CDOLLAR Programming Language

BY

Jemin Information Technology

About the Author and Preface

This CDOLLAR is Designed by Analzing many Research papers Using CDOLLAR one can build Datastructures as Fast As could. I Thank God for this wisdom given to me…

----------Wilmix Jemin J,Jemin Information Technology

This EBOOK is Printed in India.

To Make Software Fast like Rabbit movement

and a global redistribution of prosperity

Acknowledgments

We'd like to acknowledge all of the people who played important roles in the creation

of this book. We'd also like to thank all of the developers who've spent time reading this manuscript

and pointing out all of the problems.

Finally, we'd like to extend a sincere thank you to the people who participated in the

CDOLLAR Program. In particular, those who've left feedback in the Author

Online forum have had a strong impact on the quality of the final printed product.

And for providing English translations of the text resources, we'd like to thank

Github and our supporters.

Thanks to all!

---------WILMIX JEMIN J

About this Book

Welcome to CDOLLAR! If you've picked up this book, we suspect you're a C/C++ or JAVA or Dotnet Professional.

working with database who's somehow or other heard about database like sqlserver or oracle.

Perhaps you've worked with the Other Technologies in the past, perhaps you've worked with

another Technologies , or perhaps this is your first step into CDollar P.L.

Whichever path has led you here, you're probably looking for a good introduction

to the new CDOLLAR Programming Language. This book intends to give you that introduction

and much more. If you've never heard of CDOLLAR, we cover the basics in enough

depth to keep you in tow. If you know what CDOLLAR does, but want a deeper understanding

of how it does it, we'll provide that too.

Roadmap

Book is focused on CDOLLAR Programming Language , if you have knowledge or experience about JAVA and C# you can easily focus it.

But Minimum JAVA or C# Technical Knowledge is required to focus on Studying, Designing CDOLLAR Modules .

CDOLLAR is an Advanced Technology focused on Software

Development.

The Brief Contents

Code conventions

The following typographical conventions are used throughout the book:

■ Courier typeface is used in all code listings.

■ Courier typeface is used within text for certain code words.

■ Italics are used for emphasis and to introduce new terms.

■ Code annotations are used in place of inline comments in the code. These highlight important concepts or areas of the code.

Code downloads

This will get you the CDOLLAR.zip file by purchasing it.
a couple of CDOLLAR archive files —as well as some documentation of the source. Instructions on how to install the application are contained in a README file in that download.

Unit-1 : Introduction to CDollar Programming Language

Definition:

CDollar Programming Language is a modern technology consists of JAVA OOPS, Behave like C/C++ OOPS, Networking, RUN and compile at same time,used in Software Development ,Research, and ,Advanced OOPs.

It is mainly used in software field.

It is used in case of Billing,Forms ,Constructing Datastructures , Reports,Security,and,complex problems.

ABOUT CDOLLAR Programming Language

CDollar first name is "OLIVE Technology" which represents OLIVE TREE . Olive Technology is renamed as CDOLLAR.

CDollar v.3 is invented in C/C++ and Java.

But also uses Attractive Syntax .

The CDollar is classified into extension which is

a).cdollar(simillar to java and C++ combination).

b) .C$ (For Intermediate code)

c) CWE-.cdollar to produce Prototype files

e) CDollar Unix => for running the program in unix Os.

f) CDollar with C#

g) CJAVA

So Cdollar is Platform independant Language.

CDollar Program Structure(.cdollar)

Beginning Section : **<CDollar>**

Documentation Section

Package Statement;

<IMPORT><optional>

<USE> Statement;

Interface Statement

LOGIC SECTION

Class Declaration

protected Shared void main(String args[])

{

}

CLOSE LOGIC SECTION

ENDING SECTION : ?>

Explanation:

CDollar Beginning section is <CDollar> ; beginning your CDollar program

CDollar Ending section is ?> ; Ending your CDollar program

Documentation Section means you can include description

with comments.

Package statement means you had to include Cdollar program in Package ....

<USE> statement to import all the packages.

Interface statement for supporting multiple inheritance.

Logic section for writing Cdollar logic with Class followed by main method .

after writing logic close the logic section.

SYNTAX FOR CDOLLAR (.cdollar) (beautiful syntax)

<CDollar>

<IMPORT>

<%

<! CDollar OOPS Logic !>

%>

?>

note: This should be saved in filename.cdollar

CJAVA Programming Syntax:

Beginning Section :<CJAVA>

NAMESPACE SECTION :<PACK><namespace name>

{

CLASS SECTION :<CLASS><classname>

```
  {



     public void main()
     {

< ! Write   CJAVA LOGIC>



     }
  }
```

Ending Section : }

CJAVA SYNTAX:

```
<CJAVA>
<PACK><namespace name>
{


<CLASS><classname>
  {



     public void main()
     {
```

< ! Write CJAVA LOGIC>

```
    }
  }
}
```

CDollar -CWE Programming Structure

Beginning Section :<CDollar>

Import Section :<USE> CUTIL; //optional to load CDollar library packages

NAMESPACE SECTION :<PACK> DTS

LOGIC SECTION :<%

CLASS SECTION : <CLASS> roots

```
 {
    public FLOAT CDollar-MAIN()
    {
```

<H> // use this for code Obsucation

CLOSE LOGIC SECTION : %>

ENDING SECTION :?>

CDollar -CWE SYNTAX

```
<CDollar>
<USE> CUTIL; //optional to load CDollar library packages
<PACK> DTS
<%
<CLASS> roots
  {
    public FLOAT CDollar-MAIN()
    {
<H> // use this for code Obsucation


%>


?>
```

CDOLLAR COMPILER WORKFLOW

How CdollarP1 Technology Works?

———————————————————-

At first .cjava is compiled by Cdollarcc compiler

it produces .exe filename.

How CdollarP2 Technology Works?

_____-

At first .cdollar is compiled by Cdollarc compiler

And it translate to .C$ file with intermediate code that hacker can't understood.

Cdollarv.4 translator uses CDC friend compiler Which compiles

the Cdollar program.

After that Cdollarv.4 translates to .wl class files and use

CDRUN filename.wl automatically to run the Program.

So converting the bytes codes in .wl class file makes the progam

to run faster than other compilers..

==========

CDollarv.4

We know that CDollarc is the compiler, But CDollarv.4 is a translator which translates

your program to .wl files or .Exe files...

This .wl files are futhure use.

CDollarv.4 version is focused on windows...

How to compile cdollar Program in windows?

Cdollarc <filename.cdollar>

How to run cdollar Program in windows?

CDRUN filename

Why CDollar?

CDollar is used for creating libraries ; CDollar is formed in C/C++

in year 2004.

CDollar is modified in java technology in year 2013, 2015,2016.

CDollar is the combination of JAVA , C/C++, and Advanced OOPS.

it will only accept the shortest attractive syntax.

CDollar first name is "OLIVE Technology" which represents OLIVE TREE . Olive Technology

is renamed as CDOLLAR.

Note: a) The Meaning of CDollar is combination of C++ and JAVA OOPS

concepts.

c) CDollar version 1 contains a build compiler .cdollar.

where .cdollar has the features like java,C# , and C/C++.

SYNTAX-1 (used only for CWEB - .cdollar)

<CDollar>

<USE> packages;

<%

<! CDollar-.cdollar OOPS Logic and main functions !>

%>

?>

How CDollar is formed ? What are its Advantages Over Native language JAVA Programming?

CDollar is formed in C++ OOPS concepts..

JAVA borrowed C++ OOPS concepts but

CDollar borrowed C++ OOPS concepts and JAVA oops and it has

Attractive syntax ; Plus in-build functions

for Program and it is responsible for creating

libraries (.wl). JAVA has attained the Programming

standards, But CDollar attains combination of C Technology

and JAVA Technology advantages.

CDollar Generates .wl class files

but JAVA Generated .class files.

Cdollar Has Advanced OOPS than JAVA 1.8.

KEYWORDS

CDollarc Keywords

==========

<CDollar> ?><IMPORT><Finally> UnShared

abstract boolean break byte

case <CATCH> char class const

continue default do double else

enum <--- final finally float

for goto if --> <USE>

instanceof int interface long native

<NEW> package private protected public

return short Shared strictfp <SUPER>

switch synchronized <IS>

throw throws transient <TRY> void volatile while

<% %>

==========

CDollarcc and CJAVA Keywords

==========

<CJAVA><CDOllar>

abstract add as ascending

async await base bool

break by byte case

catch char checked <CLASS>

const continue decimal default

delegate descending do double

dynamic else enum <EQUALS>

explicit extern false finally

fixed float for foreach

from get global goto

group if implicit in

int interface internal into

is join let lock

long <PACK> <NEW> null

object on operator orderby

out override params partial

private protected public readonly

ref remove return sbyte

sealed select set short

sizeof stackalloc Shared string

struct switch this throw

true <TRY> typeof uint

ulong unchecked unsafe ushort

using value var virtual

void volatile where while

yield <% %>

OTHER KEYWORDS IN CDOLLAR

AND -> AND operator

NOT -> NOT operator

-> NOTEQUALS

RUN -> Runnable used in thread

TH-> Thread

<EXE> -> Exception

Friends -> Frend function

OTHER ATTRACTIVE SYMBOLS in CDOLLAR

--> => implements

<-- => extends

DATATYPES

CDollarcc DATATYPES

The eight primitive data types in Java are:

boolean, the type whose values are either true or false.

char, the character type whose values are 16-bit Unicode characters

the arithmetic types:

the integral types:

- byte
- short
- int
- long

the floating-point types:

- float
- double

Values of class type are references. Strings are references to an instance of class String.

Primitive Data Types

There are eight primitive datatypes supported by CDollarc. Primitive datatypes are predefined by the language and named by a keyword. Let us now look into the eight primitive data types in detail.

byte

Byte data type is an 8-bit signed two's complement integer

Minimum value is -128 (-2^7)

Maximum value is 127 (inclusive)(2^7 -1)

Default value is 0

Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an integer.

Example: byte a = 800, byte b = -850

short

Short data type is a 16-bit signed two's complement integer

Minimum value is -32,768 (-2^15)

Maximum value is 32,767 (inclusive) (2^15 -1)

Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an integer

Default value is 0.

Example: short s = 77000, short r = -880000

int

Int data type is a 32-bit signed two's complement integer.

Minimum value is - 2,147,483,648 (-2^31)

Maximum value is 2,147,483,647(inclusive) (2^31 -1)

Integer is generally used as the default data type for integral values unless there is a concern about memory.

The default value is 0

Example: int a = 340000, int b = -7600000

long

Long data type is a 64-bit signed two's complement integer

Minimum value is -9,223,372,036,854,775,808(-2^63)

Maximum value is 9,223,372,036,854,775,807 (inclusive)(2^63 -1)

This type is used when a wider range than int is needed

Default value is 0L

Example: long a = 34990L, long b = -92000000L

float

Float data type is a single-precision 32-bit IEEE 754 floating point

Float is mainly used to save memory in large arrays of floating point numbers

Default value is 0.0f

Float data type is never used for precise values such as currency

Example: float f1 = 54.5f

double

double data type is a double-precision 64-bit IEEE 754 floating point

This data type is generally used as the default data type for decimal values, generally the default choice

Double data type should never be used for precise values such as currency

Default value is 0.0d

Example: double d1 = 15.7

boolean

boolean data type represents one bit of information

There are only two possible values: true and false

This data type is used for simple flags that track true/false conditions

Default value is false

Example: boolean one = true

Char

char data type is a single 16-bit Unicode character

Char data type is used to store any character

Example: char letterA = 'S'

Conditional Operator (? :) in CDollarc

Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide, which value should be assigned to the variable. The operator is written as −

variable x = (expression) ? value if true : value if false

PRIMITIVE DATATYPES in CDOLLARCC

The following table lists the available value types in CDollarcc (v.1)

bool Boolean value True or False False

byte 8-bit unsigned integer 0 to 255 0

char 16-bit Unicode character U +0000 to U +ffff '\0'

decimal 128-bit precise decimal values with 28-29 significant digits (-7.9 x 1028 to 7.9 x 1028) / 100 to 28 0.0M

double	64-bit double-precision floating point type	(+/-)5.0 x 10-324 to (+/-)1.7 x 10308	0.0D
float	32-bit single-precision floating point type	-3.4 x 1038 to + 3.4 x 1038	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

OPERATORS in CDollarcc

Operator Type	Category	Precedence
Unary	postfix	expr++ expr--
	prefix	++expr --expr +expr -expr ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<<>>>>
Relational	comparison	<><= >= instanceof
equality		== NOT=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	AND
	logical OR	OR

Ternary ternary ? :

Assignment assignment = += -= *= /= %= &= ^= |= <<= >>= >>>=

CDollarcc has the following type of operators:

Arithmetic Operators

Relational Operators

Logical Operators

Bitwise Operators

Assignment Operators

Misc Operators

Arithmetic Operators

Example:

Assume variable A holds 1 and variable B holds 7 then:

Operator	Description	Example
+	Adds two operands	A + B = 8
-	Subtracts second operand from the first	A - B = -6
*	Multiplies both operands	A * B = 7
/	Divides numerator by de-numerator	B / A = 7
%	Modulus Operator and remainder of after an integer division	B % A = 0
++	Increment operator increases integer value by one	A++ = 2

-- Decrement operator decreases integer value by one A-- = 0

Relational Operators

Assume variable A holds 30 and variable B holds 10, then:

Show Examples

Operator Description Example

== Checks if the values of two operands are equal or not, if yes then condition becomes true. (A == B) is not true.

!= Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. (A != B) is true.

> Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. (A > B) is true.

< Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. (A < B) is not true.

>= Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. (A >= B) is true.

<= Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. (A <= B) is not true.

Logical Operators

Assume variable A holds Boolean value true and variable B holds Boolean value false, then:

Operator Description Example

&& Called Logical AND operator. If both the operands are non zero then condition becomes true. (A && B) is false.

|| Called Logical OR Operator. If any of the two operands is non zero then condition becomes true. (A || B) is true.

! Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. !(A && B) is true.

Bitwise Operators

Bitwise operator works on bits and perform bit by bit operation. The truth tables for &, |, and ^ are as follows:

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Operator Description Example

& Binary AND Operator copies a bit to the result if it exists in both operands.

| Binary OR Operator copies a bit if it exists in either operand.

^ Binary XOR Operator copies the bit if it is set in one operand but not both.

~ Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.

<< Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.

>> Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

Assignment Operators

There are following assignment operators supported by CDollarcc:

Operator Description Example

= Simple assignment operator, Assigns values from right side operands to left side operand C1 = A1 + B1 assigns value of A1 + B1 into C1

+= Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand C 1+= A1 is equivalent to C 1= C1 + A1

-= Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand C1 -= A1 is equivalent to C 1= C1 - A1

*= Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand C1 *= A1 is equivalent to C1 = C1 * A1

/= Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand C 1/= A1 is equivalent to C1 = C1 / A1

%= Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand C1 %= A1 is equivalent to C1 = C1 % A1

<<= Left shift AND assignment operator C1 <<= 2 is same as C1 = C1 << 2

>>= Right shift AND assignment operator C 1>>= 2 is same as C 1= C1 >> 2

&= Bitwise AND assignment operator C1 &= 2 is same as C 1= C1 & 2

^= bitwise exclusive OR and assignment operator C1 ^= 2 is same as C1 = C1 ^ 2

|= bitwise inclusive OR and assignment operator C1 |= 2 is same as C1 = C1 | 2

Miscellaneous Operators

There are few other important operators including sizeof, typeof and ? : supported by Cdollarcc.

Operator Description Example

sizeof() Returns the size of a data type. sizeof(int), returns 4.

typeof()Returns the type of a class. typeof(StreamReader);

& Returns the address of an variable. ANDa; returns actual address of the variable.

* Pointer to a variable. *a creates pointer named 'a' to a variable.

? : Conditional Expression If Condition is true ? Then value A : Otherwise value B

is Determines whether an object is of a certain type. If(Girafee is animal) // checks if Girafee is an object of the Animal class.

as Cast without raising an exception if the cast fails. Object obj = new StringReader("Wilmix");

StringReader r = obj as StringReader

Operator Precedence in CDollarcc

Operator precedence of the expression.some operators have higher precedence than others; for example, the multiplication or division operator has higher precedence than the addition operator.

For example x = 6 + 12 * 2; here, x is assigned 30, not 36 because operator * has higher precedence than +, so the first evaluation takes place for 12*2 and then 6 is added into it.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators are evaluated first.

Category	**Operator**	**Associativity**
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right

Shift	<<>>	Left to right
Relational	<<= >>=	Left to right
Equality== NOT=		Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	AND	Left to right
Logical OR	OR	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma ,		Left to right

==========

UNIT-2 : CDollar Statements

==========

Cdollar Statements consists of Print statements,

Read Statements , LOOPING Statements

Read Statements

Console.ReadKey();==> Read a vaue from console

Read .sreadln() => Read a String

Read .creadln() => Read only Character

Print Statements

SYNTAX:

CDollar.out.println(String +value);

SYNTAX:

/* USAGE OF PRINTLN */

Print. Println(String s1,int t)

Print. Println(String s1,char t)

Print. Println(String s1,String t)

Print. Println(String s1,double t)

Print. Println(String s1,float t)

Print. printf(String s1,int t)

Print. printf(String s1,char t)

Print. printf(String s1,String t)

Types of Looping Statement

==========

For Loop

========

For Loop operates when the condition met >= or <= or < or >.

At first counter is intialized to a value and it is followed by a condition

and it is followed by increment or decrement operator

A block inside the for loop to be executed if the condition met until false.

SYNTAX:

=======

for (index=intialize value; index <> condition ; incrementor or decrementor)

{

<! BLOCK STATEMENTS !>

}

While Loop

==========

While Loop operates when the condition met >= or <= or < or > or == .which is tested at the TOP of the loop.

A block inside the WHILE loop to be executed if the condition met until false.

SYNTAX:

=======

while (index <> condition)

{

<! BLOCK STATEMENTS !>

}

Do – While Loop

================

Do - While Loop operates when the condition met >= or <= or < or > or == ; which is tested at the bottom of the loop.

A block inside the Do-WHILE loop to be executed if the condition met until false.

SYNTAX:

=======

do

{

<! BLOCK STATEMENTS !>

}

while (index <> condition)

foreach

========

The for-each loop introduced in CDollarc. It is mainly used to traverse array or collection elements.

The advantage of for-each loop is that it eliminates the possibility of bugs and makes the code more readable.

Advantage of for-each loop:

==========

It makes the clear consise of the code.

It eliminates the possibility of programming errors.

SYNTAX:

=======

CDollarc

========

for (index : collections)

{

<! BLOCK STATEMENTS !>

}

CDollarcc

========

for (index in collections)

{

<! BLOCK STATEMENTS !>

}

Types of Conditional Statement

==========

If Statement

==========

If Statement operates when the condition met it will

execute the block inside the if statement.

SYNTAX:

=======

if (condition1 ..... Condition.n)

{

<! BLOCK STATEMENTS>

}

If – Else statement

==========

If -Else Statement operates when the condition met it will

execute the block inside the if statement

or else execute the block inside else statement.

SYNTAX:

=======

if (condition1 ..... Condition.n)

{

<! BLOCK STATEMENTS>

}

else

{

<! BLOCK STATEMENTS>

}

If- Else-if statement

==========

If -Else Statement operates when the condition met it will

execute the block inside the if statement

or else execute the block inside if -else statement followed by a condition.

SYNTAX:

=======

if (condition1 ..... Condition.n)

{

<! BLOCK STATEMENTS>

}

else if (condition1 ..... Condition.n)

{

<! BLOCK STATEMENTS>

}

Switch Statement

==========

Switch statement will test for the eqaulity when there is match with the value of expression.

The Statement inside the default statement is executed last when if none of the above case is satisfied.

if the statement is not followed by break then

another switch statement with equality is executed next.

or else it will skip that statement.

SYNTAX:

=======

switch (variable) {

```
 case v1:
      statements
      break;
 case v2:
      statements
      break;
 case v3:
 case v4:
      statements
 . . .
 default:
      statements
      break;
}
```

Types of Flow Control Statement

==========

Return Statement

==========

Return Statement is used to return a value

when a Function is a return type.

syntax: return value;

Continue Statement

==========

Continue statement is used to continue the loop.

SYNTAX:

=======

continue;

Break Statement

==========

Break statement is used to Skip from the loop.

SYNTAX:

=======

break;

Goto Statement

==========

Goto Statement is used as a climber to goto another block and execute

it.

SYNTAX:

=======

goto label;

Throw Statement

==========

Usually the throw statement is used with try-catch or try-finally statements.

A throw statement can be used in a catch block to re-throw the exception that the catch block caught.

SYNTAX:

=======

throw exception;

ARRAYS

ARRAY is to store a value in a location

which uses stack dts..

SYNTAX for one dimension and multi dimension

==========

<DATATYPE><variablename> Array [dimension1] ... [dimension-n]

eg)

int a11 Array [100];

```
a11[0]=1000;
CDollar.out.println("/n"+a11[0]);
```

O/p

1000

CDollar Pointers

What is Pointers?

Variables that hold memory address are called pointers.

Why we mainly use Pointers?

Pointers reduces the length and complexity of the program,

They increase the execution speed.

It holds the memmory addres..

SYNTAX of CDollar Pointers:

{*} <pointername> Pointers (<VALUE>);

CDollar Functions

Functions are otherwise known as methods or apis can return or not return a value.

Functions are of two types with this cases they are

A) Function with or without return type using or without parameters.

b) CDollar Operator overloading functions

SYNTAX:

=======

In CDollar they are basically declared like this:

<visibility><return type><name>(<parameters>)

{

<function code>

}

CDollar Operator Overloading function:

A function using opertor to perform operations

on a functions with parameters.

eg)

```
public Shared void operator *(int s1 ,int s2)
{
s3=s1 * s2;
CDollar.out.println(""+s3);
}
public Shared void LIB( )
{
operator *(10,10);
// You are passing * Multiply Operator in the main Program
operator *(200,10000);
}
```

Program -1 : CDollar functions with Pointers

```
<CDollar>
<IMPORT>
<%
public class func
{
Shared int a=100;
Shared {*} l1 Pointer(a);
Shared int b=10000;
Shared {*} l2 Pointer(b);

Shared {*} l3 Pointer(0);
 public void CC( ) throws <EXE>
{

int a=0;
                    func.exchange(l1,l2);
CDollar.out.println(""+ l1.get(0)+""+l2.get(0));
}
public Shared void exchange(<OBJECT> a,<OBJECT> b)
{
l3=l1;
```

```
l1=l2;

l2=l3;

}

}

%>

?>
```

==========

UNIT -3: CLASS AND OBJECTS

==========

Coding Standards of CDollar

<CDollar>

<IMPORT>

<USE> package;

<%

%>

?>

Note : <% and %> is used to write class and it's logic.

ALL Program should Start with <CDollar> means starting of a Program and

succeded by <IMPORT>

to load all CDollar packages and ?> Means End of the Program.

but we use <USE> to load the particular packages and we save the memory.

What do you meant by CDollar class and Object?

Class defines a collection of objects,api ,constants,and variables;

that is shared by an object of a class.

What do you meant by abstract class?

It defines the common properties and behaviour of a class.

In which Scenario Abstract and interface is used? WHy?

Interface:

==========

–> If your child classes should all implement a certain group of methods/functionalities but each of the child classes is free

to provide its own implementation then use interfaces.

interface is called as friend in Cdollar.

eg) CLASS C extends A, C

Class C extends A,c can be achieved using interface

which is other wise known as friend f(x) in Cdollar.

This can be written in another form of cdollar as ....

eg)

friend A

{

}

```
friend C

{




}


class c --> A, c
```

Abstract Classes

==========

–> When you have a requirement where your base class should provide default implementation of certain methods

whereas other methods should be open to being overridden by child classes use abstract classes.

–> The purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share.

For example a class library may define an abstract class that is used as a parameter to many of its

functions and require programmers using that library to provide their own implementation of the class

by creating a derived class.

abstract abc

{

```
abstract void display();

}

public class abc1 <--- abc

{

public void display()

{

CDollar.out.println("welcome");

}

}
```

Use an abstract class

==========

When creating a class library which will be widely distributed or reused—especially to clients, use an abstract class in preference to an interface.

Use an abstract class to define a common base class for a family of types.

Use an abstract class to provide default behavior.

Subclass only a base class in a hierarchy to which the class logically belongs.

Use an interface

==========

When creating a standalone project which can be changed at will, use an interface in preference to an abstract class; because, it offers more design flexibility.

Use interfaces to introduce polymorphic behavior without subclassing and to model multiple inheritance—allowing a specific type to support numerous behaviors.

Use an interface to design a polymorphic hierarchy for value types.

Use an interface when an immutable contract is really intended.

A well-designed interface defines a very specific range of functionality. Split up interfaces that contain unrelated functionality.

Program -1: Abc5.cdollar

==========

```
<CDollar>
<%

abstract  class  Abc51

{

  abstract void display();

}

public  class  Abc5  <--- Abc51

{
 Shared void display()
{
CDollar.out.println("Wilmix"+"jemin");
}
```

```
 public void  CDOLLAR-Main( )


{

display();


}

}

%>


?>
```

How to compile it?

CDollarc abc1.cdollar

Output:

======

When you run using CDRUN....

K:\CDollar>CDRUN abc1

you will get a error ....

how you find error?

type out.txt in command prompt

```
K:\CDollar>type  out.txt


Abc5.:30: error: display() in Abc5 cannot override display() in Abc51

 static void display()

           ^

  overriding method is static

1 error
```

Program-2: abcd.cdollar

==========

```
<CDollar>

<%

class   abc

{


void  display(String s)


{


CDollar.out.println("We  learn  C ,  Dotnet  ,and  ,CDollar");

}
```

```
}

class   abcd   <---  abc

{

void  display(String s)

{

<SUPER>(s);

CDollar.out.println("We  learn  C ,  Dotnet  ,and  ,JDollar");

}

}

%>

?>
```

```
K:\CDollar>type out.txt

abcd:30: error: call to super must be first statement in constructor

super(s);

    ^

1 error
```

Program-3: A.cdollar

==========

```
<CDollar>

<%

  class A
{

  public A()  { CDollar.out.println("A's  called"+"n"); }
}

class B
{
public B()  { CDollar.out.println("B's  called"+"n"); }
}

public class C
{
public C()  { { CDollar.out.println("C's  called"+"n"); } }
```

```
public void  CDOLLAR-Main( ) throws <EXE>
{
<NEW> A();
<NEW> B();
<NEW> C();
}
}

%>
?>
```

Input:

=======

```
CDollarc  A.cdollar
```

Output:

=======

```
K:\CDollar>CDRUN C

<table bgcolor=green>
<td>A's  calledn B's  calledn C's  calledn <td>

</table>
A's  calledn B's  calledn C's  calledn
```

Program-4: Duplicates.cdollar

==========

```
<CDollar>
<USE><CJAVA>.util.*;
<%
 public class Duplicates {

  public void  CDOLLAR-Main( ) throws  <EXE>

{

<S><Emp> ts = <NEW> Tree<S><Emp>(<NEW> EmpComp()); //<S> indicates  set  which  will  learn  in
CDollar  collections

//which  remove  duplicates.. now  pass  the  employee  object  in  a s set.

      ts.add(<NEW> Emp(201,"John",40000));
      ts.add(<NEW> Emp(302,"Krish",44500));
```

```
        ts.add(<NEW> Emp(146,"Tom",20000));
        ts.add(<NEW> Emp(543,"Abdul",10000));
        ts.add(<NEW> Emp(12,"Dinesh",50000));
        //adding duplicate entry
        ts.add(<NEW> Emp(146,"Tom",20000));

        ts.add(<NEW> Emp(7777,"777",7777777));
        //check duplicate entry is there or not
        for(Emp e:ts){
            CDollar.out.println(e);
        }
    }
}

class EmpComp --> Comparator<Emp>{

    @Override
    public int compare(Emp e1, Emp e2) {
        if(e1.getEmpId() == e2.getEmpId()){
            return 0;
        } if(e1.getEmpId() < e2.getEmpId()){
            return 1;
        } else {
            return -1;
        }
```

```
    }
}

class Emp {

    private int empId;
    private <Str> empName;
    private int empSal;

    public Emp(int id, <Str> name, int sal){ // <Str>  means   String  in  CDollar
<IS>.empId = id;
<IS>.empName = name;
<IS>.empSal = sal;
    }

    public int getEmpId() {
        return empId;
    }

    public void <S>EmpId(int empId) {
<IS>.empId = empId;
    }

    public <Str> getEmpName() {
        return empName;
```

```
    }

    public void <S>EmpName(<Str> empName) { // <S> means set in cdollar
<IS>.empName = empName;
    }

    public int getEmpSal() {
        return empSal;
    }
    public void <S>EmpSal(int empSal) {
<IS>.empSal = empSal;

//<IS> means this in cdollar
    }

    public <Str> to<Str>(){
        return empId+" : "+empName+" : "+empSal;
    }
}

%>


?>
```

Compile using Duplicates.cdollar

Output:

G:\CDollar>CDRUN Duplicates

<table bgcolor=green>

<td>7777 : 777 : 7777777543 : Abdul : 10000302 : Krish : 44500201 : John : 40000

146 : Tom : 2000012 : Dinesh : 50000<td>

</table>

7777 : 777 : 7777777543 : Abdul : 10000302 : Krish : 44500201 : John : 40000146

: Tom : 2000012 : Dinesh : 50000

Program-5: Geometry.cdollar

==========

<CDollar>

<%

```
class Polygon {
  Shared int width, height;
  public Shared int s_values (int a, int b)
     { width=a; height=b; return(0); }
}
```

```
class Rectangle  <--- Polygon {
  public int area()
     { return width*height; }
}

class Triangle  <--- Polygon {
  public int area()
     { return width*height/2; }
}

class  Geome<TRY>{
 public void  CDOLLAR-Main( ) {
  Rectangle rect = <NEW>  Rectangle();
  Triangle trgl= <NEW> Triangle();

  int t= Polygon.s_values (4,5)  * Polygon.s_values (4,5);
  CDollar.out.println( "Rect area="+rect.area());
  CDollar.out.println( "Triange  Area="+trgl.area());

}
}
%>
```

?>

F1:\CDollar>CDRUN Geometry

<table bgcolor=green>

<td>Rect area=20Triange Area=10<td>

</table>

Rect area=20Triange Area=10

Program-6: student.cdollar

==========

<CDollar>

<%

public class student

{

Shared int sno; Shared int m1,m2,m3;

Shared double avg=0.0;

```
 public void  CDOLLAR-Main( )
        {



student   s  =  <NEW>  student();


sno=1;


m1=234;

m2=456;

m3=656;


avg=((m1=m2+m3)/3);


CDollar.out.println(""+avg);


}

}


%>



?>
```

Output:

<table bgcolor=green>

<td>370.0<td>

</table>

370.0

Program-7: TA.cdollar

======================

<CDollar>

<%

class Person {

// Data members of person

Person(){}

public Person(int x) { CDollar.out.println("Person::Person(int) called"+x); }

}

class Faculty {

public Faculty(int x)

{

```
<NEW> Person(x);
      CDollar.out.println("Faculty::Faculty(int ) called"+x);
   }
}


class Student  {
  // data members of Student
public  Student(int x) {
<NEW> Person(x);
       CDollar.out.println("Student::Student(int ) called"+ x);
   }
}


class TA   {
   TA(int x)  {
<NEW> Faculty(x);
<NEW>  Student(x);


   CDollar.out.println("TA::TA(int ) called"+x);
```

```
   }


 public void  CDOLLAR-Main( )
{
<NEW>  TA(30);



}

}
%>



?>
```

OUTPUT:

```
<table bgcolor=green>
<td>Person::Person(int ) called30Faculty::Faculty(int ) called30Person::Person(i
nt ) called30Student::Student(int ) called30TA::TA(int ) called30<td>

</table>
Person::Person(int ) called30Faculty::Faculty(int ) called30Person::Person(int )
```

called30Student::Student(int) called30TA::TA(int) called30

Program-8: WHILES.cdollar

==========

```
<CDollar>

<%
public class WHILES
{
  public Shared void main(String  args[])  throws  <EXE>
{
int a=0;
while (a <=10)
{
a++;
CDollar.out.println("value="+a);
if ( a==9) continue;
else break;
}
```

}

}

%>

?>

output

value=1

INNER and OUTER CLASS

Inner class are nested inside outer class even if the fields

declared as private members.

<CDollar>

<IMPORT>

<%

class Outer {

private int privInt = 10;

public void createInnerClass() {

Inner inClass = <NEW> Inner(); //creating innerclass object and calling method

access.

inClass.access();

```
}

class Inner { // Inner class

public void access() {

CDollar.out.println("The outer classs privInt is " + privInt);

}

}

}

%>
```

OVERLOADING AND OVERRIDING functions

OVERLOADING

A functions with same name but different signature is called

as Overloading concept.

```
public void display(int i , String j) {}
```

=> If you pass int and string values from main program it will call

this function.

```
ABC a = <NEW> ABC(10,"ewew");

public void display(int i, int j) {}

ABC a = <NEW> ABC(10,20);
```

=> If you pass int and int values it will call this function.

OVERRIDING

A function with same name and same signature

will cause overriding....

Overriding can be avoided by using super() keyword.

in another class.

DATASTRUCTURES in cdollar

Program-1: abc1.cdollar

```
<CDollar>

<%
class LL1
{
private LL1 nextNode = null;
private String datum = null;
public LL1()
{
LL1 list = <NEW> LL1("0 C");
list.add("1 CDOLLAR");
list.add("2 GDOLLAR");
list.add("3 CHDOLLAR");
list.add("4 JDOLLAR");
list.add("5 JSTAR");
list.add("6 JSAUCER");
for (int i = 0; i NOT= list.size(); i = i + 1)
{
```

```
CDollar.out.println(""+list.get(i).StringConvert());
}
}
public LL1(String datum)
{
<SUPER>();
<IS>.datum = datum;
}
public void add(String datum)
{
if (nextNode NOT= null)
{
nextNode.add(datum);
return;
}
nextNode = <NEW> LL1(datum);
}
public String get(int i)
{
if (i == 0)
return datum;
return nextNode.get(i - 1);
}
public int size()
{
```

```
if (nextNode == null)

return 1;

return nextNode.size() + 1;

}

}

class abc1

{

public void  CDOLLAR-Main( )

{

int i;

CDollar.out.println("\nList of Technologies in year "+"2016 ");

LL1 list = <NEW> LL1();

String i1="weew";

CDollar.out.println("wilmix"+i1);

CDollar.out.println(" \njemin"+"is going");

}

}

%>

?>
```

How to compile it?

CDollarc abc1.cdollar

How to run CDollar and see the output stored in .wl file?

K:\CDollar>CDRUN abc1

<table bgcolor=green>

<td>List of Technologies in year 2016 0 C1 CDOLLAR2 GDOLLAR3 CHDOLLAR4 JDOLLAR5

JSTAR6 JSAUCERwilmixweew jeminis going<td>

</table>

List of Technologies in year 2016 0 C1 CDOLLAR2 GDOLLAR3 CHDOLLAR4 JDOLLAR5 JSTA

R6 JSAUCERwilmixweew jeminis going

==========

WHILES.cdollar

<CDollar>

<%

public class WHILES

{

  public Shared void main(String args[]) throws <EXE>

{

int a=0;

```
while (a <=10)

{

a++;

CDollar.out.println("value="+a);

if ( a==9) continue;

else break;

}




}

}


%>


?>
```

output

value=1

OPERATOR OVERLOADING

Operator overloading is an important concept in CDollar. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.

EXAMPLES:

CDOLLAR PROGRAM With OPerator Overloading

```
<CDollar>
<PACK> Area
{
<CLASS> Rectangle
  {
    private double length;   // Length of a Rectangle
    private double breadth;  // Breadth of a Rectangle
    private double height;   // Height of a Rectangle

    public double GETKEYVolume()
    {
      return length * breadth * height;
    }

    public void <SET>Length( double len )
    {
      length = len;
    }

    public void <SET>Breadth( double bre )
```

```
    {
      breadth = bre;
    }

    public void <SET>Height( double hei )
    {
      height = hei;
    }

    // Overload + operator to add two Rectangle objects.
    public Shared Rectangle operator+ (Rectangle b, Rectangle c)
    {
      Rectangle Rectangle <NEW>Rectangle();
      Rectangle.length = b.length + c.length;
      Rectangle.breadth = b.breadth + c.breadth;
      Rectangle.height = b.height + c.height;
      return Rectangle;
    }

  }

<CLASS> Tester
  {
    public FLOAT CDollar-MAIN()
    {
```

```
        Rectangle rectangle1 <NEW>Rectangle();   // Declare rectangle1 of type Rectangle

        Rectangle rectangle2 <NEW>Rectangle();   // Declare rectangle2 of type Rectangle

        Rectangle rectangle3 <NEW>Rectangle();   // Declare rectangle3 of type Rectangle

        double volume = 0.0;    // Store the volume of a Rectangle here


        // Rectangle 1 specification

        rectangle1.<SET>Length(6.0);

        rectangle1.<SET>Breadth(7.0);

        rectangle1.<SET>Height(5.0);


        // Rectangle 2 specification

        rectangle2.<SET>Length(12.0);

        rectangle2.<SET>Breadth(13.0);

        rectangle2.<SET>Height(10.0);
//<SET>  means  set  in  Cdollar
        // volume of Rectangle 1

        volume = rectangle1.GETKEYVolume();

<PRINTLN>("Volume of rectangle1 : {0}", volume);


        // volume of Rectangle 2

        volume = rectangle2.GETKEYVolume();

<PRINTLN>("Volume of rectangle2 : {0}", volume);


        // Add two object as follows:

        rectangle3 = rectangle1 + rectangle2;
```

```
        // volume of Rectangle 3
        volume = rectangle3.GETKEYVolume();


//GETKEY means get in cdollar
<PRINTLN>("Volume of rectangle3 : {0}", volume);


    }
  }
}
```

Advanced Topics in FILE

<BUFFINPUTSTREAM> =>you can use <MARK> and <RESET> keyword

to move backward in a buffered input stream.

<DOUTPUTSTREAM> =>

which can be used to write the stream or to do other operations on the stream.

IT USES <WRITEBYTES> TO Write into file.

<DINPUTSTREAM > => USED TO READ STRING FROM

THE FILE. It Uses <READBYTES> TO READ FROM A FILE.

<BREADER> =>Buffered reader uses <READLINE> to read a line from a

file.

<BWRITER> => Buffered writer uses <WRITELINE> to write a line from a

file.

<INPUTREADER> =>

The InputReader is intended to wrap an InputStream, thereby turning the byte based input stream into a

character based Reader.

<FILE> => used fort creating reading ,Appending, and writing string to a file.

Note: Character files are read and written using <FileWriter> and <FileReader>. Writing Streams of

character is best suited using FileWriter.

FileReader

FileReader assumes that default character encoding and default byte-buffer size are appropriate.

FileReader reads character stream.

<FILESREADER>

FileWriter

FileWriter assumes that default character encoding and default byte-buffer size are appropriate.

<FILESWRITER>

OUTPUTWRITER

<OUTPUTWRITER>

The OutputWriter is intended to wrap an OutputStream, thereby turning the byte based output stream

into a character based Writer.

Better than all this random Access file is the best use....So file concepts are over. SO we ask developers to concenterate onRandomAccess file....

==========

UNIT -4 : CDOLLAR COLLECTIONS

==========

String

String is represented by <Str> notation.

a) <Str><strname> = new <Str> ();

This statement is used to create an object...

b) <Str><strname> = value;

But this Statement will not create an object...

but it stores the value...

the differences between

a) if (s1==s2)

== means it is used to compare the values...

b) if s1.EQ(s2)

EQ means EQUALS is used to compare objects..

CDOLLAR COLLECTIONS

Why we use collections in our software development?

Because for various projects we will use various kinds of

datastructures that's why collections are focused.

Q: What are the Important concepts of Software Development?

ARRAYLIST

SYNTAX:

<AList><Type> arraylistobjectname = new <AList><Type>();

But type may be Object, int, Double,String,etc.

Why we focus Arraylist ?

Since ArrayList involves Powerful insertion and search mechanism when

compared to array.

So we focus it.

Some built in functions available in ArrayList they are add and remove.

syntax : arraylistobjectname.add(<datatype>);

syntax: arraylistobjectname.remove(<datatype>);

How did you iterate the ArrayList?

<WR> syntax means Iterator; this is the shortest syntax of Iterator.

<WR> iteratorname = <CollectionOBJECT>.record();

LinkedList

<LList><Type> arraylistobjectname <NEW><LList><Type>

But type may be Object, int, Double,String,etc.

As according to collection concepts , built in functions are Designed for

LinkedList they are add and remove.

syntax : Linkedlistobjectname.add(<datatype>);

syntax: Linkedlistobjectname.remove(<datatype>);

syntax : Linkedlistobjectname.addFirst(<datatype>);

syntax: Linkedlistobjectname.removeFirst(<datatype>);

syntax : Linkedlistobjectname.addLast(<datatype>);

syntax: Linkedlistobjectname.removeLast(<datatype>);

Actually when you study about Datastructures of LinkedList

and here we Designed the LinkedList using the LinkedList code

as mentioned in above that is LinkedList.c$. And add more functions...

and we use CDollar Generics...

What is the function of LinkedList? Why we use LinkedList?

In ArrayList You can't insert element in to the middle

or first or last so LinkedList is focused....

LinkedList is a Good example of Train....

VECTOR

Vector also has the same Datastructures of ArrayList;

but why we focus? . So vector is simillar to Arraylist.

So we can mention in short notation as VList.

but Vector is synchronized and ArrayList is not Synchronized.

Vector use Enumerator and Iterator but ArrayList use only Iterator.

<VList><VectorObject> = <NEW><VList> ();

<VectorObject>.addE(elements);

but vector used add functions

<VectorObject>.first(); => Represent First Element...

<VectorObject>.last(); => Represent Last Element...

<VectorObject>.removeAll(elements); => It is used to remove all elements..

<VectorObject>.removeAt(elements); => remove at Particular position

<VectorObject>.remove(object); => remove the first occurance of the given element

<VectorObject>.remove(index); => Remove by Index or position.

==========

More about COLLECTIONS

SET

So Set is represented in Cdollar as <S>

Syntax:

<S> Objectname = new <S>();

Difference between Set and List?

List allow duplicates but Set did not allow duplicates...

Set did not allow insertion at middle.

For listing the elements in Ascending or descending order

we had to use TreeSet.

TREESET

Treeset represent a collection that uses Tree datastructure for storage

Items in the collections are stored in Ascending or descending order.

<TS> objectname = new <TS>();

objectname.add(elements);

OTHER COLLECTIONS CONCEPTS

<M> => map MEANS IT CONTAINS KEYS AND VALUE PAIRS...

HashSet

SYNTAX:

<HSET><hashsetname> = <NEW><HSET>();

HASHMAP

SYNTAX:

<HMAP><HASHMAPNAME> = <NEW><HMAP>();

<HASHMAPNAME>.PUT(key,valuepairs);

<HASHMAPNAME>.GETKEY(index);

<HMAP> mp = <NEW><HMAP>();

mp .PUT(1, 234);

CDollar.out.println(""+mp.GETKEY(1));

HASHTABLE

SYNTAX:

<HTABLE><HASHTABLENAME> = <NEW><HTABLE>();

<HASHTABLENAME>.PUT(key,valuepairs);

<HASHTABLENAME>.GETKEY(index);

<HTABLE> mp = <NEW><HTABLE>();

mp.PUT(1, 234);

CDollar.out.println(""+mp.GETKEY(0));

// This elements can be retrieved by using GETKEY().

note: hash determines a order in which elements are

stored in the hash; SO it will display according

to hash stored order.

ADVANCED CONCEPTS of CDOLLAR

GENERIC STACK

<GS> (we will see later)

ITERATOR

Iterator iterate about collection

in the forward direction and not in backward direction.

and it will iterate record wise from the List or collection.

<WR> it = collectionobj.record();

where collection obj may be list,arraylsit, and so on.

for EG)

<WR> it = ar.record(); //iterate arraylist.

while(it.<HAS>) // if it has more elements from arraylist

{

<OBJECT> el = it.<NEXT>;

//<OBJECT> is the object..

//<NEXT> is used to list next element from the collection

CDollar.out.println(" data= "+el);

//print the elements

}

Listiterator:

Normally Iterator will not move backward directions

by using iterator. It can be done by using Listiterator.

<LR><listiterateobject> = Object.<LISTLR>();

eg)

<LR> it1 = ar.<LISTLR>; // iterate the arraylist in backward direction

if you use <PREVIOUS> keyword

while(it1.<HASP>) // if the iterator has more elements

{

<OBJECT> el1 = it1.<PREVIOUS>;

//move to previous record from arraylist

CDollar.out.println(" data1= "+el1);

//print the object

}

ARRAYSSORT

<A>.<SORT>(arrayname);

SO if you want to sort an array you

must use the keyword <A>.<SORT>(ar);

That means the give array is sorted in ascending order and

store it in array

ARRAYBINARYSEARCH

<A>.<BinarySearch>(arrayname,position)

This will search the array in binarysearch wise...

according to the given position.

Exception and ERROR

Exception is a abnormal condition that arise during

the code sequence at run time.

What are the two Types of Exception?

Checked and Unchecked Exception.

Exception that arise during the run time are called as

Unchecked exception.

Thrown exception are refer to the checked exception.

Unchecked Exception

<Arithmetic> -> Arithmetic exception

<ArrayIndex> -> ArrayIndex outof bound exception

<ArrayStore> -> Assignment to an array of an incompactible type.

<IndexOut> -> Index out of boud exception

<NegativeArraySize> -> Array Created with a negative array size.

<NullPointer> -> Null pointer exception

<NumberFormat> -> invalid conversion of string to numeric format.

typing or giving data string as input in integer datatype.

<SecurityException> -> Attempt to violate security

<StringIndex> ->Attempt to acess index ouside the bounds of a string.

Checked Exception

<ClassNotFound> -.> class not found

<CloneNotFound>-> Clone keyword is absent

<IllegalAcess> -> Access to a class is denied.

<Instanation>-> Attempt to create an object of abstract class or friendly functions

<Interrupted>=> One thread has interrupted by another thread.

<NoSuchField> => A request field doesnot exist

<NoSuchMethod>=> Request emthod doesnot exist.

<TRY> -> try in C/JAVA

<CATCH> -> catch in c/java

<Finally> -> final in c/java

SYNTAX:

<TRY>

{

< Executable good statements>

}

<CATCH> (<EXE> e)

{

CDollar.out.println(""+e);

}

<Finally>

{

<Final block statements>

}

Explanation:

When ever the Exception is true statements inside a try

block is executed; otherwise

statements inside a catch block is executed.

Exception occurs or not

final block get executed..

FINAL in CDOLLAR

UnShared keyword means final in CDollar

eg)

UnShared int i=9;

// if a variable is declared as final

that value can't be changed.

eg)

UnShared class abc

{

.....

}

if the class is declared as UnShared it can't

be overridden.

so if the method is declared as UnShared

such method can't be overriden by another class method.

GARBAGE COLLECTION

<RECYCLE> => Garbage colection

eg) if you allocate elements a=9;

but doesnot use in the program so such object

are garbage collected by using <RECYCLE> keyword.

THREADS

A thread is a path of execution that run on CPU

and process is a collection of thread that share a same virtual

memory.

threads.cdollar

<CDollar>

<%

class threads

```
{

   public void  CDOLLAR-Main( )

   {

      My thread1 = <NEW> My("thread1: ");

      My thread2 = <NEW> My("thread2: ");

      thread1.<START>;

      thread2.<START>;

      boolean thread1IsAlive = true;

      boolean thread2IsAlive = true;

      do {

        if (thread1IsAlive AND NOTthread1.isAlive()) {

           thread1IsAlive = false;

          CDollar.out.println("MY   DOG 1 is dead.");
```

```
        }

        if (thread2IsAlive AND NOTthread2.isAlive()) {

            thread2IsAlive = false;

            CDollar.out.println("MY  DOG 2 is dead.");

        }

      } while(thread1IsAlive || thread2IsAlive);

   }

}
```

```
class My <--- TH

{

Shared <Str> message[] ={ "CDollar", "is", "combination", "of", "JAVA", "and   c"};
```

```
   public My(<Str> id)

   {

<SUPER>(id);

   }



   public void <RUN>

   {

      SyncOut.displayList("welcome",message);

   }



   void randomWait()

   {
```

```
<TRY> {

<SLEEP>((long)(3000*Math.random()));

      } <CATCH> (<EXE> x) {

        CDollar.out.println("Interrupted!");

      }

   }

}



class SyncOut

{

public Shared  void displayList(<Str> name,<Str> list[])

{
```

```
for(int i=0;i<list.length;++i) {

My t = (My) TH.currentTH();

t.randomWait();

CDollar.out.println(name+list[i]);

}
}
}
%>

?>
```

OUTPUT:

```
<table bgcolor=green>
<td>welcomeCDollarwelcomeiswelcomeCDollarwelcomecombinationwelcomeiswelcomeofwel
comecombinationwelcomeofwelcomeJAVAwelcomeJAVAwelcomeand  cMY  DOG 2 is dead.we
lcomeand  cMY  DOG 1 is dead.<td>

</table>
welcomeCDollarwelcomeiswelcomeCDollarwelcomecombinationwelcomeiswelcomeofwelcome
```

combinationwelcomeofwelcomeJAVAwelcomeJAVAwelcomeand cMY DOG 2 is dead.welcom

eand cMY DOG 1 is dead.

concat.cdollar

```
<CDollar>
<USE><CJAVA>.io.*;
<USE><CJAVA>.util.*;
<%
// Advanced concepts : Here Friends is a helper function used in other classes
friends toy
{
public void display();
}
// friends will act like friend function in C++.
class concat1 --> toy
//---> indicates implements toy
{
public void display()
{
CDollar.out.println("CDollar is going to be finished");
}
}
public class concat
```

```
{
Shared int counter=4;
//Shared means static and which can be accessed over all the objects of
variables.
//<EXE> means throws Exception
// <S> means set
//<WR> indicates iterator
//<SBD> means string builder
//<SB> is String Buffer
// Differences is StringBuffer is Synchronized and
//and String Builder is not Synchronized
//AND means && in JAVA
//NOT means ! in JAVA
//TH means Thread in CDollar
//int <Arrayname> Array [nooflocations] (ARRAY SYNTAX)
//addE means AddElements
public  Shared void  LIB( ) throws <EXE>
{
int i;
String i1="weew";
Print.Println("wilmix",i1);
Print.Println(" \njemin","is going");
<S><Integer> ar2 = <NEW><TS><Integer> ();
ar2.add(100);
<WR> it = ar2.record();
```

```
while (it.<HAS>)

{

<OBJECT> el= it.next();

Print.Println("/n",el.StringConvert());

}

<SBD> sb = <NEW><SBD>("weldone wilmix");

int a;

a=105;

if (( a >100) AND (a<=106))

CDollar.out.println("/n"+a);

if (a NOT= 0)

CDollar.out.println("/n"+a);

a+=2+counter;

if (a # 107)

CDollar.out.println("/n"+a);

TH t = <NEW> TH();

int a11[] <Array> int[100];

a11[0]=1000;

CDollar.out.println("/n"+a11[0]);

<VList> v = <NEW><VList>();

v.addE(100);

CDollar.out.println("/n"+v.get(0));

v.removeAt(0);

v.addE(1001);

v.addE(1002);
```

```
CDollar.out.println("/n"+v.first());

// This is the program for writing program in random access file ;

//it means it can be randomly inserted and retrieved at any location */

//writing word in random accessfile

<RANDOMACCESSFILE> rf = <NEW><RANDOMACCESSFILE>("concat.c$","rw");

rf.write("Hello World".getBytes());

rf.close();

rf.close();

concat1 obj = <NEW> concat1();

obj.display();

}

}

%>

?>
```

Compile:

```
CDollarc  concat.cdollar
```

So What will be the Output?

```
K:\CDollar>CDRUN concat

<table bgcolor=green>

<td>wilmixweew  jeminis going/n100/n105/n105/n111/n1000/n100/n1001<td>

</table>

wilmixweew  jeminis going/n100/n105/n105/n111/n1000/n100/n1001
```

GEN.cdollar

```
<CDollar>
<%
 public class  GEN<T>

 {
 T t;
 T  display(T  t1)

 {

 t=t1;
 return(t);



 }



 public void  CDOLLAR-Main( )

 {
```

```
 GEN <Integer> i  = <NEW>  GEN<Integer> ();


 CDollar.out.println(""+ i.display(10));
  }


  }

%>


?>
```

```
F1:\CDollar>CDRUN GEN

<table bgcolor=green>

<td>10<td>


</table>

10
```

misc.cdollar

==========

```
<CDollar>


108
```

```
<USE><CJAVA>.util.*;




<%
  abstract class  misc <--- TH

{

<VOLATILE> int v1; //synchronized  happen  at variable level
Shared <TRANS> int v;




~ //destructor in  cdollar
{

CDollar.out.println("object  is    deleted");

}
public  int  getnum1()

{
```

```
return(v1);
}

Shared int s3;
public Shared void  operator *(int s1 ,int s2)
{

s3=s1 * s2;

CDollar.out.println(""+s3);
}

 public void  CDOLLAR-Main( ) throws  <EXE>
{

int  a[]  <Array> int[1000];


 operator *(10,10);//operator overloading

 operator *(200,10000);//operator overloading
```

```
<AList> ar = <NEW><AList>();



for(int i=999;i>=0;i--)
{

a[i]=i;

ar.add(i);
}

<A>.<SORT>(a);



CDollar.out.println(" binary "+<A>.<BinarySearch>(a,5));//perform binary search and element 5 occurs at 5th location



<WR> it = ar.record();
```

```
while(it.<HAS>)


{
<OBJECT> el  = it.<NEXT>;


CDollar.out.println(" data= "+el);




}
<LR> it1 = ar.<LISTLR>;
while(it1.<HASP>)


{
<OBJECT> el1  = it1.<PREVIOUS>;//move previous


CDollar.out.println(" data1= "+el1);
}




<------
while  (es.<HASEMORE>)


{
```

```
<OBJECT> el11 = es.<NEXTEL>;


CDollar.out.println(" data1= "+el1);

}

------>



<DATE> d2 = <NEW><DATE>();



CDollar.out.println("month="+<Month>);

CDollar.out.println("Year="+<Y>);

CDollar.out.println("Hour="+<H>);

CDollar.out.println("Sec="+<SEC>);




v=20;


<FOUTPUTSTREAM> os1 = <NEW><FOUTPUTSTREAM>("out11.txt");

<OOUTPUTSTREAM> d = <NEW><OOUTPUTSTREAM>(os1);


d.<WRITEOBJ>(v);// v is transient so it is saved.

d.<WRITEOBJ>(ar);//ar is not transient so not saved
```

```
<FINPUTSTREAM> osd  =  <NEW><FINPUTSTREAM>("out11.txt");

<OINPUTSTREAM>  br  =  <NEW><OINPUTSTREAM>(osd);


<OBJECT>  ar7  =  br.<OBJECTREAD>;


CDollar.out.println("data="+ar7);



<STACK> s=  <NEW><STACK>();


s.<PUSH>(100);

s.<PUSH>(2000);

s.<POP>;


CDollar.out.println("stackdata="+s);



<PRIORITYQUEUE>  q =  <NEW><PRIORITYQUEUE>();


q.add(1000);


q.add(544);

q.add(66);

q.add(667888);

CDollar.out.println(""+q);
```

```
}

}

%>

?>
```

Output:

1002000000 binary 5 data= 999 data= 998 data= 997 data= 996 data= 995 data= 994

data= 993 data= 992 data= 991 data= 990 data= 989 data= 988 data= 987 data= 986

data= 985 data= 984 data= 983 data= 982 data= 981 data= 980 data= 979 data= 978

data= 977 data= 976 data= 975 data= 974 data= 973 data= 972 data= 971 data= 970

data= 969 data= 968 data= 967 data= 966 data= 965 data= 964 data= 963 data= 962

data= 961 data= 960 data= 959 data= 958 data= 957 data= 956 data= 955 data= 954

data= 953 data= 952 data= 951 data= 950 data= 949 data= 948 data= 947 data= 946

data= 945 data= 944 data= 943 data= 942 data= 941 data= 940 data= 939 data= 938

data= 937 data= 936 data= 935 data= 934 data= 933 data= 932 data= 931 data= 930

data= 929 data= 928 data= 927 data= 926 data= 925 data= 924 data= 923 data= 922

data= 921 data= 920 data= 919 data= 918 data= 917 data= 916 data= 915 data= 914

data= 913 data= 912 data= 911 data= 910 data= 909 data= 908 data= 907 data= 906

data= 905 data= 904 data= 903 data= 902 data= 901 data= 900 data= 899 data= 898

data= 897 data= 896 data= 895 data= 894 data= 893 data= 892 data= 891 data= 890

data= 889 data= 888 data= 887 data= 886 data= 885 data= 884 data= 883 data= 882

data= 881 data= 880 data= 879 data= 878 data= 877 data= 876 data= 875 data= 874

data= 873 data= 872 data= 871 data= 870 data= 869 data= 868 data= 867 data= 866

data= 865 data= 864 data= 863 data= 862 data= 861 data= 860 data= 859 data= 858

data= 857 data= 856 data= 855 data= 854 data= 853 data= 852 data= 851 data= 850

data= 849 data= 848 data= 847 data= 846 data= 845 data= 844 data= 843 data= 842

data= 841 data= 840 data= 839 data= 838 data= 837 data= 836 data= 835 data= 834

data= 833 data= 832 data= 831 data= 830 data= 829 data= 828 data= 827 data= 826

data= 825 data= 824 data= 823 data= 822 data= 821 data= 820 data= 819 data= 818

data= 817 data= 816 data= 815 data= 814 data= 813 data= 812 data= 811 data= 810

data= 809 data= 808 data= 807 data= 806 data= 805 data= 804 data= 803 data= 802

data= 801 data= 800 data= 799 data= 798 data= 797 data= 796 data= 795 data= 794

data= 793 data= 792 data= 791 data= 790 data= 789 data= 788 data= 787 data= 786

data= 785 data= 784 data= 783 data= 782 data= 781 data= 780 data= 779 data= 778

data= 777 data= 776 data= 775 data= 774 data= 773 data= 772 data= 771 data= 770

data= 769 data= 768 data= 767 data= 766 data= 765 data= 764 data= 763 data= 762

data= 761 data= 760 data= 759 data= 758 data= 757 data= 756 data= 755 data= 754

data= 753 data= 752 data= 751 data= 750 data= 749 data= 748 data= 747 data= 746

data= 745 data= 744 data= 743 data= 742 data= 741 data= 740 data= 739 data= 738

data= 737 data= 736 data= 735 data= 734 data= 733 data= 732 data= 731 data= 730

data= 729 data= 728 data= 727 data= 726 data= 725 data= 724 data= 723 data= 722

data= 721 data= 720 data= 719 data= 718 data= 717 data= 716 data= 715 data= 714

data= 713 data= 712 data= 711 data= 710 data= 709 data= 708 data= 707 data= 706

data= 705 data= 704 data= 703 data= 702 data= 701 data= 700 data= 699 data= 698

data= 697 data= 696 data= 695 data= 694 data= 693 data= 692 data= 691 data= 690

data= 689 data= 688 data= 687 data= 686 data= 685 data= 684 data= 683 data= 682

data= 681 data= 680 data= 679 data= 678 data= 677 data= 676 data= 675 data= 674

data= 673 data= 672 data= 671 data= 670 data= 669 data= 668 data= 667 data= 666

data= 665 data= 664 data= 663 data= 662 data= 661 data= 660 data= 659 data= 658

data= 657 data= 656 data= 655 data= 654 data= 653 data= 652 data= 651 data= 650

data= 649 data= 648 data= 647 data= 646 data= 645 data= 644 data= 643 data= 642

data= 641 data= 640 data= 639 data= 638 data= 637 data= 636 data= 635 data= 634

data= 633 data= 632 data= 631 data= 630 data= 629 data= 628 data= 627 data= 626

data= 625 data= 624 data= 623 data= 622 data= 621 data= 620 data= 619 data= 618

data= 617 data= 616 data= 615 data= 614 data= 613 data= 612 data= 611 data= 610

data= 609 data= 608 data= 607 data= 606 data= 605 data= 604 data= 603 data= 602

data= 601 data= 600 data= 599 data= 598 data= 597 data= 596 data= 595 data= 594

data= 593 data= 592 data= 591 data= 590 data= 589 data= 588 data= 587 data= 586

data= 585 data= 584 data= 583 data= 582 data= 581 data= 580 data= 579 data= 578

data= 577 data= 576 data= 575 data= 574 data= 573 data= 572 data= 571 data= 570

data= 569 data= 568 data= 567 data= 566 data= 565 data= 564 data= 563 data= 562

data= 561 data= 560 data= 559 data= 558 data= 557 data= 556 data= 555 data= 554

data= 553 data= 552 data= 551 data= 550 data= 549 data= 548 data= 547 data= 546

data= 545 data= 544 data= 543 data= 542 data= 541 data= 540 data= 539 data= 538

data= 537 data= 536 data= 535 data= 534 data= 533 data= 532 data= 531 data= 530

data= 529 data= 528 data= 527 data= 526 data= 525 data= 524 data= 523 data= 522

data= 521 data= 520 data= 519 data= 518 data= 517 data= 516 data= 515 data= 514

data= 513 data= 512 data= 511 data= 510 data= 509 data= 508 data= 507 data= 506

data= 505 data= 504 data= 503 data= 502 data= 501 data= 500 data= 499 data= 498

data= 497 data= 496 data= 495 data= 494 data= 493 data= 492 data= 491 data= 490

data= 489 data= 488 data= 487 data= 486 data= 485 data= 484 data= 483 data= 482

data= 481 data= 480 data= 479 data= 478 data= 477 data= 476 data= 475 data= 474

data= 473 data= 472 data= 471 data= 470 data= 469 data= 468 data= 467 data= 466

data= 465 data= 464 data= 463 data= 462 data= 461 data= 460 data= 459 data= 458

data= 457 data= 456 data= 455 data= 454 data= 453 data= 452 data= 451 data= 450

data= 449 data= 448 data= 447 data= 446 data= 445 data= 444 data= 443 data= 442

data= 441 data= 440 data= 439 data= 438 data= 437 data= 436 data= 435 data= 434

data= 433 data= 432 data= 431 data= 430 data= 429 data= 428 data= 427 data= 426

data= 425 data= 424 data= 423 data= 422 data= 421 data= 420 data= 419 data= 418

data= 417 data= 416 data= 415 data= 414 data= 413 data= 412 data= 411 data= 410

data= 409 data= 408 data= 407 data= 406 data= 405 data= 404 data= 403 data= 402

data= 401 data= 400 data= 399 data= 398 data= 397 data= 396 data= 395 data= 394

data= 393 data= 392 data= 391 data= 390 data= 389 data= 388 data= 387 data= 386

data= 385 data= 384 data= 383 data= 382 data= 381 data= 380 data= 379 data= 378

data= 377 data= 376 data= 375 data= 374 data= 373 data= 372 data= 371 data= 370

data= 369 data= 368 data= 367 data= 366 data= 365 data= 364 data= 363 data= 362

data= 361 data= 360 data= 359 data= 358 data= 357 data= 356 data= 355 data= 354

data= 353 data= 352 data= 351 data= 350 data= 349 data= 348 data= 347 data= 346

data= 345 data= 344 data= 343 data= 342 data= 341 data= 340 data= 339 data= 338

data= 337 data= 336 data= 335 data= 334 data= 333 data= 332 data= 331 data= 330

data= 329 data= 328 data= 327 data= 326 data= 325 data= 324 data= 323 data= 322

data= 321 data= 320 data= 319 data= 318 data= 317 data= 316 data= 315 data= 314

data= 313 data= 312 data= 311 data= 310 data= 309 data= 308 data= 307 data= 306

data= 305 data= 304 data= 303 data= 302 data= 301 data= 300 data= 299 data= 298

data= 297 data= 296 data= 295 data= 294 data= 293 data= 292 data= 291 data= 290

data= 289 data= 288 data= 287 data= 286 data= 285 data= 284 data= 283 data= 282

data= 281 data= 280 data= 279 data= 278 data= 277 data= 276 data= 275 data= 274

data= 273 data= 272 data= 271 data= 270 data= 269 data= 268 data= 267 data= 266

data= 265 data= 264 data= 263 data= 262 data= 261 data= 260 data= 259 data= 258

data= 257 data= 256 data= 255 data= 254 data= 253 data= 252 data= 251 data= 250

data= 249 data= 248 data= 247 data= 246 data= 245 data= 244 data= 243 data= 242

data= 241 data= 240 data= 239 data= 238 data= 237 data= 236 data= 235 data= 234

data= 233 data= 232 data= 231 data= 230 data= 229 data= 228 data= 227 data= 226

data= 225 data= 224 data= 223 data= 222 data= 221 data= 220 data= 219 data= 218

data= 217 data= 216 data= 215 data= 214 data= 213 data= 212 data= 211 data= 210

data= 209 data= 208 data= 207 data= 206 data= 205 data= 204 data= 203 data= 202

data= 201 data= 200 data= 199 data= 198 data= 197 data= 196 data= 195 data= 194

data= 193 data= 192 data= 191 data= 190 data= 189 data= 188 data= 187 data= 186

data= 185 data= 184 data= 183 data= 182 data= 181 data= 180 data= 179 data= 178

data= 177 data= 176 data= 175 data= 174 data= 173 data= 172 data= 171 data= 170

data= 169 data= 168 data= 167 data= 166 data= 165 data= 164 data= 163 data= 162

data= 161 data= 160 data= 159 data= 158 data= 157 data= 156 data= 155 data= 154

data= 153 data= 152 data= 151 data= 150 data= 149 data= 148 data= 147 data= 146

data= 145 data= 144 data= 143 data= 142 data= 141 data= 140 data= 139 data= 138

data= 137 data= 136 data= 135 data= 134 data= 133 data= 132 data= 131 data= 130

data= 129 data= 128 data= 127 data= 126 data= 125 data= 124 data= 123 data= 122

data= 121 data= 120 data= 119 data= 118 data= 117 data= 116 data= 115 data= 114

data= 113 data= 112 data= 111 data= 110 data= 109 data= 108 data= 107 data= 106

data= 105 data= 104 data= 103 data= 102 data= 101 data= 100 data= 99 data= 98 da

ta= 97 data= 96 data= 95 data= 94 data= 93 data= 92 data= 91 data= 90 data= 89 d

ata= 88 data= 87 data= 86 data= 85 data= 84 data= 83 data= 82 data= 81 data= 80

data= 79 data= 78 data= 77 data= 76 data= 75 data= 74 data= 73 data= 72 data= 71

data= 70 data= 69 data= 68 data= 67 data= 66 data= 65 data= 64 data= 63 data= 6

2 data= 61 data= 60 data= 59 data= 58 data= 57 data= 56 data= 55 data= 54 data=

53 data= 52 data= 51 data= 50 data= 49 data= 48 data= 47 data= 46 data= 45 data=

44 data= 43 data= 42 data= 41 data= 40 data= 39 data= 38 data= 37 data= 36 data

= 35 data= 34 data= 33 data= 32 data= 31 data= 30 data= 29 data= 28 data= 27 dat

a= 26 data= 25 data= 24 data= 23 data= 22 data= 21 data= 20 data= 19 data= 18 da

ta= 17 data= 16 data= 15 data= 14 data= 13 data= 12 data= 11 data= 10 data= 9 da

ta= 8 data= 7 data= 6 data= 5 data= 4 data= 3 data= 2 data= 1 data= 0month=2Year

=1Hour=10Sec=13data=20stackdata=[100][66, 1000, 544, 667888]

Program1.cdollar

==========

<CDollar>

<USE><CJAVA>.util.*;

<%

class Program1

{

public void CDOLLAR-Main( )

{

int i;

CDollar.out.println("\nList of Technologies in year "+"2016 ");

```
String i1="weew";

CDollar.out.println("wilmix"+i1);


CDollar.out.println("Hiram is going"+"today");

CDollar.out.println("Hiram  Age  is ="+"45 ");


CDollar.out.println("Hiram    is working "+"in  Abc Bank\n");

CDollar.out.println("  \njemin"+"is going");


<TS>   ar2 =  <NEW><TS>();


ar2.add("100");

ar2.add("22");


CDollar.out.println("jeminjhjhjh"+ar2.StringConvert());


CDollar.out.println("no:  2/782 ,ds street,california-2322"+ar2.StringConvert());









}
```

```
}

%>

?>
```

output:

List of Technologies in year 2016 wilmixweewHiram is goingtodayHiram Age is =45 Hiram is working in Abc Bank jeminis goingjeminjhjhjh[100, 22]no: 2 /782 ,ds street,california-2322[100, 22]

SYNCHRONIZED:

If multiple clients want to access the shared resource

synchronization provide the way for the multiple clients

not for the specific one.

BIG Program for Synchronization

```
<CDollar>

<%

 class threads

{

   public void  CDOLLAR-Main( )

   {

      My thread1 = <NEW> My("thread1: ");

      My thread2 = <NEW> My("thread2: ");

      thread1.<START>;

      thread2.<START>;

      boolean thread1IsAlive = true;

      boolean thread2IsAlive = true;
```

```
      do {

        if (thread1IsAlive AND NOTthread1.isAlive()) {

           thread1IsAlive = false;

          CDollar.out.println("MY   DOG 1 is dead.");

        }

        if (thread2IsAlive AND NOTthread2.isAlive()) {

           thread2IsAlive = false;

           CDollar.out.println("MY  DOG 2 is dead.");

        }

      } while(thread1IsAlive || thread2IsAlive);

   }

}
```

```
class My <--- TH

{

Shared <Str> message[] ={ "CDollar", "is", "combination", "of", "JAVA", "and   c"};



   public My(<Str> id)

   {

<SUPER>(id);

   }



   public void <RUN>

   {

      SyncOut.displayList("welcome",message);
```

```
    }



    void randomWait()

    {

<TRY> {

<SLEEP>((long)(3000*Math.random()));

        } <CATCH> (<EXE> x) {

          CDollar.out.println("Interrupted!");

        }

    }

}
```

```
class SyncOut

{

public Shared synchronized void  displayList(<Str> name,<Str> list[])

{

for(int i=0;i<list.length;++i) {

My t = (My) TH.currentTH();

t.randomWait();

CDollar.out.println(name+list[i]);

}

}

}
```

%>

?>

Output:

welcomeCDollarwelcomeiswelcomecombinationwelcomeofwelcomeJAVAwelcomeand cMY DOG 1 is dead.welcomeCDollarwelcomeiswelcomecombinationwelcomeofwelcomeJAVAwelcomeand cMY DOG 2 is dead.

WAIT

It WAITS indefinitely for another thread of execution until it receives notify or notify all message.

<WAIT>

NOTIFY AND NOTIFYALL

<NOTIFY> ->

The keyword process waits for a single thread waiting on a object monitor.

ALL ->

The keyword process waits for a multiple thread waiting on a object monitor.

JOIN

---

join() method

This Process join with another thread after another thread finishes the

execution.

eg) Thread t1 => Wait for 100 seconds

and Thread t2 => will execute after Thread t1 complets the execution.

This is the meaning of join process in Thread.

CDOLLAR ADVANCED CONCEPTS

STACK

Stack means lastin first out.

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO)

principle.

In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item

out of the stack.

<STACK> s= <NEW><STACK>();

s.<PUSH>(100); // PUSH operator to push the elements into stack

s.<PUSH>(2000);

s.<POP>;

CDollar.out.println(""+s);

Output:

[100]

Priority Queue:

Many applications require that we process items having keys in order, but not necessarily in full sorted order and not necessarily all at once.

Often, we collect a set of items, then process the one with the largest key, then perhaps collect more items, then process the one with the current largest key,

and so forth. An appropriate data type in such an environment supports two operations:

remove the maximum and insert. Such a data type is called a priority queue.

```
<PRIORITYQUEUE> q = <NEW><PRIORITYQUEUE>();
q.add(1000);
q.add(544);
q.add(66);
q.add(667888);
CDollar.out.println(""+q);
```

Output:

[66, 1000, 544, 667888]

Destructor:

Destructor means object is going to be destroyed.

```
~
{
CDollar.out.println("object is deleted");
// this means object is going to be destroyed.
}
```

where ~ is the Destructor

GENERICS

GENRICS means which is used to Pass Type as argument as class

for example if you want to pass String , int, float datatypes at the

same time and if you use display method to display the value of any

datatype

so Generic is most useful in that case.

```
<CDollar>
<%
public class GEN<T>
{
T t;
T display(T t1)
{
t=t1;
return(t);
}
public  Shared void  LIB( )
{
GEN <Integer> i = <NEW> GEN<Integer> ();
CDollar.out.println(""+ i.display(10));
}
}
```

```
%>

?>
```

==========

UNIT-5 : CDollar Advanced Collections

==========

BUCKET

Bucket are used to store key,value data, and Generated Random number

where datatype may be string ,object ,etc.

SYNTAX:

Bucket<DATATYPE> list = <NEW> Bucket<DATATYPE>(<DATATYPEVALUE>);

list.KeyAdd(<DATATYPEVALUE>);

list.add(<DATATYPEVALUE>);

list.RandomAdd();

list.Display(list);

Advantages

Using Bucket you can also Retrieve the values stored n position.

Searching and Insertion is fast than other DTS.

Random Indexing is possible.

eg) If you store a duplicate value such Random key will be different.

It also used to add many values.

EXTEND

Extend class is used in CDollar since to provide multiple inheritence

about 100000000 classes . Extends class also list values in methods and

constructor values.

Extend means a Bucket contains List of class and it is also

Behave like Bucket. So it is also one of the Advanced concepts in CDollar.

SYNTAX:

EXTEND <<DATATYPE>> list11 = <NEW> EXTEND <<DATATYPE>> (STRING);

list.KeyAdd(<DATATYPEVALUE>);

list.add(<DATATYPEVALUE>);

list.RandomAdd();

list.Display(list);

Advantages:

It is also used to add many values

Indexing is possible

Value can also be list by index and behave like bucket.

It list only the class value and object value.

It is stateless.

PIPE:

PIPE is used to maintain stateful state.

It is used for DataFlow in a Program. We can also add the values,

Constructor values of one class and other class and display it.

It also list the values from the Bucket.

SYNTAX:

Pipe <<DATATYPE>> list11 = <NEW> Pipe <<DATATYPE>> (STRING);

list.KeyAdd(<DATATYPEVALUE>);

list.add(<DATATYPEVALUE>);

list.RandomAdd();

list.Display(list);

Why we Prefer CDollar for software Field?

Used in BILLS, Forms ,Reports,Charts, any software project , GRAPHICS to web etc.

BUCKET

Bucket are used to store key,value data, and Generated Random number

where datatype may be string ,object ,etc.

SYNTAX:

Bucket<DATATYPE> list = <NEW> Bucket<DATATYPE>(<DATATYPEVALUE>);

list.KeyAdd(<DATATYPEVALUE>);

list.add(<DATATYPEVALUE>);

list.RandomAdd();

list.Display(list);

Advantages

Using Bucket you can also Retrieve the values stored n position.

Searching and Insertion is fast than other DTS.

Random Indexing is possible.

eg) If you store a duplicate value such Random key will be different.

It also used to add many values.

EXTEND

Extend class is used in CDollar since to provide multiple inheritence

about 100000000 classes . Extends class also list values in methods and

constructor values.

Extend means a Bucket contains List of class and it is also

Behave like Bucket. So it is also one of the Advanced concepts in CDollar.

SYNTAX:

EXTEND <<DATATYPE>> list11 = <NEW> EXTEND <<DATATYPE>> (STRING);

list.KeyAdd(<DATATYPEVALUE>);

list.add(<DATATYPEVALUE>);

list.RandomAdd();

list.Display(list);

Advantages:

It is also used to add many values

Indexing is possible

Value can also be list by index and behave like bucket.

It list only the class value and object value.

It is stateless.

PIPE:

PIPE is used to maintain stateful state.

It is used for DataFlow in a Program. We can also add the values,

Constructor values of one class and other class and display it.

It also list the values from the Bucket.

SYNTAX:

Pipe <<DATATYPE>> list11 = <NEW> Pipe <<DATATYPE>> (STRING);

list.KeyAdd(<DATATYPEVALUE>);

list.add(<DATATYPEVALUE>);

list.RandomAdd();

list.Display(list);

Why we Prefer CDollar for software Field?

Used in BILLS, Forms ,Reports,Charts, any software project , GRAPHICS to web etc.

CDollar ADvantages over JAVA and other Programming Languages

A) CDollar is the combination of JAVA , C/C++, and Advanced OOPS.

b) CDollar will only accept the shortest attractive syntax.

c) CDollar also used for construction of any datastructures.

d) CDollar helps the developers to provide inheritance by not using extends

keyword

and call the class in main program when use in linux.

e) CDOLLAR Solves diamond Problem with multiple Inheritance when used in linux.

f) It also supports friendly function, pointers , and structures.

g) CDollar support Virtual memmory and garbage collection.

h) It is efficient, fast and easy to understand, and it is a OOPS Technology.

i) CDollar is a High level language.

j) CDollar is highly portable language

k) Using CDollar you can create any datastructures as libraries and

use it in your Application program.

l) CDollar language is a structured and object programming language.

m) CDollar has OOPS concepts like JAVA.

n) CDollar have the concept of Packages,etc.

o) CDollar have the concept of constructor or destructor and had magic oops concepts.

p) It Support functions with Default Arguments

q) It Supports Exception handling

r) It Support Generic Programming

s) It have pointer and Nodes..

t) CDollar is much simpler oops concepts, which leads to faster development and less mental

overhead.

u) CDollar is almost always explicitly compiled

w) CDollar is easy to learn. CDollar was designed to be easy to use and is therefore easy to write,

compile, debug, and learn than other programming languages.

CDollar is object-oriented. This allows you to create modular programs and reusable code.

CDollar is platform-independent.

x) CDollar creates .wl and .exe or .dll files and it can be used with CDollar main program (CWE EDitor) to create a complete software.

y) CDollar will compile and run at same time where other technology can't

do

z) CDollar is mainly used in complex programming , Billing the

goods,Graphics,etc

AA) CDollar is platform independant language

BB) CDollar is an interactive Technology.

Disadvantages of CDollar Technology

---

a) CDollar doesn't concenterate mostly

on GUI but mostly on invention of new

datastructures,OOPS, Advanced OOPS..

b) CDollar doesnot focused on

cloud computing...

Note: SO CDOLLAR is a medium

programming language in IT and WRIT sector which

is mainly focused on security, datastructures,

,OOPS, Advanced OOPS in software development field only.

FAQS

A) A C Programmer or any oops developer can easily

study it....

Note:

CDollarv.2 ,CDollarv.1 ,CDollarv.3 is not a Version. Cdollarv.3 is a improved compiler.

CDollarv.3 ,CDC is a compiler and CDollarv.4 is a Translator and translate to

.wl files and

CDRUN is responsible for running the CDollar Program.

CDOLLAR MAIN Program Syntax AND ADVANCED CONCEPTS PROGRAM.

(.cdollar-CWE)

Syntax:

```
<CDollar>

<PACK><nAMESPACE>
<%
<CLASS><CLASSNAME>
  {
     public FLOAT CDollar-MAIN()
     {

<! CDOLLAR LOGIC!>



%>
```

?>

BAG

=====

Bag is the extension of LinkedHashmap and it is the fastest

datastructures than Dictionary.

SYNTAX:

=======

Bag object = new Bag();

object .put(key,value);

Functions

getValues(key) => it is used to get the values for a particular key

get(key,loc) => it is used to get the value stored at a loc (indexing

purpose)

boolean containsValue(object Value) => To check the value present in bag or

not.

put(key,value) => it is used to add key and value in Bag

remove(key ,value) => It is used to remove key and value.

TreeList

========

TreeList simillar to Bucket but store items in tree format.

TreeList list = new TreeList ("BUCKETS");

list.KeyAdd(KEY);

list.add(VALUE1);

list.RandomAdd(RANDOMNO);

list.DisplayO(list,0);

MASK

====

It is the extension of Tree Structure and it can store many values

using mask object and we can also retrieve the values stored in mask.

Mask m = new Mask(<DATATYPE>);

m.add(multiple values);

m.getR(Loc); => Get the values stored in right position

m.getL(LOC) => Get the values stored in left position

HEAP:

====

Creates a tree , puts the data into tree in a fairly balanced way and displays

the tree's size and data in a tree by performing an inorder traversal.

Heap hob = new Heap(<datatype>);

hob.add(datum);

hob = new Heap(key,value1,value2);

Bucktist

==========

Bucktist is simillar to Bucket but it is used to addd two values with one

key.

Bucktist l = null;

l= new Bucktist(key,value1,value2);

WICKET

=======

Wicket is used to store multiple values using same object with

4 values per key.

Syntax:

Wicket list12;

list12=new Wicket(key,v1,v2,v3,v4);

list12.Display();

list12.Display(list12,location);

EXAMPLE -1: BAG

<CDollar>

<PACK> MyP

<%

<CLASS> Programs

{

public FLOAT CDollar-MAIN()

{

Bag b <NEW> Bag();

```
b.PUT(1,34);

b.PUT(2,444);


<PRINTLN>(""+b);









%>


?>
```

EXAMPLE:2 : CDOLARARRAYS

==========

```
<CDollar>


<USE><CDOLLARS>.util;


<PACK> MyP

{

<CLASS> Programs

   {
```

```
public FLOAT CDollar-MAIN()
{



<AList>  ar <NEW><AList> ();
 for  (int  i=0;i<=100000;i++)
 ar.add(i);




<CDOLLARARRAYS> list1 <NEW><CDOLLARARRAYS>("ANIMALS ");
      list1.add("1 horse");
 list1.add("2 pig");
 list1.add("3 cow");
 list1.add("4 goat");
 list1.add("5 chicken");

 list1.add("6 ostrich");
 list1.add(ar.StringConvert());
 list1.Display();
```

```
    %>

?>
```

EXAMPLE-3: CREATE AN BOOTLOADER Using CDOLLAR

```
<CDollar>

<PACK> MYOS
{
<CLASS> MYOs
  {
public FLOAT CDollar-MAIN(){


<PRINTLN>("HelloWorld for booting MYOS");



    %>

?>
```

EXAMPE-4: POINTERS

```
<CDollar>
<PACK> MyP
{
<CLASS> Programs
   {
      public FLOAT CDollar-MAIN()
      {


<Str>  s="dsdds";

 {*} l Pointers (s);

l.add(s);

 for (int i = 0; i NOT= l.size(); i = i + 1)
 {

<OBJECT> obj=l.GETKEY(i);
```

```
<PRINTLN>(obj);




 }









    %>


?>
```

EXAMPLE-5: DICTIONARY

```
<CDollar>
<USE> System.Collections.Generic;

<PACK> MyP
{
<CLASS> Programs
   {
```

```
    public FLOAT CDollar-MAIN()
    {



 Dictionary<string, string> openWith  <NEW>  Dictionary<string, string>();
openWith.Add("txt", "notepad.exe");
openWith.Add("bmp", "paint.exe");
openWith.Add("dib", "paint.exe");
openWith.Add("rtf", "wordpad.exe");




   %>

?>
```

Example-6: EXTEND

```
<CDollar>

<IMPORT>



<PACK> MyP

<%
<CLASS> Programs
<%
      public FLOAT CDollar-MAIN()

      {


 EXTEND list <NEW> EXTEND("BUCKETS");



    list.KeyAdd("1101");
```

```
            list.add("jemin");

            list.RandomAdd();

            list.Display(list);

<PRINTLN>(""+list.DisplayO(list,1));

%>

?>
```

EXAMPLE-7: HEAP

```
<CDollar>

<PACK> MyP
```

```
{
<CLASS> Programs
   {
       public FLOAT CDollar-MAIN()
      {



 Heap root <NEW> Heap("wilmix");


 for (int i = 0; i <= 10; i = i + 1)
 {
 root.add("item " + i);
 }


<PRINTLN>(root.size() );
 root.printTree();




    %>


?>
```

Example-8: LArray

```
<CDollar>
<USE><CDOLLARS>.util;

<PACK> MyP
{
<CLASS> Programs
  {
      public FLOAT CDollar-MAIN()
     {

 LArray root <NEW> LArray("root");

<AList>  ar <NEW><AList>();


 for  (int  i=0;i<=1000;i++)

 ar.add(i);

 root.add("wilmix");
 root.add("jemin");
 root.add("shalom");
```

```
root.add("1010");

root.add("101");

root.add("201");

root.add(ar.StringConvert());

root.add("100000000");

//print the tree's size and contents


root.printTree();









   %>


?>
```

Example-9 : PIPE

```
<CDollar>

<PACK> MyP

{
<CLASS> Programs
   {
public FLOAT CDollar-MAIN()
      {


Pipe list  <NEW> Pipe("BUCKETS");



    list.KeyAdd("1101");

                list.add("jemin");

                list.RandomAdd();
```

```
                list.Display(list);

<PRINTLN>(""+list.DisplayO(list,1));

    %>

?>
```

EXAMPLE-10: TREELIST

```
<CDollar>

<PACK> MyP

{

<CLASS> Programs

   {
```

```
public FLOAT CDollar-MAIN()
{



TreeList list  <NEW> TreeList ("BUCKETS");



    list.KeyAdd("1101");

            list.add("jemin");


            list.RandomAdd("1111");

TreeList list2 <NEW> TreeList("BUCKETS");
 list2.KeyAdd("1102");

            list2.add("rahul");
```

```
            list2.RandomAdd("1112");




<PRINTLN>("DATA="+list.DisplayO(list,0));



<PRINTLN>("DATA="+list2.DisplayO(list2,0));



   %>

?>
```

Example-11 : MASK

```
<CDollar>

<USE><CDOLLARS>.util;



<PACK> My

{

<CLASS> Programs

  {

      public FLOAT CDollar-MAIN()

    {



 MASK root <NEW> MASK("wilmix");


 for (int i = 0; i NOT= 10; i = i + 1)

 {

 root.add("item " + i);

 }
```

```
root <NEW> MASK("root1",1211211,54441);


root  <NEW> MASK("root2",121121,5444);

root  <NEW> MASK("root5",99121888,"5");


root  <NEW> MASK("root3",12112,544);


root  <NEW> MASK("root4",1211,54);


root  <NEW> MASK("root51",121,5);



root.printTree();









   %>


?>
```

Example-12 : WICKET

```
<CDollar>


<PACK> MyPo
{
<CLASS> Programs
  {
       public FLOAT CDollar-MAIN()
     {
 Wicket  list12;
 list12 <NEW> Wicket(1000,10002,43433,4343,5555451);
 list12 <NEW> Wicket(10001,100021,434331,4343,5555452);
 list12 <NEW> Wicket(10002,100022,434332,4343,5555453);
 list12 <NEW> Wicket(10003,100023,434333,4343,5555454);
 list12 <NEW> Wicket(10004,100024,434334,4343,5555455);
 list12 <NEW> Wicket(10005,100025,434335,4343,5555456);


 list12.Display(list12);
```

```
<PRINTLN>("DATA="+list12.DisplayO(list12,0));



    %>

?>
```

Example-13 : STRUCTURE

```
<CDollar>
<PACK> MyPoi
{
<CLASS> Programs
   {
        public FLOAT CDollar-MAIN()

       {


<Str>  s="dsdds";

 {*} l Pointers(s);
```

```
l.add(s);


 for (int i = 0; i NOT= l.size(); i = i + 1)

 {


<OBJECT> obj=l.GETKEY(i);

<PRINTLN>(obj);




 }




<STRUCTURE>  list <NEW><STRUCTURE> (l.GETKEY(0));


 for (int i11 = 0; i11 NOT= list.size(); i11 = i11 + 1)

 {




<OBJECT>  el=list.ret(i11);
```

```
<PRINTLN>("SNO= "+el);

            }


   %>

?>
```

Example-14 : BUCKETIST

```
<CDollar>


<PACK> MyP

{

<CLASS> Programs

   {
```

```
        public FLOAT CDollar-MAIN()
      {




Bucketist bp   <NEW> Bucketist("wilmix");


bp   <NEW> Bucketist(1,222,434);
bp   <NEW> Bucketist(1,222,434);



bp.Display(bp);



<PRINTLN>("DATA="+bp.DisplayO(bp,1));



   %>


?>
```

==========

UNIT-6 :CDOLLAR GRAPHICS and NETWORKING

==========

CDOLLAR GRAPHICS always focus on GWT Graphics and GUI. GWT

is a heavy weighted toolkit.

CDollar Graphics can be done by extending Graphics class.

This will create a frame for that.

GWT graphics can be drawn using <PAINT> method.

Program-1

```
<CDollar>
<USE><CJAVA>.awt.*;
<%
  class  abcde <--- Graphics

{
 public void  CDOLLAR-Main( )
{
abr  a  = <NEW> abr();
```

```
a.<SIZE>(500,700);

    a.<FLOWLAYOUT>
a.<VISIBLE=TRUE>



}



Shared  class abr  <--- abcde
{
<PAINT>

{
//Color  c  =  <NEW> Color();

for(int  i=0; i<=600;i++)
{

<DRAWTEXT>("CHOOSEN  CHOICE  is",400,500);
<SETCOLOUR>(COLOR=red);
<OVAL>(10+i,10+i,50+i,50+i);
```

```
<FILLOVAL>(70+i,90+i,140+i,100+i);

<SETCOLOUR>(COLOR=blue);

<OVAL>(190+i,10+i,90+i,30+i);

<RECT>(100+i,10+i,60+i,50+i);

<SETCOLOUR>(COLOR=cyan);

<FILLRECT>(100+i,10+i,60+i,50+i);

<ROUNDRECT>(190+i,10+i,60+i,50+i,15+i,15+i);

<SETCOLOUR>(COLOR=green);

<ARC>(10+i,20+i,150+i,190+i,160+i,60+i);

<FILLARC>(230+i,15+i,150+i,200+i,150+i,75+i);


if ( i== 300)  i=i-1;

}




}


}

}

%>



?>
```

Program-2:

Draw a house using CDollar

HOUSE.cdollar

```
<CDollar>



<USE><CJAVA>.awt.*;




<%
  class  HOUSE  <--- <GRAPHICS>

{

public void  CDOLLAR-Main( )
{

houseparts a  = <NEW> houseparts();

 a.<SIZE>(500,700);
```

```
    a.<FLOWLAYOUT>
 a.<VISIBLE=TRUE>


}




Shared class houseparts  <--- HOUSE
{
<PAINT>
   {
      background(g);
      house (g);
      roof1 (g);
      roof2 (g);
      windows (g);
      framing (g);
      extras (g);
      text (g);
   }

   public void background(<DRAW> g)
   {
```

```
<SETCOLOUR>(COLOR=black);




<FILLOVAL> (15,35,170,55);

<FILLOVAL> (20,20,160,50);

<FILLOVAL> (350,50,170,55);

<FILLOVAL> (355,35,160,50);




<SETCOLOUR>(COLOR=cyan);

<FILLOVAL> (650,035,120,120);

<SETCOLOUR>(COLOR=green);

<ARC>(10,20,180,190,160,60);

<FILLARC>(230,15,150,200,150,75);
   }


   public void house (<DRAW> g)
   {
<SETCOLOUR>(COLOR=yellow);

<FILLRECT> (100,250,400,200);

<FILLRECT> (499,320,200,130);

<SETCOLOUR>(COLOR=green);
```

```
<FILLRECT> (160,150,60,90);

<FILLRECT> (245,380,110,70);

<FILLRECT> (508,350,180,100);

<SETCOLOUR>(COLOR=yellow);

<FILLOVAL> (282,412,10,10);

<FILLOVAL> (307,412,10,10);


   }


   public void roof1 (<DRAW> g)

   {

<SETCOLOUR>(COLOR=pink);

      int x[] = {98,300,501};

      int y[] = {250,130,250};

<FILLPOLYGON>(x,y,3);

   }


   public void roof2 (<DRAW> g)

   {

<SETCOLOUR>(COLOR=orange);

      int x[] = {499,499,700};

      int y[] = {320,249,320};

<FILLPOLYGON>(x,y,3);

   }
```

```
   public void windows (<DRAW> g)
   {
<SETCOLOUR>(COLOR=white);
<FILLOVAL>(521,350,68,31);
<FILLOVAL> (606,350,68,31);
<FILLRECT> (121,261,78,78);
<FILLRECT> (121,361,78,78);
<FILLRECT> (401,261,78,78);
<FILLRECT> (401,361,78,78);
<FILLRECT> (241,261,118,78);
<SETCOLOUR>(COLOR=white);
<FILLRECT> (125,265,70,70);
<FILLRECT> (125,365,70,70);
<FILLRECT>(405,265,70,70);
<FILLRECT> (405,365,70,70);
<FILLRECT> (245,265,110,70);
<FILLOVAL> (525,353,60,25);
<FILLOVAL> (610,353,60,25);
   }

   public void framing (<DRAW> g)
   {
<SETCOLOUR>(COLOR=black);
```

```
<FILLRECT> (298,380,2,70);

<FILLRECT> (508,382,180,2);

<FILLRECT> (508,417,180,2);

<SETCOLOUR>(COLOR=white);

<FILLRECT> (157,265,5,70);

<FILLRECT> (157,365,5,70);

<FILLRECT> (437,265,5,70);

<FILLRECT> (438,365,5,70);

<FILLRECT> (297,265,5,70);

<FILLRECT> (125,298,70,5);

<FILLRECT> (125,398,70,5);

<FILLRECT> (405,298,70,5);

<FILLRECT> (405,398,70,5);

<FILLRECT> (245,298,110,5);

<FILLRECT> (245,375,110,5);

<FILLRECT> (240,375,5,75);

<FILLRECT> (352,375,5,75);

<FILLRECT> (508,345,180,5);

<FILLRECT> (503,345,5,105);

<FILLRECT> (688,345,5,105);
   }


   public void extras (<DRAW> g)

   {

<SETCOLOUR>(COLOR=orange);
```

```
<FILLOVAL> (160,105,35,45);

<FILLOVAL> (170,95,35,45);

<FILLOVAL> (160,85,35,45);

<FILLOVAL> (170,35,35,45);

<FILLOVAL> (160,25,35,45);

<FILLOVAL> (170,15,35,45);

<SETCOLOUR>(COLOR=orange);

<FILLRECT> (508,450,180,150);

<FILLRECT> (245,450,107,50);

<FILLRECT> (274,500,50,40);

<FILLRECT> (274,520,250,45);

      }


   public void text (<DRAW> g)

   {

<SETCOLOUR>(COLOR=orange);

<DRAWTEXT>("House portrait by: wilmix jemin",390,70);

   }


   }




}
```

%>

?>

PROGRAM-3

---

if you select radio or listbox or combobox it should display items

in text box.

//Graphics using GWT and GWT components are heavy weighted

<CDollar>

<USE><CJAVA>.awt.*;

<%

class GUI1 <--- <GRAPHICS> --><HEAR>

{

```
Shared <RADIO> r <GWT=6>();

Shared <TEXTFIELD> l3 <GWT=3> ();

Shared <CHECKBOX> l5 <GWT=5> ("YES",false,r);

 Shared  <CHECKBOX> l51 <GWT=5> ("NO",false,r);


Shared  <LISTBOX> lb <GWT=7>();

 Shared  <COMBOBOX> cb <GWT=8>();


 Shared  <Str>  s= "";

<ITEMSTATECHANGED>

{


if  (ie.<ITEMSELECTABLE> == l5)


l3.<VALUE>("YES");



if  (ie.<ITEMSELECTABLE> == l51)


l3.<VALUE>("NO");



if  (ie.<ITEMSELECTABLE> == cb)

l3.<VALUE>(((<COMBOBOX>) ie.<ITEMSELECTABLE>).<SELECTITEM>);
```

```
if  (ie.<ITEMSELECTABLE> == lb)

l3.<VALUE>(((<LISTBOX>) ie.<ITEMSELECTABLE>).<SELECTITEM>);

s=l3.<ASSIGN>();

}

  public void  CDOLLAR-Main( ) {

 abrpaint g  =<NEW>  abrpaint();

<IMAGE>
    //GUI1  g  = <NEW> GUI1();
<LABEL> l1 <GWT=1> ("CDollar  GUI  Programming");
<BUTTON> l2 <GWT=2> ("CDollar  GUI  Programming");
//<TEXTFIELD> l3 <GWT=3> ();
<TEXTAREA> l4 <GWT=4> (12,40);
//<CHECKBOX> l5 <GWT=5> ("Yes");

l5.<SOUND>(g);
//<CHECKBOX> l51 <GWT=5> ("NO");
```

```
l51.<SOUND>(g);


      l3.<VALUE>("<THIS> is a textbox");

      l4.<APPEND>("Number of columns in this textarea: " +   l4.<COLS>);


      //the add() method of the Frame class is

      //used to add components to the frame

       g.add(l1);

      g.add(l2);

      g.add(l3);

       g.add(l4);

   g.add(l5);

    g.add(l51);


lb.add("CDOLLAR");

lb.add("JAVA");

lb.add("JDOLLAR");

lb.add("C");

lb.add("MAC");

 g.add(lb);lb.<SOUND>(g);


cb.add("CDOLLAR");

cb.add("JAVA");

cb.add("JDOLLAR");

cb.add("C");
```

```
cb.add("MAC");


   g.add(cb);cb.<SOUND>(g);




    g.<SIZE>(500,700);
    g.<FLOWLAYOUT>
     g.<VISIBLE=TRUE>


 }


 Shared  class abrpaint  <--- GUI1
{
<PAINT>

{

<DRAWTEXT>("CHOOSEN  CHOICE  is"+s,400,500);

<RECT>(20,10,100,60);

}
```

```
}

}

%>

?>
```

CDOLLAR Networking

N/w are essential to our life. Intenet is born due to networking and

A method of Client -server communications

gives like a house - to house interaction.

CLIENT SERVER PROGRAM

<CDollar>

```
<PACK>  MYOS

{

<CLASS> MYOs

   {

public FLOAT CDollar-MAIN(){



<CLIENT>("localhost","1099");


<SERVER>("1099");









    %>


?>
```

==========

UNIT:7 : CDollar Security,CDOLLAR with WNOSQL DB

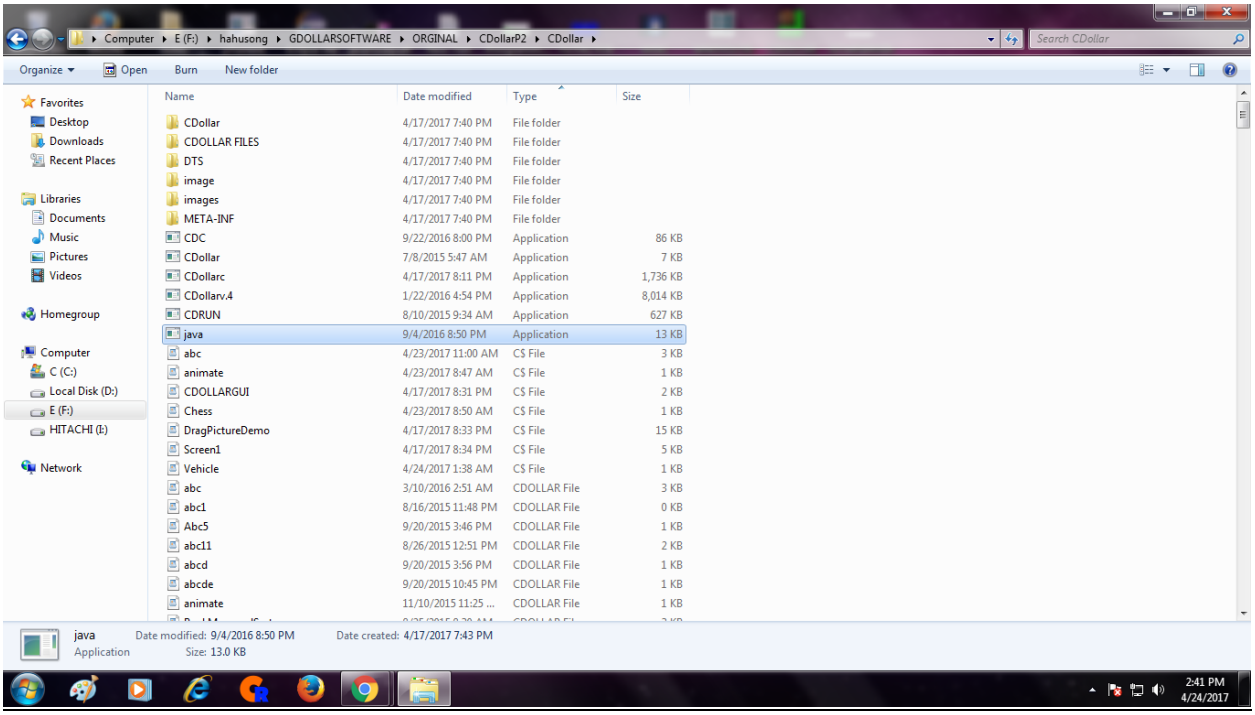
==========

CDOLLAR with WNOSQL

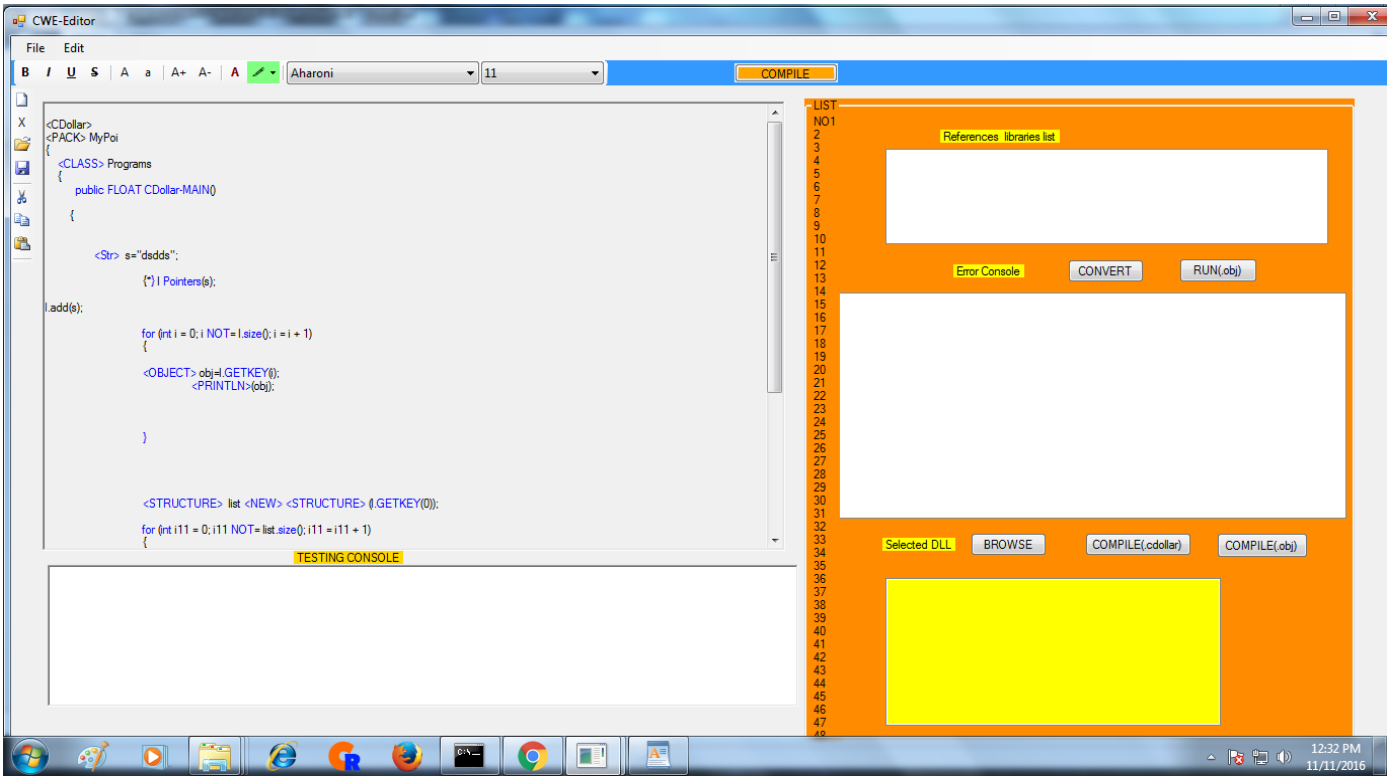
```
<CDollar>
<USE> CDollar.WDBA; //use   cdolla.wdba packages
<USE><CDOLLARS>.util; /use cdollar.util packages
<USE><CDOLLARS>.lang;
<USE> WDBA;
<PACK> WDBAexample
{
   public <CLASS> example1
   {

      Shared void Main()

      {
string g = WDBASQL.WDBASQLS("datastores", "USEDATABASE", "dbpwds",
"C:\\Programs\\CDOLLAR\\WNOSQLProgramfiles\\WNO");
//  declare   directory  of  .wdba files



         string t = WDBASQL.WDBASQLS("dbuser", "dbpwds", 1, "wilmix78", "wilmix78", 1, 5, g);

 // supply username  and  password

         string s1 = "SelectAll from columns 0 to 20 , 1 to 1 ?= XXX By X f(x) : {0,1,2}: {3,4,5} :{2,4}";
//selectall  columns  from 0 to  20 for  row  and  cols   1,1  respectively.

         string s11 = "RIGHTJOIN from student 0 to 1 , 1 to 4 ?= emp For X f(x) : {0,1,2,3,4,5,6,7,8,9,10,11} :
{0,1,2,3,4,5,6,7,8,9,10,11} : {0}";
//perform   right  join  between  query  student  and  emp.
```

```
<PRINTLN>(""+SQL.WDBAQUERY( s1, t));
//print selectall from columns table
<PRINTLN>("" + SQL.WDBAQUERY(s11, t));
//print right join query.
      }
   }
}
```

a)SCREENSHOTS OF CDOLLAR Project Structure



b) CDOLLAR _CWE EDITOR



==========

CJAVA

<CJAVA>

<USE> package;
<PACK> package
{

<CLASS> classname
{

public void main()
{

<! code logic !>

```
    }
  }
}
```

Example-1
========

At first add TreeExample.dll to properties file

and compile the program using CDollarcc Tree.cjava

```
<CJAVA>

<USE> Tree;
<PACK> Tree
{

<CLASS> Tree
  {


    public void main()
    {

TreeExample.call();// call the api of CDollar.


    }
  }
}
```

CDOLLAR with JSTAR

For CDollar with JSTAR carry .exe and .dll and

put it in JSTAR server for futhure use.

==========

UNIT -8: CDOLLAR MOCK EXERCISES

==========

CDOLLAR MOCK EXERCISES

(1 100 = 100 marks)*

A) Develop a Accounting software with Credit/Debit /Discount on the goods sold and name of the Bank be ABC Ltd ,
use it in text format.(10)

B) Develop a Electricity bill using CDollar

Advanced OOPS(10) in Console.

C) Briefly Describe C$ Program work flow(5)

d) Write a Cdollar program to perform a animation.(5)

D) Briefly Describes Cdollar Advantages and Disadvantages(5)

E) Develop a School Management system Project

1) to Insert students 2) Delete unwanted students
3) Update student details 4) List student details acc to rollno
5) Sort the student names with details in Ascending order
*use WNOSQL Db. (1 *15 =15 marks)*

f) ***CDollar Mini project***

In a Atm Transcation

i) Write a C$ program for Atm Transcation form using Cdollar Graphics.

ii) USE CDollar with WNOSQL DB

iii) Perform Transcations like Debit

1000rs or you wish.

If the A/c is low popup message is

displayed that a/c is low.

iv) Prepare a Report Based on C$ Transcations

G) Prepare a CHESS BOARD. (10)

==========

CDOLLAR Reference Documentation

CDollar Latest Tutorial will be updated in the given below Websites.....

CDollar TUTORIALS

kindly go thru the given below urls for CDollar Tutorial..

CDollar Part1 Tutorial url :
https://sites.google.com/site/gdollarprogrammingtutorial/gdollar-cdollartutorial

CDollar Advanced DTS :
https://sites.google.com/site/gdollarprogrammingtutorial/gdollaradvanceddts

CDollar Part2 Tutorial : :
https://sites.google.com/site/gdollarprogrammingtutorial/cdollarpart2