



International Applications

based on

- Dr. International, Developing International Software 2nd Edition, Microsoft 2003
- G. Smith-Ferrier, .NET Internationalization: The Developer's Guide to Building Global Windows and Web Applications, Addison Wesley Professional, 2006

Locale

- The collection of features of the user's environment that is dependent on language, country/region, and cultural conventions
- In software terms, a locale is a set of information associated with a place or a culture
 - name and identifier of the spoken language
 - script used to write the language
 - cultural conventions
 - it determines: keyboard layout, date, time, number, and currency format
- Windows XP supports 135 locales

Locale Samples

	English (US)	Polish	Japanese	UAE Arabic
Country	United States	Poland	Japan	UAE
Language	English	Polish	Japanese	Arabic
Written Script	Latin	Central Europe	Kana, kanji	Arabic
Direction of Text	Left to right	Left to right	Left to right horz. or right to left vert.	Right to left
Code Page	1252	1250	932	1256
Currency Symbol	\$	zł	¥	د.إ.
Long-Date Format	January 27, 2004	27 stycznia 2004	2004 年1月27日	٢٧ يناير ٢٠٠٤
Short-Date Format	1/27/04	2004-01-27	04/01/27	04/01/27
Time Format	1:00 pm	13:00	13:00	١:٠٠ م
Calendar	Gregorian	Gregorian	Gregorian (Localized)	Gregorian (Localized)
Paper Size	U.S. Letter	A4	A4	A4
Decimal Separator	.	,	.	,
List Separator	,	;	,	;
Thousands Separator	,	(space)	,	,

Single, World-Ready Binary

■ Advantages:

- reduction of development hassle and costs
- no need of conditional compiling
- no need to maintain separate source codes and development teams
- all language versions can be shipped almost simultaneously
- software updates are simpler – the same version of patches can be applied to all languages
- simpler for customers' global Information Technology (IT) departments (a single service pack instead of many different versions)

Globalization

The process of developing a program core whose features and code design are not solely based on a single language or locale.

Globalization Tasks

- Write fully implemented Unicode applications
- Don't make locale-specific assumptions when displaying data
- Handle different languages and methods for entering text
- Don't depend on a given font
- Be aware that you might need to handle complex scripts
- Allow the user to select the language of the user interface (UI)
- Create an easy-to-localize UI (localizability)
- Make sure your UI is mirroring-aware for bidirectional languages

Unicode

■ UTF-8

- 1 byte for US-ASCII characters (U+0000 to U+007F)
- 2 bytes for Latin, Greek, Cyrillic, Armenian, Hebrew, Arabic, Syria and Thaana (U+0080 to U+07FF)
- 3 bytes for Basic Multilingual Plane
- 4 bytes for UFT-16 surrogate pairs

■ UTF-16

- 2 bytes per character
- surrogate pairs – pairs of Unicode (UTF-16) characters

■ UTF-32

- 4 bytes per character

Windows Support for Unicode

- Windows 95, 98, and Me are not Unicode-based
 - Microsoft Layer for Unicode adds support for Unicode
- Windows NT 3.51 supports Unicode
- Since Windows NT 4 Unicode is a native encoding
 - the fundamental representation of text is UTF-16
 - texts of non-Unicode applications are converted

Polish Diacritics

		ą	ć	ę	ł	ń	ó	ś	ź	ż
Unicode	Dec	50309	50311	50329	50562	50564	50099	50587	50618	50620
	Hex	C485	C487	C499	C582	C584	C3B3	C59B	C5BA	C5BC
ISO-8859-2	Dec	177	230	234	179	241	243	182	188	191
	Hex	B1	E6	EA	B3	F1	F3	B6	BC	BF
Windows CP-1250	Dec	185	230	234	179	241	243	156	9C	191
	Hex	B9	E6	EA	B3	F1	F3	159	9F	BF

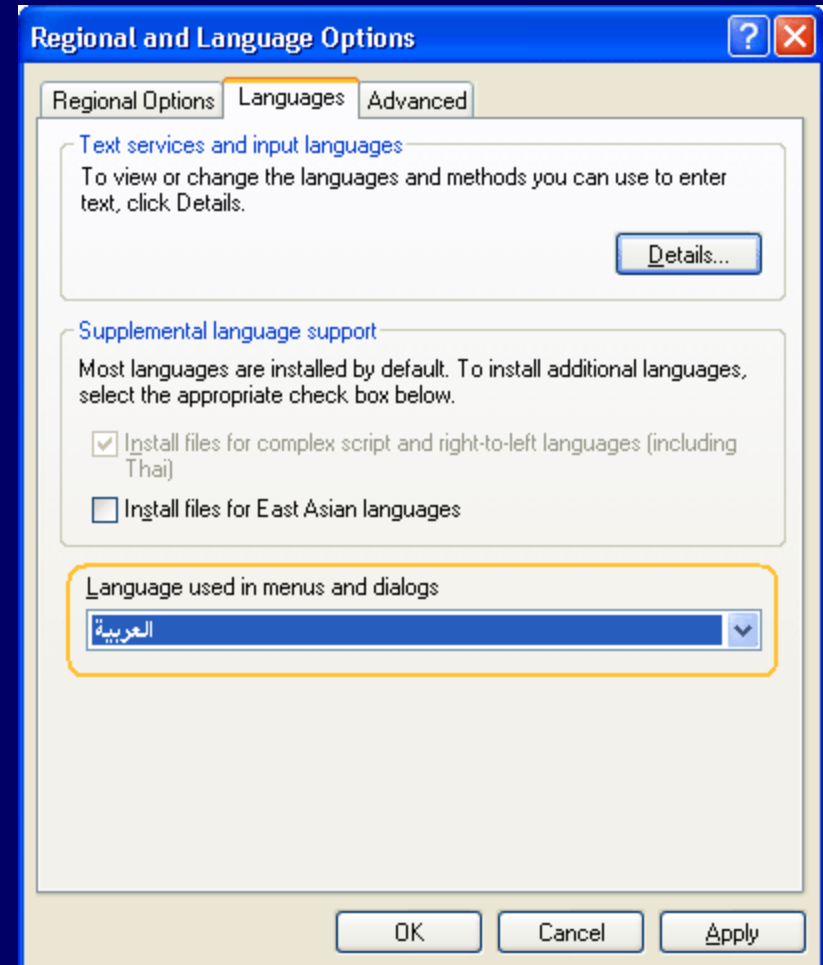
		Ą	Ć	Ę	Ł	Ń	Ó	Ś	Ź	Ż
Unicode	Dec	50308	50310	50328	50561	50563	50067	50586	50617	50619
	Hex	C484	C486	C498	C581	C583	C393	C59A	C5B9	C5BB
ISO-8859-2	Dec	161	198	202	163	209	211	166	172	175
	Hex	A1	C6	CA	A3	D1	D3	A6	AC	AF
Windows CP-1250	Dec	165	198	202	163	209	211	140	143	175
	Hex	A5	C6	CA	A3	D1	D3	8C	8F	AF

Locale and Cultural Awareness

- Date and calendar
- Time
- Currency
- Casing
- Sorting and string comparison
- Number
- Addresses
- Paper size
- Telephone numbers
- Units of measure

Multilingual User Interface (MUI)

- Set of language-specific resource files that can be added to the English version of Windows 2000, Windows XP Professional, and Windows Vista Enterprise and Ultimate
- UI language of the operating system to be changed to one of many supported languages



Localizability

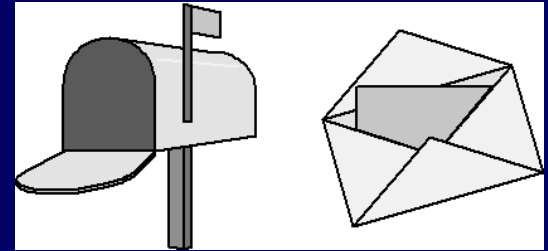
The design of the software code base and resources such that a program can be localized into different language editions without any changes to the source code.

Isolating Localizable Resources

- Localizable items:
menus, messages, dialog boxes, prompts, images, sounds, toolbars, status bars, constants
- Types of strings:
 - UI resources – must be localized
 - font and locale information, folder names, account names, etc.
– should be localized but with cautiousness
 - debug strings – there is no need of localization
 - functional resources (registry keys, strings communicated between components etc.) – must not be localized

Localizability Considerations for the UI

- Element resizing
- Localizability of UI controls
- Images and icons
- Content localizability guidelines:
 - keep content simple
 - follow basic writing style principles
 - respect cultural and local sensitivity (also in art and multimedia)
 - write for easy recycling and reduced localization costs
 - design the help system with global functionality



Localization

The process of adapting a program for a local market.

Localization

- Translation
 - all of a product's text, menus, dialog boxes, buttons, wizards, online Help, printed documentation, packaging, and CD labels
 - multimedia files (synchronization of spoken text)
- Adaptation to the particular locale
 - currency, address, number, and date formats
 - sort order
 - font
 - right-to-left layout
- The final localized version of the original software product should look and feel as if it had been designed in the user's home country

Localization Elements

- Text
- Layout
- Graphics and multimedia
- Keyboard shortcuts
- Fonts
- Locale data and character set
- Build process and packaging

Testing

Deployment and General Functionality

- Application setup under different language environments
- Application setup on an operating system with MUI installed and where user's default UI language is different from English
- General functionality with various system and user-locale settings
- Uninstalling the application with different locale settings
- Multiplatform installation

Text Handling

- Interactive text input using different input locales
- Clipboard operations with multilingual text
- Device-independent multilingual output
- Font independence
- Text handling in the UI
- Cursor movement and positioning
- Handling of character encoding-double byte character set (DBCS)
- Handling of OEM and Windows encodings
- Handling of complex scripts
- Handling buffer sizes for multibyte character set (MBCS) text
- Language-independent data persistence

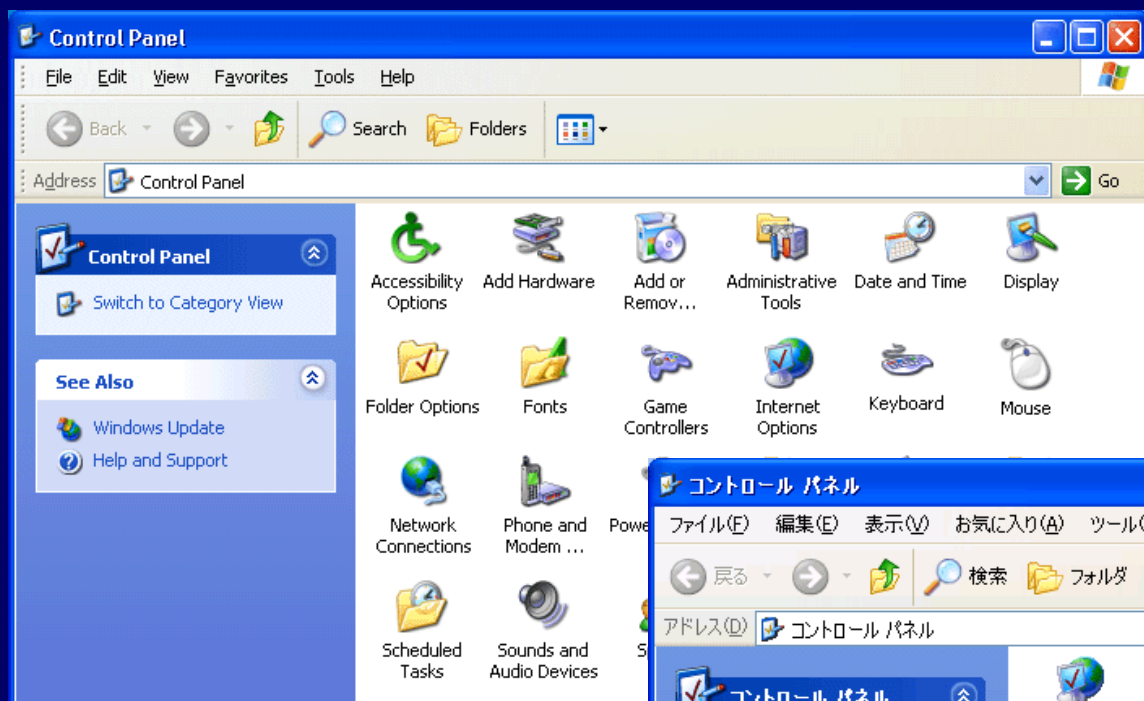
Locale Awareness

- Locale-independent data persistence
 - convert to local before displaying
- Adherence to locale standards
 - A.M./P.M., names of months, currency symbol etc.
- Dynamic usage of format separators
- Paper and envelope sizes
- Measurement-systems independence

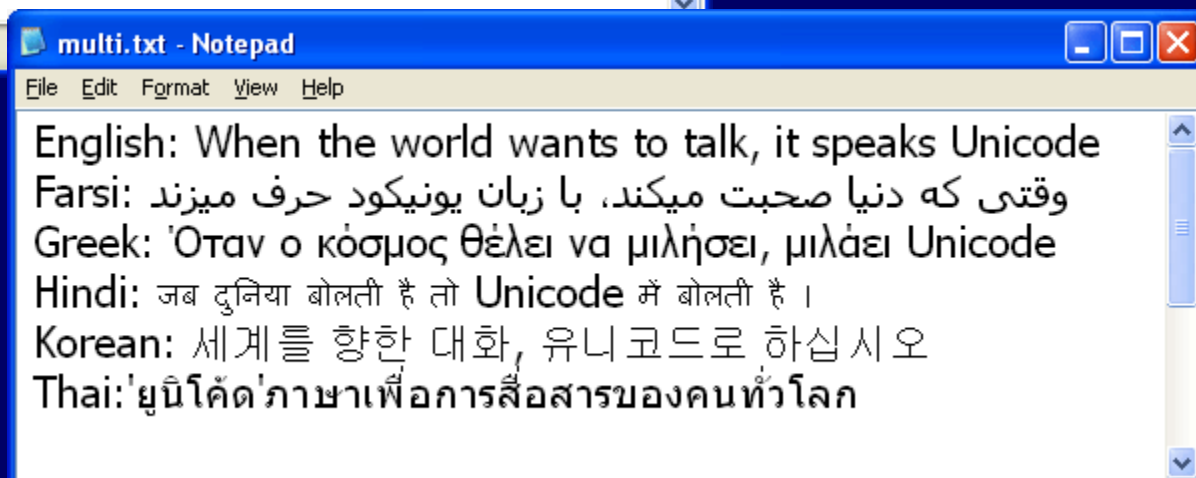
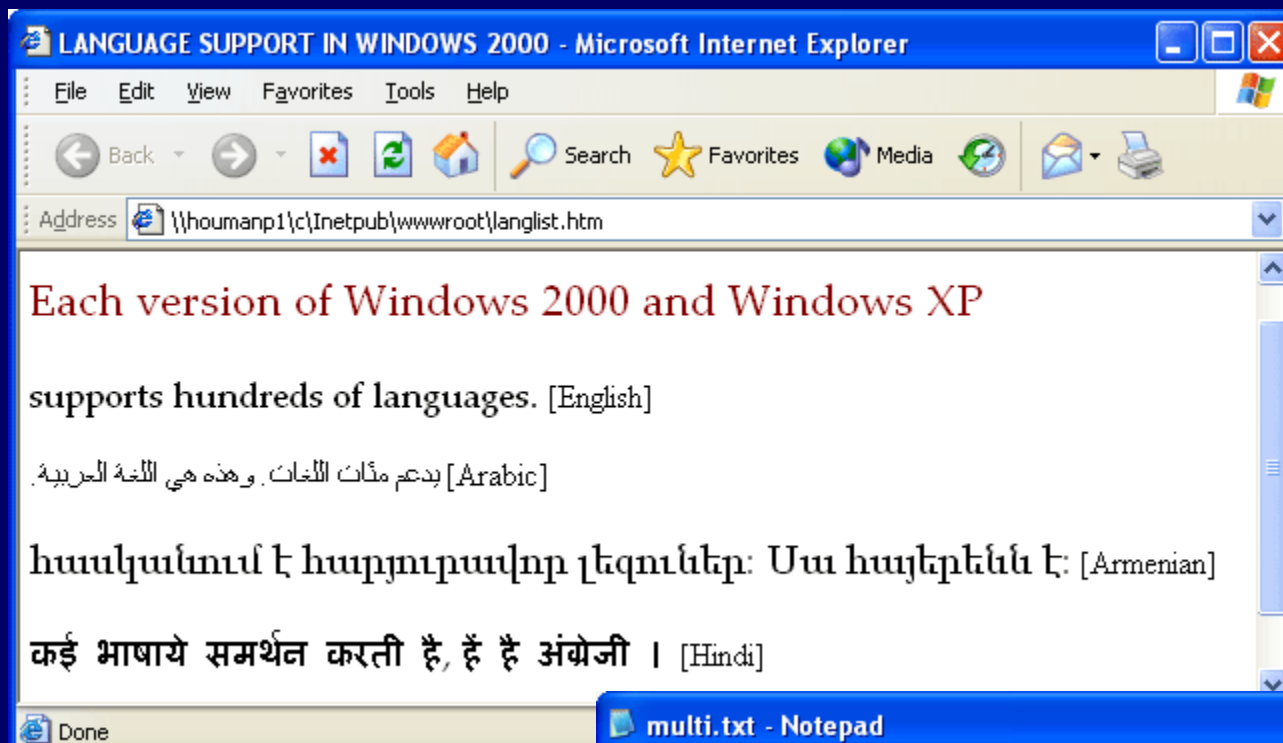
Localizability

- Isolation of the localizable resources
- Locale-independent handling of resources
- Handling of string dependencies
- Language-independence of the file names
- Font-size and display-resolution independence
- Localization-independent functionality
- UI adjustable to the UI text growth
- Mirroring-awareness in UI composition
- Localizability of nontextual resources
- Run-time string composition
- Dynamic retrieval of system-defined names (usernames, folders etc.)

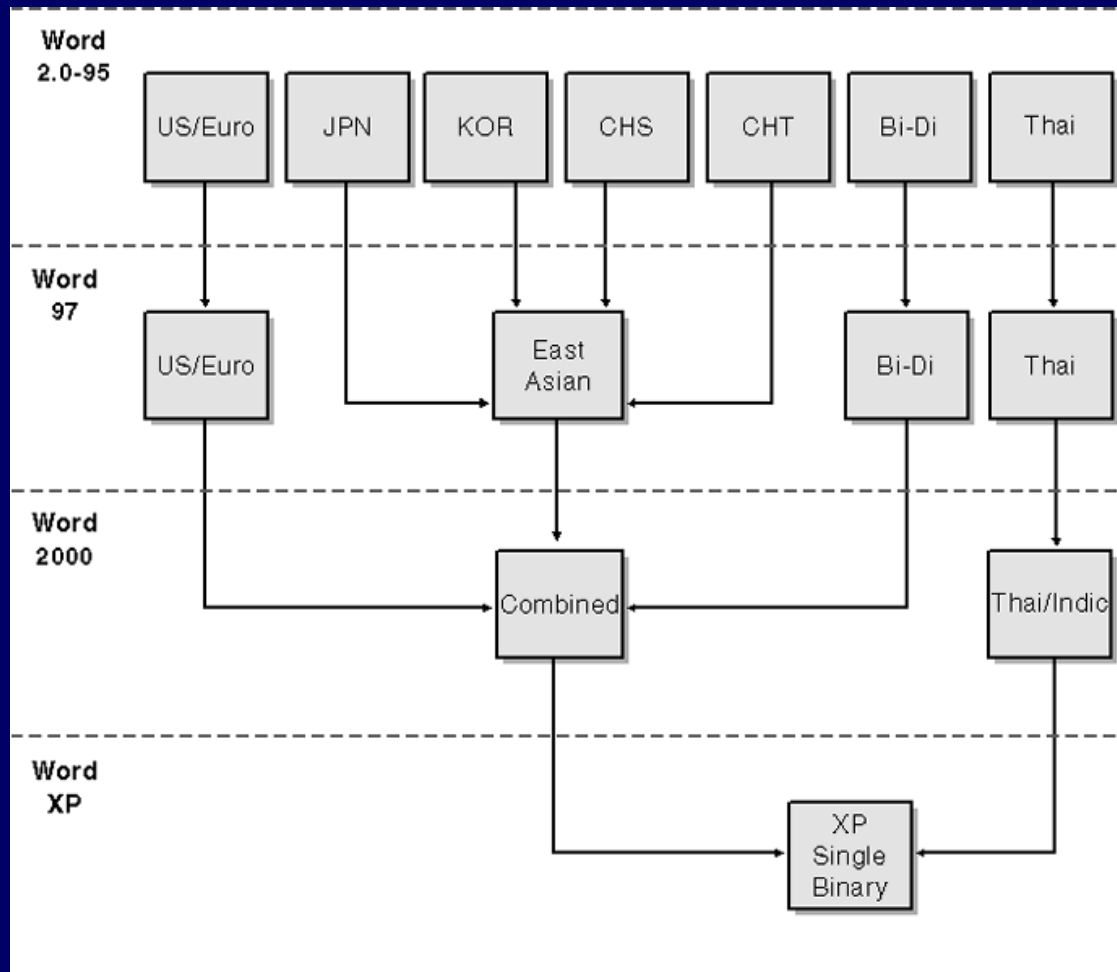
Samples







■ History of Microsoft Word development



.NET Framework

.NET Framework Languages

- The .NET Framework since the 2.0 version is available in 24 languages (also in Polish)
- The .NET Framework Redistributable Package is a single package that supports all languages
 - The English version of the .NET Framework is always installed
- To get localized versions of all the text in the .NET Framework install the .NET Framework Language Packs
 - Many language packs can be installed on a single machine
 - .NET applications will use the resources from the language pack according to the current setting of the `System.Threading.Thread.CurrentThread.CurrentUICulture` property

Encodings in the .NET Framework

- .NET Framework uses UTF-16 to represent characters
 - in some cases it uses UTF-8 internally
- Decoding and encoding using **System.Text** namespace
 - UTF-16 – **UnicodeEncoding** class
 - UTF-8 – **UTF8Encoding** class
 - ASCII – **ASCIIEncoding** class
 - Windows/ISO encodings – **Encoding** class

Cultures

- Cultures are the fundamental building block of internationalization in the .NET Framework
- A culture is a language and, optionally, a region
- It is represented by the `System.Globalization.CultureInfo` class

```
//CultureInfo cultureInfo = new CultureInfo("en");  
CultureInfo cultureInfo = new CultureInfo("en-GB");  
  
string shortDatePattern =  
    cultureInfo.DateTimeFormat.ShortDatePattern;  
  
string currencySymbol =  
    cultureInfo.NumberFormat.CurrencySymbol;
```

CurrentCulture and CurrentUICulture

■ Thread.CurrentCulture

- It represents the default culture for all classes in the **System.Globalization** namespace
- It affects issues such as culture-specific formatting (such as date/time and number/currency formats), parsing, and sorting
- It defaults to the Win32 function **GetUserDefaultLCID**

■ Thread.CurrentUICulture

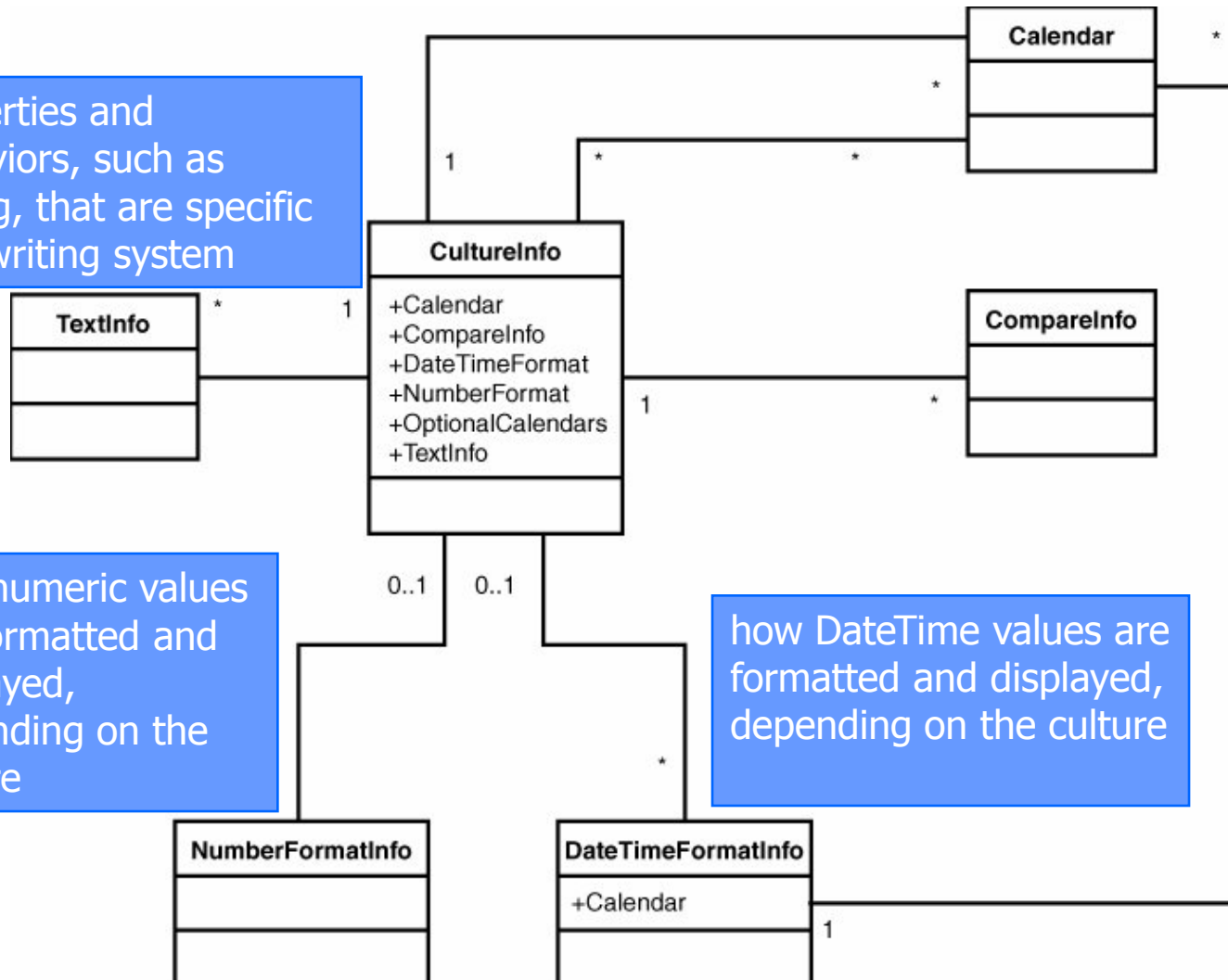
- It represents the default culture used by **ResourceManager** methods
- It affects the retrieval of user interface resources such as strings and bitmaps
- It defaults to the Win32 function **GetUserDefaultUILanguage**

■ Each thread must have its **CurrentCulture** and **CurrentUICulture** properties explicitly and manually set

- They are not inherited from the "parent" thread

CultureInfo

properties and behaviors, such as casing, that are specific to a writing system



how numeric values are formatted and displayed, depending on the culture

how DateTime values are formatted and displayed, depending on the culture

performing basic operations involving days, weeks, months, and years

a set of methods for culture-sensitive string comparisons

String Comparisons

- Possibilities for string comparisons:
 - `==` operator or `String.Equals`
 - they perform a case-sensitive, culture-insensitive comparison
 - `String.CompareTo`, `String.Compare`, `String.CompareOrdinal`
 - a comparison with 0 will always be accurate, regardless of cultural considerations
 - the return results will vary according to the given culture if the strings are not equal



Graphical User Interface Guidelines

User Interface

- The user interface is the part of a computer and its software that people can see, hear, touch, talk to, or otherwise understand or direct
- Input - how a person communicates his or her needs or desires to the computer
 - the keyboard, mouse, trackball, one's finger, one's voice
- Output - how the computer conveys the results of its computations and requirements to the user
 - the display screen, voice and sound

Graphical User Interface

- In a graphical interface, the primary interaction mechanism is a pointing device of some kind
 - this device is the electronic equivalent to the human hand
- WIMP interface: windows, icons, menus, and pointers

Advantages of Graphical Systems

- Symbols recognized faster than text
- Faster learning, easier remembering
- More natural
- Exploits visual/spatial cues
- Increased feeling of control
- Less anxiety concerning use
- More attractive
- May consume less space (icons vs. text)
- Replaces national languages
- Low typing requirements

Disadvantages of Graphical Systems

- Greater design complexity
- Learning still necessary
- Lack of experimentally-derived design guidelines
- Inconsistencies in technique and terminology
- Not always familiar
- Few tested icons exist
- Inefficient for touch typists
- Inefficient for expert users
- The futz and fiddle factor
- Hardware limitations

General Principles of Good GUI Design

Widget's Behaviour and Appearance

- The user must be able to anticipate a widget's behaviour from its visual properties
- Every widget in an application should behave consistently
- If an application requires a new widget that behaves differently from a common widget, give the new one distinctive appearance

Consistency at the Platform Level

- The user must be able to anticipate the behaviour of the program using knowledge gained from other programs
 - applies to: widgets, mouse gestures, accelerator keys, placement of menu, icon and toolbar glyphs
- Maintain consistency with the host platform
 - maintaining consistency with the host platform trumps achieving consistency of the application across platforms

User Warnings and Error Dialogs

- Good GUI interfaces rarely need or use warnings and error dialogs
 - exceptions: hardware problems (e.g. disk failure, lost modem connection), warnings that ask the user's permission to perform an irreversible and potentially erroneous step
- Design program interface to help users enter appropriate data
 - use special widgets (lists of values, masks in edit fields etc.)
- Avoid inappropriate task sequencing by disabling the dependent step until all its dependencies are satisfied

User Feedback

- Feedback at the program level
 - user should know whether a step is in progress or completed
 - design every screen in an application so a novice user can easily tell what steps, especially critical steps, have been performed
- Feedback at the widget level
 - widgets should provide visual feedback (e.g. pushing a button, checking a checkbox etc.)

Environment Safe for Exploration

- Great interfaces invite and reward exploration, and offer the novice both the thrill of discovery and the satisfaction of unassisted accomplishment
- Allow users to undo and redo – they can explore the application without fear of corrupting the data

Self-Evident Applications

- The goal is to create an interface that needs no explanation
 - good applications have comprehensive manuals and online help
 - in great applications novice users rarely need to refer to the manuals or online help

Sounds, Colors and Animations

- Sound, color, animation and multimedia presentations are appropriate for education or entertainment
 - include these components only if they improve users' ability to accomplish tasks
- Follow platform's conventions which describe the appropriate use of sound, color and animation

Users' Work Environment

- Help users customize and preserve their preferred work environment
- Applications designed for multiple users or installed on different computers must address additional issues
 - hardware specific video issues – screen size, video resolution, color depth
 - user-specific video issues – disabilities
- Tailoring the basic interface
 - fonts, colors, sounds, toolbar location and appearance, menu entries
 - predefined schemes are very helpful

Modal Behaviours

- Modal behaviour forces the user to perform tasks in a specific order or otherwise modifies the user's expected responses
 - it restricts more intuitive or natural responses
- Wizards simplify complex tasks but restrict flexibility

User Interface Transparency

- Design the interface so that your users can accomplish their tasks while being minimally aware of the interface itself
 - user's attention is drawn away from the interface and naturally directed at the task itself

References

- Wilbert O. Galitz, The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, Second Edition, 2002
- Microsoft Windows User Experience, 1999
- Everett N. McKay, Developing user interfaces for Microsoft Windows, 1999
- Larry E. Wood, User Interface Design: Bridging the Gap from User Requirements to Design, 1997
- Leslie Cortes, Designing a Graphical User Interface, 1997