया होगा और हमें पूरा विश्वास है कि पूरी पुस्तक आपको और भी ज्यादा पसन्द आएगी।