Assignment ⇒ (1) SDK (2) Wm-erase (3) Wm-paint

(1) SDK

(Software development kit)

आपण कोणत्याही platform वर किंवा कोणत्या technology वर काम करत असेल तर ते कोणत्या कोणत्या SDK मध्ये करत असतो.

आजकाल काहींचा गैरसमज आहे की SDK हा शब्द फक्त Win32 चा आहे पण हे चुकीचे आहे.

SDK हे genral term असून ती कुठल्याही Hardcoded/specific नाहीये.

SDK is Genral term.

ex. computer is genral term

microprocessor is genral term

intel चा microprocessor हे specific term.

ज्यामध्ये खालील 6 गोष्टी आहे त्या SDK शी Analogous आहे.

(1) compiler, linker, loader / interpreter, assembler disassembler / debugger.
(2) header file / namespaces / packages
(3) API / library /

API < procedural
OO (object oriented)

(4) Documentation

(5) VM / RE

(6) IDE

वरील (6) गोष्टींना SDK बनवते पण (5) & (6) हे optional आहे. पहिले (4) compulsory पाहिजे

1024 | 512 | 448 | 440 byte
— cpu understand

Explanation:-

① Sdk मध्ये पहिला part म्हणजे toolchain (compiler, linker, loader/interpreter, assembler disassembler, debugger)

आपण जेव्हा एखादा program लिहतो तेव्हा आपण एखाद्या language मध्ये तो लिहतो. computer ला direct ती language समजत नाही. मी तो program computer च्या भाषेत convert जो करतो त्यालाच compiler म्हणतात. (i.e. human understandble to machine understandble) (cpu & microprocessor)
प्रत्येक language ला तीन स्वतंत्र compiler असतो
linker : ① machine understandable to os understandble ② link the library function those library which is not written by you.
loader = loader is part of os which is actually execute your program within os.
interpreter = understanding program line by line
assembler = basement मध्ये mostly used
disassembler = C, C++, मध्ये error checking साठी reverse engineering करतात. हे करणारी जो भाषा तो disassembler.

header file | namespace | packages.
C, C++ | C | java

API म्हणजे function
① procedural level
② object oriented.

JVM | RE (virtual machine | runtime environment)
IDE (integrated development tool)

example of SDK — compiler + linker

Win32

1) compiler Name = cl.exe
2) linker Name = link.exe
3) assembler Name = masm (microsoft assembler)
4) disassembler = dasm
5) debugger = windbg
6) documentation : MSDN (microsoft developer network)
7) loader : OS

.Net

1) compiler = cl.exe
2) linker = al.exe
3) loader = operating system
4) assembler = ilasm.exe (intermidiate assembler)
5) disassembler = ildasm.exe
6) debugger = windbg

namespace

1) Object oriented API (FCL = framework class library)
2) DOCS : MSDN
3) IDE : visual studio
4) VM : CLR

java

1) compiler = javac.exe , interpreter = java.exe
2) loader = class loader
3) assembly = bytecode
4) debugger = jdb
5) packages
6) API - OO - (JFC)
7) DOCS : java help
8) JRE & JVM (virtual machine)
9) IDE = Eclipse , Netbeans , Symantec cafe

② WM_CREATE :-

जेव्हा CreateWindow() call होते. तेव्हा जो पहिला message sent होतो तो WM_CREATE असतो.

- this message is sent only once.

* WM_CREATE केव्हा used करतो.

window create होताना काही important गोष्टी initialized कराव्या असतील किंवा काही गोष्टी initialized कराव्या असतील की त्या throughout the program वापरणीय तेव्हा WM_CREATE वापरतो.

- हा message दिसत नाही त्याचा #define macro स्वरूपात windows.h मध्ये आहे (Winuser.h)

#define WM_CREATE 0x0001 (MSDN)

* parameter

wparam = this parameter is not used.

lparam = A pointer to a CREATESTRUCT structure that contains information about the window being created.

ex.

```
LRESULT CALLBACK WndProc(HWND hwnd, UNT iMsg
    WPARAM wParam LPARAM lParam)
{
    switch(iMsg) {
    case: WM_CREATE:
    MessageBox()(hwnd, "WM_CREATE is sent",
                "messages", MB_OK);
        break;
}
```

so basically:

WM_CREATE is there so you can perform one time operations like you might create an edit control here & give it initial text.

③ WM_PAINT

It is message painting either on the screen or on the printer. WM_PAINT message is sent in following eight situations.

① when a cursor of the mouse moves across the client area of the window
② when the whole window is moved to other location on the screen.
③ when icon is dragged & dropped across the client area.
④ when the window is first created & displayed (UpdateWindow()).
⑤ whenever menu is dragged & it overlaps the window.
⑥ whenever client area is over-lapped by another window (fully or partially)
⑦ whenever scrolling operation is done
⑧ ~~whenever menu is dragged & it~~
⑧ Resizing the window.

out of above eight jobs, 1, 2, 3, 4, 5 are done by operating system. you don't have to worry about its logic & code.

* WM_PAINT मध्ये खालील गोष्टी करु नये.
(i) message box दाखवू नये
(ii) memory allocation करु नये.
(iii) function ला call करु नये.

* जेव्हा जेव्हा mouse आपण हालवतो तेव्हा WM_PAINT call होतच म्हणून वरील गोष्टी WM_PAINT मध्ये लिहु नये.

* example:

```
LRESULT CALLBACK WndProc (parameter of callback func)
{
  HDC hdc;
  PAINTSTRUCT PS;
  RECT rc;
  char str[] = TEXT("Hello World");
  switch(imsg)
  {
    case : WM_PAINT
    GetClientRect(hwnd, &rc);
    hdc = BeginPaint(hwnd, &ps);
    // set the text colour.
      setTextColour(hdc, RGB(0, 255, 0));
    // set background colour
      setBkColour[hdc, RGB(0,0,0)];
    // draw the actual text
    DrawText(hdc, str, -1, &rc, DT_SINGLELINE
                              | DT_VCENTER);

    EndPaint(hwnd, &ps);

    break;
  }
```