

2006

Sun

01

Jan

For Week # 001-364

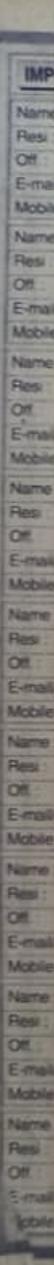
SUN	TUE	WED	THU	FRI	SAT	SUN
26	27	28	29	30	31	1

FEB

JAVA

Evolving of Programming language.

1. Machine language: Here the programmer had to write the instructions in 0 or 1. Since the computer only understand the 0 and 1. Since the computer is an electronic device, it can only understand the binary bit 0 and 1.
difficulties :- Writing instructions in 0 or 1 was very difficult and the program will have every operation carried out by the machine label operations. Carried out by the instructions should be mentioned by the programmer explicitly.
2. Assembly language:- In Assembly language some symbols are specified. These symbols represents a specific operation by system. For example ADD, SUB, MUL etc are used for Adding, Subtracting, Multiplying the oprands. But the difficulties in Assembly language is, the programmer has to remember all the symbols to write a specific program. And the programmer has a knowledge about the internal operation of the system. Here the programmer has to consider the memory Mngr., for efficiently use of memory, so it is very difficult for a simple user to write Assembly programming. Debugging is very difficult in Assembly Programming.



2006, Mon
02 Jan 2nd

3-High level language : - The High level languages are developed to reduce the difficulties in low level language. Here the user can write the program in English like language. Once it is very easy to learn. The 1st category of high level languages are FORTRAN, COBOL, BASIC etc. These programming languages are developed for specific purposes. For example FORTRAN was extensively used in Scientific purposes and it was not very good for system codes. While BASIC is easy to learn but was not very powerful and it was not well structured. This programming language were depend upon Goto Statement for Program Control so it was very difficult to understand the program which uses extensively Goto statement. So these since these languages was in specific uses so they uses a limited data types and limited functions.

4. 'C' Language:- Dennis Ritchie invented a powerful language known as C language. C programming language has language can eliminated the problems of previous language. It was powerful efficient, structured language. It has a full set of data types, and system defined functions are embedded in a language so that it has wide range of applications. It was very easy to learn and written in English like language.

PROBLEMS OR DRAWBACKS: Though C is a rich set of data types and consist of wide range of functions, and also a structured, its prime focus was on functions not data. It was a procedural-Oriented Programming language. In C, all the data are declared as public or global so there was no security. It has no technique to reuse the code, hence the size of the program increases as complexity increases. i.e. it was very difficult to write real-time program by C.

4- C++ Programming language:- The C++ programming language was developed, now which can meet every requirement of real time programming. It was developed on top-down approach. i.e. In C++, maximum focus will given to data not the function. It also provide security to data, and reusable feature to programmer. It also inherit all the features of C. So it was known as Successor of C language and also called as Object Oriented Programming language. So By using the different features of C++ a programme can write complex program with minimum no. of codes.

5- Java Programming language: Java was developed on 1992-1995 which introduce on 1995 World Wide Web. It was going to introduce in Market. At that time the Developers need a Software which can run in any

2006

Wed

04

Jan

2nd Week • 004-361

JAN	S	M	T	W	T	F	S	S	M	T	W	F
	1	2	3	4	5	6	7	8	9	10	11	12
	13	14	15	16	17	18	19	20	21	22	23	24

platform, or any environment, i.e. Every electronic device has a microchip which has its own CPU. It needs a software to operate. The developer develops a Java, through which they write a software code which is a platform independent and can run in any platform or any environment. Also in internet, the data we transported from a location to another. So they need a software which is platform independent, secure, environment independent, network independent. So to fulfill this requirement Java is proved.

Java and Internet:

In a network two types of objects are transmitted between the Server and the Client.

i) Passive information:-

ii) Active information:-

The information in E-mail, files downloaded are nothing but Passive information and the information like self executing programs, are Active information.

So in network or internet, it is desirable that the active information should be secure enough to download and should be portable. While i.e. While downloading or files or program from internet, there should be no threat to virus, and since in a network systems are having different

Work To Do

Important

Phones

2006

Thu

05

Jan

2nd Week • 005-360

FEB	S	M	T	W	T	F	S	S	M	T	W	F
	1	2	3	4	5	6	7	8	9	10	11	12
	13	14	15	16	17	18	19	20	21	22	23	24

Configuration and different environment, the program should be portable so that it will execute in all the system irrespective of configuration and environment.

Applet:- Java addresses those corner with a new program called Applet. An applet is designed to be application designed to be transmitted over the internet and executed by a Java compatible web browser. It can dynamically downloaded across the network. It is a program that react to user input and dynamically changed. Since Java assures security by providing a "firewall" between a network application and your computer, Applets can safely downloaded by a Java compatible web browser, with no fear of viral infection.

Java and Byte Code:- The security and portability of Java program is due to the non-executable code of the output of Java compiler. When a Java compiler compiles the Java program, the result is not an executable program but a non-executable bytecode. Bytecode is a highly optimized set of instructions designed to execute by Java Virtual Machine (JVM). Since the JVM can interpret the Java bytecode, it offers the security. And also translating a Java program into bytecode helps makes it easier to run a program in a wide variety of environment. Since Java environment has bytecode and if the system can be loaded or implemented by JVM.

Work To Do

Important

Phones

IMP
Name
Res.
Off.
E-mail
Mobile
Name
Res.
Off.
E-mail
Mobile

2006~

06

Jan

2nd Week • 007-351

WORK

Fri

JAN	S	M	T	W	T	F	S	M	T	W	F
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24

Just-In-Time (JIT):- JIT is a Compiler for bytecode. When the JIT Compiler is present in JVM, it compile the bytecode into executable code in real time on demand basis. It's not possible to compile an entire Java program into its executable code all at once, because Java performs various run-time checks that can be done only at runtime so JIT compile the code when it is needed during execution. JIT provides dynamic compilation of bytecode along with portability and security.

Dynamism of Java:-

Java is a simple, secure, portable, object-oriented, robust, multithreaded, architectural-neutral, interpreted, high performance, distributed, dynamic programming language. Simple; Java is designed to be a very simple language. For a professional programmer to learn, it inherits the properties and syntax of C and C++. It also inherits the Object Oriented Concept of C++ so a programming professional having knowledge about C and C++ can easily understand Java. Another thing that Java compiler does is that it eliminates the confusing concept of C++ like, Pointer etc.

Portable:- Since Java is portable it has to error free or bug free while creating the program. Java promote the technique in which so many complex errors can be eliminated by developing process and run-time process so the output is an error-free program.

Work To Do

Important

Phone

2006

Sat

07

Jan

FEB	S	M	T	W	T	F	S	M	T	W	F
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24

Ex:- In traditional programming language like C/C++, the memory management can be a difficult task. Here the programmer can de-allocate / free the memory reference manually. If the programmer forgets to free memory that has been previously allocated or try to free the memory that another part of the code is still using, a serious problem occurs. This can be eliminated by Java by executing a automated program called Garbage Collector (Which collects all the unused objects).

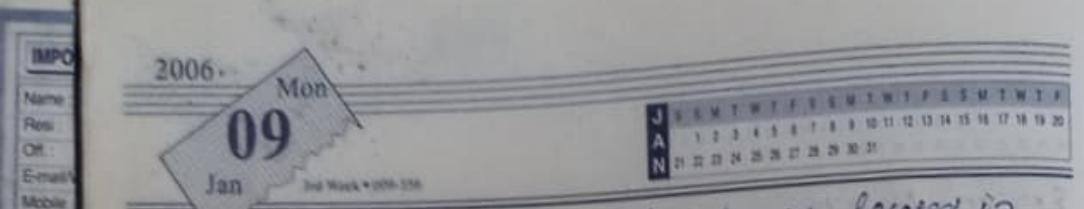
Some exceptional conditions like divided by zero can be handled by exception handling technique which is provided by Java. Architectural-Neutral:- One of the main problem facing the programmes is that there is no guarantee such that if you write a program today, will run tomorrow. This is due to the upgradations of CPU, OS, System, resources etc. This can be eliminated by Java by developing JVM which can run the bytecode at any environment provided lack System have JVM installed so it makes Java Architectural Neutral.

Multithreaded:- Java is designed to meet the real-world requirement of creating interactive Network programs. In Multithreaded programming, the different components of a program can execute simultaneously while gives the concept of multiprocess system.

Work To Do

Important

Phone

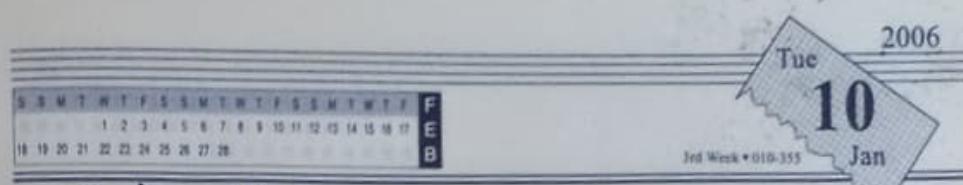


Distributed: Since all internet computers are located in many geographical areas so Java is designed to support distributed environment of the internet. This is due to the support of TCP/IP protocol by Java. The features of Java like RMI (Remote Method Invocation) enables a system to access the Method or files that reside in remote computers. This feature bring an unparalleled layer of abstraction to Client-Server Programming.

Dynamic: Java programs carry with them substantial amount of run-time type information that is used to verify and resolve to object at run-time. This makes it possible to dynamically link code in a safe and expedient manner.

Interpreted and High performance: Java enables the creation of cross-platform program by compiling into an intermediate representation called Java byte code. This code can be interpreted into our system that provide a Java virtual machine. Main Java JVM is now available with Just-In-Time (JIT) Compiler so that it can directly translate into it's native code which enhance the performance of the system.

Object-Oriented: In Java everything is treated as an object. Since object is present, it provide all the OOPS features like, Encapsulation, data abstraction, inheritance, Polymorphism etc. It provide security and reusability of code, so it enable us to create an object oriented program easily which is secure and reliable.



QOP PRINCIPLES

- 1 Programming language called Object Oriented Programming language of it supports Allene three principles.

 - ▷ Encapsulation, (ii) Inheritance, (iii) Polymorphism
 - i) Encapsulation - It is the mechanism that binds together code and the data it manipulates and keeps both safe from outside interference and misuse of object Oriented language code and data may be combined in such a way that a self-contained black box is created. When code and data are linked together in this fashion ~~a class~~ ^{class} is created which support encapsulation.

Within a class code, data or logic may be private to that class/object or public. Private data or code is known to or accessible to the program that exists outside the class. When a code/data are in public other part of a program may access it though it is defined within the class i.e. the public parts of an object/class are used to provide a controlling interface to the private elements of that class.

iii) Inheritance: It is the process by which one object can acquire the properties of another object it supports the concept of classification i.e. hierarchical classification. Without the use of classification, each object would have to define explicitly all of its characteristics. However, through the use of classification, an object need only define those qualities that make it unique within its class. It is the inheritance mechanism that makes it possible for one object to be a specialized instance of more general class. Class

IMPO
Name
Resi
Off
E-mail/W
Mobile
Name
Resi
Off
E-mail/W
Mobile

2006

Wed

11

Jan

3rd Week • 011-354

S	S	M	T	W	T	F	S	M	T	W	F	S	M	T	F
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
18	19	20	21	22	23	24	25	26	27	28	29	30	31		

iii) Polymorphism: Polymorphism is a feature or an attribute that allows one interface to control access to a general class of actions. The specific action selected is determined by the exact nature of the situation. Polymorphism helps reduce complexity by allowing the same interface to be used to access a general class of actions. It is the compiler's job to select the specific action do it applies to each situation.

Example of Java Program:

1) In Eclipse:

1) Prog. to display a msg. on the screen.

CLASS EX1

```
public static void main(String args[])
{
    System.out.println("Hello");
}
```

The file name must be EX1.java since it is the same as class name which contains the main() function. This is because a program may contain more than one class and at compile time all the classes are converted to its class file with same name as of class in Java program.

Work To Do

Important

Phones

2006

Thu

12

Jan

3rd Week • 012-353

S	S	M	T	F	S	M	T	F	S	M	T	F	S	M	T
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
18	19	20	21	22	23	24	25	26	27	28	29	30	31		

so when JVM interprets the byte code it searches for the main() function. so the main() program file name must the same name as the class name which contains the main() function.

The Public Key Word is an access specifier which is used to access the main() function from outside the class. This is JVM which is an outsider and is not responsible for interpreting the bytecode. so public key word is used.

The static keyword is an access provider is used in main() function. ED Calling Since JVM is a static so it can call main() which is static function directly without creating object of the class EX1.

void is a keyword which is a returning type. It is also for main() function. The main() function is returning null so void is a return.

String args[] :- This is a parameter in main() function. This parameter is an object String object which takes String at command line.

System.out.println(); This statement is used to print a msg. on the screen. System is a class which contains the I/O related functions. Out is a Stream object which is a buffer to from out the stored msg. is displayed on screen. println() is a building in System class which prints job is to display a msg. on the screen.

Work To Do

Important

Phones

IMPO

Name

Resi

Off

E-mail

Mobile

2006

Fri

13

Jan

3rd Week • 013-352

J	J	E	M	T	W	T	F	S	M	T	W	T	F
A	B	C	D	E	F	G	H	I	J	K	L	M	N
JAN	21	22	23	24	25	26	27	28	29	30	31		

DATA TYPE :- Java identifies eight data types of data. These data are classified into 4.

- **Primitives :-** In Java there are 8 primitive types we defined.

i) **byte :-**

ii) **short :-**

iii) **int :-**

iv) **long :-**

v) **byte :-** This is a signed 8-bit type and that has a range -128 to 127. byte types are useful when one working with stream of data from a network file. It is also useful when one working with raw binary data that may not be compatible to Java's other built-in type.

Syntax :- byte <variable name>;
byte a, b;

vi) **Short :-** It is a signed 16-bit type. So it's range is -32768 to +32767. This type is used in 16 bit computers.

Syntax :- short <variable name>;
short s, t;

vii) **int :-** It is a signed 32-bit type that has a range -2,147,483,648 to 2,147,483,647. This type of data commonly used in control loops, Control Array etc.

Work To Do

Important

Phones

2006

Sat

14

Jan

FEB

3rd Week • 014-351

Syntax :- long <variable name>;
long a, b;

viii) **long :-** long is a signed 64-bit type and useful for large calculations. Here int is not enough larger to store the value of long range is

-9,223,372,036,854,775,898 to 9,223,372,036,854,775,897

Syntax :- long <variable name>;

long a, b;

Ex2. JAVA

11 Proj to compute distance light travels

Class Ex2

{
public static void main (String ac[]){

}
}

lightSpeed;

long day;

long second, distance;

lightSpeed = 186000; // miles per sec

day = 1000

second = day * 24 * 60 * 60; // total sec

distance = lightSpeed * second;

System.out.println (" " + day + " days light will ");

System.out.println (" travel about " + distance + " miles");

}

Work To Do

Important

Phones

15 Sun

2006

16

Jan

Mon

JAN	
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30
31	

a) Floating-Point types: 37 Java has two types one of which are predefined in floating type.

i) float

ii) double

ii) float :- specifies single precision values which uses 32 bit of storage and is useful when we need a fractional component but does't require a large degree of precision of its range is 3.4×10^{-38} to 3.4×10^{38}

Syntax: float variable name;
float a, b;

ii) double:- specifies double precision values which uses 64 bit of storage, when one need to maintain accuracy over many iterations, calculation, double is used. Its range is 1.7×10^{-308} to 1.7×10^{308}

Syntax: double variable name;
double a, b;

Ex. JAV

class CD & Ex3

public static void main(String args[])

```
double pi, r, a;
pi = 3.1416;
r = pi * r * r;
a = pi * r * r;
```

Work To Do	Reported	Planned

2006

17

Jan

Tue

JAN	
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30
31	

FEB

System.out.println("Area of Circle is " + a);

3) Characters :- on java character type data we store in CHW type data. Java uses 16 bit unicode characters to represent characters unicode characters a fully interpretation. Character set (ISO) can represent all the characters found in human language its range is 0 to 65536. Since java is designed attempt to be written for worldwide use, it makes the string that it would use unicode to represent characters.
Syntax: char variable name;
char a;

Syntax :-

class Ex4

{
public static void main(String args[])

{
char ch1, ch2;

ch1 = 88;

ch2 = 'y';

System.out.println("CH1 and CH2 : " + ch1 + " " + ch2);
}

Work To Do	Reported	Planned

2006

Wed

18

Jan

A8 WORK • 018.347

	S	E	M	T	W	T	F	S	M	T	W	F								
JAN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	21	22	23	24	25	26	27	28	29	30	31									

Output : Ch1 and Ch2: x y

Ch1 is assigned 88 Which is Ascii value of x.

~~88~~ ASCII characters occupies 127 values
of Unicode character set

4) Boolean: Java desires a new data type boolean
for logical values. It's value either true or
false. It is used in relational operators etc. It
is 1 bit long.

Syntax: `boolean <variable name>;`
`boolean a;`

Ex5.java

Class Ex5

{

`public static void main (String ac[])`

{

`boolean b;`

`b= false;`

`System.out.println ("b is "+b);`

`b= true;`

`if (b)`

`System.out.println ("This is Executed");`

`System.out.println ("10718 is "+(10718));`

,

}

Work To Do

Important

Notes

S	S	M	T	W	T	F	S	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28

F
E
B

Java Literals :- Literals means Constants. In Java there are five types of literals.

i) Integer literals :- Any Whole number values are integer literals. Integer literals are three types. decimal number, hexadecimal number and octal number.

Decimal number :- Any a Whole number Ex:- 1, 9, 22 are in decimal numbers since their base is 10.

Octal numbers :- The base of octal numbers is 8.

so it can contains the number 0 to 7. It can be represented by leading '0'. Ex:- 06, 07 etc.

hexadecimal numbers :- The base of hexadecimal numbers is 16 so it contains the same numbers 0 to 15 and A through F can replace the number from 10 to 15.

They can be represented by leading '0X'.

Ex:- 0XF, 0x9 etc.

When an integer literals is assigned to long variable, then they ~~that literal as~~ a L or l is appended at the end of char literal. Ex 123L = 0x11L 0x9l etc.

ii) Floating type basic literals :- The floating numbers are represented by two ways.

i) standard notation :- It consists of a Whole number component followed by a decimal point followed by a fractional component. Ex 2.0, 3.1412 etc.

ii) Scientific notation :- It is a standard notation.

Floating point number plus a suffix E that specifies a power of 10 by which the number is to be multiplied.

Ex :- 6.02E23, 6.09E-02 etc.

2006

Fri

20

Jan

4th Week • 030-345

By default the floating point constants are stored as double. so to convert into float one have to append `f` or `F` at the end of the double literals.

3) Boolean literals: Boolean literals we only takes two values true or false.

Ex: Boolean bit true; etc.

4) Character literals: The Character literals are the characters enclosed within single quotes.

Ex: 'a', 'Z' etc.

We can use a many escape sequence to declare the character octet as our need. This can be done by using \. eg:

Ex:- \ddd → Octal Character (ddd)

\xxxx → Hexadecimal Unicode Character (xxxx)

' → Single quote;

" → Double quote;

\b → Back slash

\r → Carriage return.

\n → New line.

\t → Tab.

\b → Backspace.

\f → Form feed.

For octal notation use backslash followed by the three digit number.

For hexadecimal notation use backslash followed by four digit number.

Work To Do

Important

Phones

JAN	SUN	M	T	W	T	F	S	SUN	M	T	W	F	S	SUN
21	22	23	24	25	26	27	28	29	30	31	1	2	3	4

2006

Sat

21

Jan

4th Week • 031-344

FEB	SUN	M	T	W	T	F	S	SUN	M	T	W	F	S	SUN
18	19	20	21	22	23	24	25	26	27	28	1	2	3	4

5) String literals: String literals are the group of characters enclosed within a pair of double quotes. Ex:- "Hello World" etc.

Type Casting: In an assignment statement, when one type of value is assigned to a variable of different type then we use type casting. i.e. to obtain the convertibility between incompatible type. we must use 'cast' which performs an explicit conversion between incompatible type.

Type Casting Can be of two types

i) Automatic Type Conversion (Convertibility between compatible type)

ii) Conversion between incompatible type.

→ Conversion between compatible type: - When one type of data is assigned to another type of variable, an automatic type conversion will take place if the following conditions are met.

a) The two types are compatible.

b) The destination type is larger than the source type.

For example, the int type is always larger than valid byte values, so no explicit cast statement is required.

byte b;

int c;

b = 10;

c = b;

22 Sun

Work To Do

Important

Phones

2006

Mon
23

Jan

5th Week • 023-342

JAN	SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	31				

b) casting incompatible types - To Create a conversion between two incompatible types, we use a Cast. A Cast is simply a explicit type conversion. its Syntax is (target type) variable;

Here target type specifies the desired type to convert the specified value to. For example if the integers value is larger than the range of a type, it will be reduced modulo type's range.

int a;

byte b;

b = (byte) a;

Only a floating-point values is assigned to an integer type then the Casting occurs, in which the fractional component of the floating point numbers are truncated, and if the size of the whole number component is too large to fit into an target integer type then that value will be reduced modulo the target type's range.

double d;

d = a;

a = (int) d;

--x--

Work To Do

Important

Phone

2006

Tue
24

Jan

FEB	SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	31				

Operators

Different operators in java are classified into following categories:

1- Arithmetic Operator: These operators are used in mathematical expressions. e.g. ~~the~~ Operators

+

-

*

/

%

++

+=

-=

*=

/=

%+=

--

Result.

Addition

Subtraction

Multiplication

Division

Modulus.

Increment.

Addition Assignment.

Subtraction Assignment

Multiplication Assignment

Division Assignment

Modulus Assignment

Decrement

The onward need in an expression eve integer or character type.

Work To Do

Important

Phone

2006

Wed

25

Jan

5th Week • 825-340

Ex:- Class Arithmetic

```

public static void main(String args[])
{
    System.out.println("Integer Arithmetic");
    int a = 1 + 1;
    int b = a * 3;
    int c = b / 4;
    int d = c - a;
    int e = -d;

    System.out.println("a=" + a);
    System.out.println("b=" + b);
    System.out.println("c=" + c);
    System.out.println("d=" + d);
    System.out.println("e=" + e);

    System.out.println("Floating Arithmetic");
    double da = 1.1;
    double db = da * 3;
    double dc = db / 4;
    double dd = dc - a;
    double de = -dd;

    System.out.println("da=" + da);
    System.out.println("db=" + db);
    System.out.println("dc=" + dc);
    System.out.println("dd=" + dd);
    System.out.println("de=" + de);
}

```

Work To Do	Important	Phone

2006

Thu

26

Jan

5th Week • 826-340

Output:-

Integer Arithmetic

a=2

b=6

c=1

d=-1

e=1

Floating point Arithmetic

da=2.0

db=6.0

dc=1.5

dd=-0.5

de=0.5

iii) Bitwise Operator:-

In Java Bitwise operators are used applying to the integer type, long, int, short, char and byte. These operators act upon the individual bits of their operands.

Overall

~

&

|

^

>>

>>>

<<

&=

|=

^=

>>=

>>>=

<<=

Bitwise unary NOT

Bitwise AND AND

Bitwise OR

Bitwise Exclusive OR

Right Shift

Shift Right Zero fill

Shift Right

Bitwise AND Assignment

Bitwise OR Assignment

Bitwise Exclusive OR Assignment

Shift Right Assignment

Shift Right Two bit Assignment

Shift Left Assignment

IMP
Name
Resi
Off.
E-mail
Mobile
Name
Resi
Off.
E-mail
Mobile

2006

Fri

27

Jan

5th Week • 027-335

J	S	M	T	F	S	M	T	F	S	M	T	F	S	M	T	F			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31									
16	19	20	21	22	23	24	25	26	27	28									

A	B	A+B	A*B	A^B	~A
0	0	0	0	0	1
0	1	1	0	1	0
1	0	1	0	1	1
1	1	1	1	0	0

Ques:-
class BitExp

```
public static void main(String args[])
{
    String binaryP7 = {"0000", "0001", "0010", "0011", "0100",
    "0101", "0110", "0111", "1000", "1001", "1010", "1011",
    "1100", "1101", "1110", "1111"};
    int a = 8;
    int b = 4;
    int c = a+b;
    int d = a*b;
    int e = (a^b);
    int f = (~a&b);(a&~b);
    System.out.println("a = "+binary[0]);
    System.out.println("b = "+binary[1]);
    System.out.println("a+b = "+binary[2]);
    System.out.println("a*b = "+binary[3]);
    System.out.println("a^b = "+binary[4]);
    System.out.println("~a&b; a&~b = "+binary[5]);
}
```

1

Work To Do

Important

Phones

2006

Sat

28

Jan

J	S	M	T	F	S	M	T	F	S	M	T	F	S	M	T	F			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31									
16	19	20	21	22	23	24	25	26	27	28									

Output:-

$$a = 0011$$

$$b = 0110$$

$$a+b = 0111$$

$$a*b = 0010$$

$$a^b = 0101$$

$$\sim a \& b; a \& \sim b = 0101$$

Left Shift (<<) and Right Shift (>>)

The Left Shift operator shift all the bits in a value to the left a specified no. of times.

Syntax:- value << num

Here the num specifies the number of positions to left-shift the value in "value". For each shift left, the high order bit is shifted out and a zero is brought in at the right.

class LeftShiftEx

{

```
public static void main(String args[])
{

```

```
byte a = 64, b;
```

```
i = a << 2;
```

```
b = (byte) (a << 2)
```

```
System.out.println("Original value of a = "+a);
System.out.println("i and b = "+i+" "+b);
```

```
}
```

```
}
```

29 Sun

Work To Do

Important

Phones

2006

Mon

30

Jan

Output

Original value of a = 2.

a and b = 2 & 3

The print statement shows that all the code
in a class has to run inside a changing number of
times.

377123 - value 377123.

Here "this" represents the address of the object to
which this line refers. In "value" - it is a variable
from which there are different file - class with same
name.Class Representations

I

public static void main(String args[])

 {
 a = 2;
 b = 3;
 System.out.println("a = " + a);
 System.out.println("b = " + b);
 c = a + b;
 d = a * b;
 }

System.out.println("a = " + a);

System.out.println("b = " + b);

System.out.println("c = " + c);

System.out.println("d = " + d);

}

Output:

a = 2

b = 3

c = 5

d = 6

2006

Tue

31

Jan

1) Relational OperatorsThe relational operators determine the relationship
that one operand has to the other. They determine the
equality and ordering.Operators

==

>

<

>=

<=

Equal to

Not equal to
Greater than
Less thanGreater than or equal to
Less than or equal toClass RelationshipsI
public static void main(String args[]){
 a = 0 + 4;{
 b = 1;

boolean C = a < b;

System.out.println("a = " + a);

System.out.println("b = " + b);

System.out.println("C = " + C);

{
}Output:-

a = 4

b = 1

C = false

2006

Wed

01

Feb 6th Week • 032-333

F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
E														
B	18	19	20	21	22	23	24	25	26	27	28			

iv) Boolean Logical Operators:- The Boolean logical operators operate only on Boolean operators. All of the binary logical operators combine two Boolean values to form a resultant Boolean value.

Operators

&

|

^

||

^=

!

a&=

|=

^|=

==

!=

?:

Result

Logical AND

Logical OR

Logical XOR

Short-Circuit OR

Short-Circuit AND

Logical Unary NOT

AND Assignment

OR Assignment

XOR Assignment

equal to

not equal to

Ternary if-Else Statement

A	B	A&B	A&B	A&B	!A
False	False	False	False	False	True
False	True	False	False	True	False
True	False	False	False	True	False
True	True	True	True	False	False

Class BooleanEx

{ public static void main (String args[]) }

Work To Do

Important

Phones

2006

Thu

02

Feb 6th Week • 033-332

F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
E														

MAR

boolean a = true;

boolean b = false;

boolean c = a || b;

boolean d = a & b;

boolean e = a ^ b;

boolean f = !a;

System.out.println ("a = " + a);

System.out.println ("b = " + b);

System.out.println ("a || b = " + c);

System.out.println ("a & b = " + d);

System.out.println ("a ^ b = " + e);

System.out.println ("!a = " + f);

}

Output:

a = true

b = false

a || b = true

a & b = false

a ^ b = true

!a = false.

→

Phones

Work To Do

Important

2006

03

Feb

2006 Week No. 201

Fri



CONTROL STATEMENTS

The Control Statements in Java are classified into two types:

① Selection Statement: These statements allow us to control the flow of program's execution based on the conditions known only during run time.

~~Ex:~~

② If statement: It is a conditional branch statement. It will can be used to route the program execution through two different paths.

Syntax:

If (condition) Statement;

else

Statement2;

If the condition is true then Statement1 is executed otherwise Statement2 will be executed.

Code: C/C++

```
public static void main(String args[])
{
    int a = 50;
    int b = 50;
```

```
if(a > b)
    System.out.println("a is greater");
else
    System.out.println("b is greater");
```

Output:

50 is greater

2006

04

Feb

2006 Week No. 201

Sat

MAR

Nested if: A nested if is an if statement that is the target of another if or else.

Syntax:

If (condition){

If (condition2){

If (condition3){

 }

 Else Statement.

}

 Else Statement.

Class: C/C++

```
public static void main(String args[])
{
    int i = 10;
    int j = 15;
    int k = 150;
    int a;
```

```
if(i == 10){
```

```
    if(j < 20) a = 10;
```

```
    if(k > 10) a = 20;
```

```
    else a = 150;
```

```
    else a = 150;
```

```
}
```

05 Sun

```
System.out.println("Value of a = " + a);
```

}

1

2006

Mon

06

Feb*

This Week • 037.328

S	M	T	W	T	F	S	M	T	W	F	S	M	T	W	F	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31			

if-else-if Ladder :- Here the if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed and the rest of the ladder is bypassed. If none of the conditions is true then the final 'else' statement will be executed.

Syntax:if (condition)
Statement;else if (condition)
Statement;else if (condition)
Statement;

:

else
Statement;class IfElseEx

```
public class void main(String args[])
{
    int month=4;
    String season;
    if(month==12 || month==1 || month==2)
        season="Winter";
    else if(month==3 || month==4 || month==5)
        season="Spring";
```

2006

Tue

07

Feb

S	M	T	W	T	F	S	M	T	W	F	S	M	T	W	F	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31			

```
else if (month==6 || month==7 || month==8)
    season="Summer";
```

```
else if (month==9 || month==10 || month==11)
    season="Autumn";
```

```
else season="invalid";
System.out.println("April is in the "+season+");");
}
```

Output:

April is in the Spring.

Switch Statement:Syntax:

switch (expression)
{

case value:

 // Statement Sequence
break;

case value:

 // Statement Sequence
break;

:

case value:

 // Statement Sequence
break;

default:

// Default Statement Sequence

2006

Wed

08

Feb 7th Week • 039-326

	F	S	S	M	T	W	T	F	S	S	M	T	W	F
E	1	2	3	4	5	6	7	8	9	10	11	12	13	14
B	16	17	18	19	20	21	22	23	24	25	26	27	28	29

The "expressions" must be of type byte, short, int or char, each of the values specified in the 'Case' statements must be of a type compatible with the expressions. Each Case value must be a unique literal. Duplicate Case values are not allowed. The value of the expression is compared with each of the literal values in the 'Case' statements. If a match is found, the code sequence following that Case statement is executed. If no match is found then default statement is executed.

class SwitchEx

{

```
public static void main(String args[])
{
    for (int i = 0; i < 6; i++)
        switch (i)
    {
```

Case 0:

```
        System.out.println("i is zero.");
        break;
```

Case 1:

```
        System.out.println("i is one.");
        break;
```

Case 2:

```
        System.out.println("i is two.");
        break;
```

Case 3:

```
        System.out.println("i is three.");
        break;
```

default:

```
        System.out.println("i is greater than 3.");
```

3

2006

Thu

09

Feb 7th Week • 040-325

	F	S	S	M	T	W	T	F	S	S	M	T	W	F
E	1	2	3	4	5	6	7	8	9	10	11	12	13	14
B	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Output:-

i is Two.

i is One.

i is two.

i is three.

i is greater than 3.

i is greater than 3.

ii) Iteration Statements:- These statements create a loop which repeatedly executes the same set of instructions until a termination condition is met. The iteration statements are.

a) for loop, b) while loop c) do-while loop.

a) for loop:

Syntax:-

```
for (initialization; condition; iteration)
```

{

body

}

When the loop first starts, the initialization part of the loop is executed. It is executed only once. Next the condition is evaluated. This must be the boolean expression. It test the loop control variable with the target value. If this expression is true then the body of the loop is executed. If it is false then the loop terminates. Next the iteration part of the loop is executed. This is an expression while increment or decrement the loop control variable.

Work To Do

Important

Phones

2006

10

Feb

10 Weeks • 2005-2006

class FindPrime

{

public static void main (String args[])

{

int num;

boolean isprime = true;

num = 19;

for (int i = 2; i < num/2; i++)

{

if (num % i == 0) {

isprime = false;

break;

}

}

if (isprime) System.out.println ("prime");
else System.out.println ("Not Prime");

{

Output: Not prime.

9 While loop:

Syntax:

while (condition)

body

}

It repeats a statement or block while it's controlling expression is true. The condition can be inside the expression. The body of the loop will be executed as long as the conditional expression

2006

11

Feb

10 Weeks • 2005-2006

MAR

is true. When the condition becomes false, the control passes to the next line of the code immediately following the loop.

class Athlete

{

public static void main (String args[])

{

int i, j;

i = 150;

j = 280;

while (i < j)

{

i = i + 1;

j = j - 1;

}

System.out.println ("Midpoint is " + i);

{

Output: Midpoint is 150.

9 Do-while loop:

Syntax:

{

}

body

;} while (condition);

Since it's conditional expression is at the bottom of the loop, the body will be processed at least once.

12 Sun

2006

Mon

13

Feb 7th Week • 044-321

F	E	B	M	T	W	T	F	S	S	M	T	W	T
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Each iteration of the do-while loop first executes the body of the loop and then evaluates the conditional expression. If the expression is true then the loop will repeat otherwise the loop terminates. The condition must be boolean expression.

class DoWhileEx

```
public static void main(String args[])
throws java.io.IOException {
    char choice; int i=50, j=10;
    do
```

```
        System.out.println("1. Addition");
        System.out.println("2. Subtraction");
        System.out.println("3. Multiplication");
        System.out.println("4. Division");
        System.out.println("5. Exit");
        System.out.println("Enter your Choice:");
        Choice = (char) System.in.read();
    }
```

```
    while(Choice < '1' || Choice > '5') {
        switch(Choice)
    }
```

Case 1:

```
        System.out.println("i+" + "j+" = "+(i+j));
        break;
```

S	S	M	T	W	F	S	S	M	T	W	T	F	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14

MAR

8th Week • 045-320 Feb

Case 2:

```
        System.out.println("i+" - "j+" = "+(i-j));
        break;
```

Case 3:

```
        System.out.println("i+" * "j+" = "+(i*j));
        break;
```

Case 4:

```
        System.out.println("i/" + "j+" = "+(i/j));
        break;
```

Case 5:

```
        break;
    }
```

I

J

Output:-

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit

Enter your Choice:

2

$$50 - 10 = 40$$

Work To Do

Important

Phones

Work To Do

Important

Phones

2006

Wed

15Feb
8th Week • 046-319

	F	S	M	T	W	T	F	S	M	T	W	T	F
B	1	2	3	4	5	6	7	8	9	10	11	12	13
	14	15	16	17	18	19	20	21	22	23	24	25	26
	27	28	29	30	31								

Command line Arguments. - Command line arguments is the information that directly follows the program's name on the command line when it is executed. To access the command line arguments we should use String array in main() function.

class CmdlineEx

```
{  
    public static void main(String arg[])  
    {  
        for(int i=0; i<arg.length; i++)  
            System.out.println("arg[" + i + "] = " + arg[i]);  
    }  
}
```

When we run the prog. as follows
java CmdlineEx this is a test 100 -1

Output:

```
arg[0] = this  
arg[1] = is  
arg[2] = a  
arg[3] = test  
arg[4] = 100  
arg[5] = -1
```

~K

2006

Thu

16Feb
8th Week • 047-319

	S	S	M	T	W	T	F	S	S	M	T	W	F
M	1	2	3	4	5	6	7	8	9	10	11	12	13
	14	15	16	17	18	19	20	21	22	23	24	25	26
	27	28	29	30	31								

CLASS & OBJECT

Class is an abstract data type which support the encapsulation, protection of properties of obj. A Class is declared by use of the 'Class' keyword.
Syntax of class:-

```
class classname  
{
```

```
    type instancevariable;  
    type instancevariable;
```

```
=
```

```
    type instance variable;
```

```
    type MethodName(Parameter list){  
        body
```

```
}
```

```
    type MethodName(Parameter list){  
        body
```

```
}
```

```
=
```

```
=
```

```
    type MethodName(Parameter list){  
        body
```

```
}
```

```
=
```

```
=
```

```
    type MethodName(Parameter list){  
        body
```

```
}
```

Work To Do

Important

Phones

Work To Do

Important

Phones

2006

Fri

17

Feb

MS Word • Page 217

variables defined within the class is known as instance variables. This function is known as methods.

Class is a template. We can't directly access the data members or methods of a class rather we can access by the help of an object. An object is an instance of a class. It has some attributes and behaviour of its class. An object is created by new keyword. By using the object created and operator we can address the data members and methods functioning of a class.

A method can have the syntax as follows:

```
left name (parameter list)
    body
```

left specifies the type of data it returns. If the method doesn't return a value, it's return type must be 'void'. The name of the method is specified by 'name'. This is the only valid space or identifier. The Parameter list is a sequence of type and identifier pairs separated by commas. Parameters are essentially variables that receive the value of the arguments passed to the method when it is called. If the method has no parameter, then the parameter list is empty.

2006

Sat

18

Feb

MS Word • Page 218

Class Box

```
{  
    double width;  
    double height;  
    double depth;  
    void volume();  
}
```

```
System.out.println("volume is ");  
System.out.println(width * height * depth);
```

```
double volume()  
{  
    return width * height * depth;  
}
```

Class BoxDemo

```
public static void main(String args){  
}  
  
Box mybox1 = new Box();  
Box mybox2 = new Box();  
  
double vol;  
  
mybox1.width = 10;  
mybox1.height = 20;  
mybox1.depth = 15;  
  
mybox2.width = 5;  
mybox2.height = 6;  
mybox2.depth = 7;
```

19 Sun

2006

Mon

20

Feb 9th Week • 052-313

FEB	SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	31				

```

myBox1. volume();
vol = myBox1. volume();
System.out.println("volume = "+ vol);
myBox2. volume();
vol = myBox2. volume();
System.out.println("volume = "+ vol);
}

```

Method Overloading: When two or more methods defined within the same class that share the same name but differing in decreasing the parameters, then the methods are said to be Overloaded. and this process is known as Method Overloading.

When an Overloaded Method is invoked, Java uses type and/or no. of arguments to determine which version of the Overloaded method to actually call. so Overloaded methods must differ in the type and/or number of their parameters. When an overloaded method is called, Java looks for a match between the argument used to call the method and the method's parameters.

```

class MethodOverloadDemo
{
    void test()
    {
        System.out.println("Hello");
    }
}

```

2006

Tue

21

Feb 9th Week • 052-313

MAR

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

```

System.out.println("10 Parameter");
}

```

```

void test(int a)
{

```

```

System.out.println("a: "+ a);
}

```

```

void test(int a, int b)
{

```

```

System.out.println("a and b: "+ a+ " "+ b);
}

```

```

double test(double a)
{

```

```

System.out.println("double a: "+ a);
return a;
}

```

```

class Overload
{

```

```

public static void main(String args[])
{

```

```

MethodOverloadDemo ob = new MethodOverloadDemo();
double result;
ob.test();

```

```

ob.test(10);

```

```

ob.test(10.20);

```

```

result = ob.test(123.2);

```

```

System.out.println("Result of ob.test(123.2): "+ result);
}

```

Work To Do

Important

Phone

Work To Do

Important

Phone

2006

Wed

22

Feb 9th Week • 053-312

Output:

No Parameters.

a:10

a and b:10, 20

double a: 123.2

Result of ob. Lese(123.2): 15178.29

CONSTRUCTOR

A Constructor initializes an object immediately upon creation. If has the same name as the class in which it resides and is syntactically similar to a method. Once defined, the constructor is automatically called immediately after the object created, before the 'new' operator completes. The constructor has no return type.

Syntax:

classname (parameter list)

{
body
}

If there the constructor has no parameter, then it is known as default constructor. If it has some parameter, it is known as parameterized constructor.

Work To Do

Important

Phone

2006

Thu

23

9th Week • 054-311

Feb

F	E	M	T	W	T	F	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30	31								

F	E	M	T	W	T	F	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30	31								

class ConstructorEx

double width;

double height;

double depth;

~~ConstructorEx()~~

{

System.out.println("Default constructor");

width = 10;

height = 10;

depth = 10;

}

double volume();

{

return width * height * depth;

}

Class ConstructorDemo

{

public static void main(String args[])

}

ConstructorEx ce = new ConstructorEx();

double vol;

vol = ce.volume();

System.out.println("Volume is :" + vol);

}

Output: Default Constructor

VOLUME IS : 1800.0

Photos

2006

Fri

24

Feb

1st Week • 005-318

// Parameterized Constructor.

class PclassEx

{

double width;

double height;

double depth;

PclassEx(double w, double h, double d)

{

width = w;

height = h;

depth = d;

}

double volume()

{

return width * height * depth;

}

class PclassDemo

{

public static void main(String args[])

PclassEx pc = new PclassEx(10, 20, 30);

double vol;

vol = pc.volume();

System.out.println("volume is:" + vol);

}

Output:- volume is: 6000.0

Work To Do

Important

Phone No.

F	E	B	S	M	T	W	T	F	S	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	

2006

Sat

25

Feb

1st Week • 005-318

Overloading Constructors

Constructors are also overloaded like normal Methods in Java. All the Constructors have same name but differ in parameter. During execution, a proper Constructor will be called, according to the argument passed during the object creation.

class Box

{

double width;

double height;

double depth;

Box()

{

width = -1;

height = -1;

depth = -1;

}

Box(double w, double h, double d)

{

width = w;

height = h;

depth = d;

}

Box(double len)

{

length = width = depth = len;

}

26 Sun

Work To Do

Important

Phone No.

Work To Do

Important

Phone No.

2006

Mon

27

Feb

10th Week • 059-307

class volume()

{
 return height * depth * width;}
class OverloadCons

public static void main(String args[]){

```

Box mybox1 = new Box(10, 20, 30);
Box mybox2 = new Box(10);
Box mybox3 = new Box();
mybox1.double vol;
vol = mybox1.volume();
System.out.println("volume = " + vol);
vol = mybox2.volume();
System.out.println("volume = " + vol);
vol = mybox3.volume();
System.out.println("volume = " + vol);
    }
```

Output:
 volume = 6000.0
 volume = 1800.0
 volume = -1.0

Work To Do

Important

Phone

2006

Tue

28

Feb

10th Week • 059-306

F	A	M	I	J	J	S	T	E	N	D	M	J	J	S	T	E	N	D	M	A	R									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

"This" Keyword:

When a method will need to refer to the object that invoked it, then the 'this' keyword is used. This can be used inside any method to refer to the current object. That is, 'this' is always a reference to the object on which the method was invoked.

Class Box

{

```

double width;
double depth;
double height;
```

```
Box(double width, double depth, double height)
```

{

```

this.width = width;
this.depth = depth;
this.height = height;
```

{

```
double volume()
```

{

```
    return width * height * depth;
```

{

class BoxDemo

{

```
public static void main(String args[])
```

{

```
    Box mybox = new Box(10, 20, 30);
    double vol;
```

Work To Do

Important

Phone

2006

Wed

01

Mar

10th Week • 060-305

```

vol = mybox.volume();
System.out.println("volume=" + vol);
}
    
```

Output:- volume=600.0

M	S	T	F	S	M	T	W	T	F	S	M	T	W	F			
MAR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31				

2006

Thu

02

Mar

10th Week • 061-304

M	S	T	F	S	M	T	W	T	F	S	M	T	W	F	
APR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

INHERITANCE

It is the feature of OOPS by which one class can get the properties and behavior of another class. The class from which the properties and behavior the properties are inherited is known as Base Class and the class which inherit the properties of base class is known as Child Class or derived class.

In Java a class can be inherited from base class by using "extends" keyword. Since in Java three access Specifiers are specified i.e. private, protected & public, we can use these specifier in Methods and variables. If a method or variable is declared as public, it can be accessed ~~by~~ ~~at~~ inside that class and also the class which inherit it. If ~~it~~ the method or variable is declared as private, they they can only be accessed in side that class ~~and~~ ~~by~~ by other class or inherited class. If a method or variable is declared as protected then they can accessed ~~by~~ ~~in~~ inside that class and also accessed by the classes and inherited classes defined in the same package.

2006

Fri

03

Mar

100 Weeks • 2002-2003

Class A

```
public class A {
    int i;
    private int j;
    void sum(int a, int b) {
        i = a;
        j = b; int c = i + j;
        j = b;
        return i;
    }
}
```

Class B Extends A

```
class B extends A {
    int total;
    void sum() {
        // total = i + j;  $\rightarrow$  Error
        total = i + getj();
    }
}
```

Class Access

```
public static void main(String args[]) {
    B Subobj = new B();
    Subobj.sum(10, 20);
    Subobj.sum();
    System.out.println("total=" + Subobj.total);
}
```

Output: Total = 30

MARCH											
SUN	MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU
1	2	3	4	5	6	7	8	9	10	11	12

2006

Sat

04

Mar

100 Weeks • 2002-2003

Super Key Word: - Super is a keyword which is used by a subclass when it needs to refer to the variables or methods of its immediate superclass. Super has two general forms.

i) Using super to call super class constructor:

A subclass can call a constructor method defined by its superclass by use of following form of Super:

Super(parameter-list)

The parameter list specifies any parameters needed by the constructor in the super class. 'super' must always be the first statement executed inside a subclass constructor.

Class Box

```
private double width;
private double height;
private double depth;
Box()
```

```
width = -1;
height = -1;
depth = -1;
```

Box(double w, double h, double d)

```
{ width = w;
  height = h;
  depth = d;
```

05 Sun

2006

Mon

06

Mar

11th Week • 065-200

double volume()

return width * height * depth;

Class BoxDight extends Box

double weight;

BoxDight(double w, double h, double d, double w)

Super(w, h, d);

Weight = w;

BoxDight()

Super();

Weight = -1;

Class DemoSuper()

public static void main(String args[])

BoxDight mybox1 = new BoxDight();

BoxDight mybox2 = new BoxDight(10, 20, 15, 3);

double vol;

vol = mybox1.volume();

System.out.println("Volum = " + vol);

In vol = mybox2.volume();

System.out.println("Volum = " + vol);

M	T	W	T	F	S	S	M	T	W	T	F	S	M	T	W	F
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31			

2006

Tue

07

Mar

11th Week • 066-200

M	T	W	T	F	S	S	M	T	W	T	F	S	M	T	W	F
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30				

APR

ii) Second case of "Super": "Super" can also be used to access a member of the Super class that has been hidden by a member of a subclass.

Super.member.

The member is either a method or a variable. The second form of Super is most applicable to situations in which member names of subclass hides the members by the same name in the superclass.

39:

Class A

{

int i;

}

Class B extends A

{

int i;

B(int a, int b)

{

Super.i = a;

i = b;

}

void show()

{

System.out.println("i is Super's i & Super.i");

System.out.println("i is subclass's i");

Work To Do

Important

Phone

2006

Wed

08

Mar

110 Work • 0807298

Class uses Super

{ public static void main (String args) }

B subobj = new B (1, 2);
subobj = & short);Multilevel Hierarchy:

In multilevel hierarchy, we can let a subclass act as a superclass of another. This hierarchy contains more than one level.

Ex:-

Class Box

{ private double width;

private double height;

private double depth;

Box (Box Obj)

{

Width = Obj. Width;

height = Obj. height;

depth = Obj. depth;

Box (double W, double h, double d)

{

Width = W;

Work To Do	Priorities	Phone

2006

Thu

09

Mar

110 Work • 0807297

M A R M A R C H 2 0 0 6
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30

A P R

height = h;

depth = d;

},

Box ()

{

width = -1;

height = -1;

depth = -1;

}

double volume()

{

refer width * height * depth;

}

Class BoxWeight extends Box

{

double weight,

BoxWeight (BoxWeight Obj)

{

Super (Obj);

Weight = Obj. weight;

}

BoxWeight (double N, double h, double d, double m)

{

Super (N, h, d);

Weight = m;

}

Work To Do	Priorities	Phone

2006

Fr

10

Mar

M	A	R
1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18
19	20	21
22	23	24
25	26	27
28	29	30
29	30	31

constructor()

Sub();

Weight = -1;

}

class Shipment extends Boxwise

{ double cost;

Shipment(Shipment ob)

{

super(ob);

cost = ob.cost;

}

Shipment(double d1, double d2, double d3, double m, double n)

{

super(d1, d2, d3);

cost = 7;

}

Shipment()

{

super();

cost = -1;

}

}

Date To Do

Important

Notes

2006

Sat

11

Mar

APR

class DemoShipment

{ public static void main (String args [])

{ Shipment shipment1 = new Shipment(10,20,15,10,3,4);

double vol;

vol = shipment1 . Volume();

System.out.println("Volume of Shipment1 is " + vol);

System.out.println("Weight of Shipment1 is " +

shipment1 . Weight());

System.out.println("Cost of Shipment1 is " +

shipment1 . cost());

}

}

ABSTRACT CLASS:

Abstract class is a Super class that declares the structure of a given abstraction without providing a complete implementation of every method in it.

An Abstract class is a Subclass that may define the generalized form that will be shared by all the subclasses, leaving it to each subclass to fill in the details.

Abstract Method is a method that is overridden by a subclass by specifying the abstract type modifier and the class which does not contain the abstract methods is known as Abstract Class.

Date To Do

Important

Notes

2006

Mon

13

Mar

12th Week • 072-293

M	T	W	T	F	S	M	T	W	F	S	M	T	W	F
A	1	2	3	4	5	6	7	8	9	10	11	12	13	14
R	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	29	30	31											

Syntax for Abstract Method

abstract type name(parameter-list);

Any class that contains one or more abstract methods must also be defined abstract. There can be no objects of an abstract class; one can't define abstract constructors or abstract static methods. Any subclass of an abstract class must either implements all the abstract methods in the superclass or be itself defined abstract.

Ex:

Abstract Class Figure

```
double dim1;
double dim2;
Figure(double a, double b)
```

dim1 = 5;

dim2 = 5;

Abstract double area();

Class Rectangle extends Figure.

Rectangle(double a, double b)

super(a, b);

Phones

Tue

2006

14

Mar

12th Week • 073-292

M	T	W	T	F	S	M	T	W	F	S	M	T	W	F
A	1	2	3	4	5	6	7	8	9	10	11	12	13	14
R	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	29	30	31											

double area()

{

System.out.println("area of Rectangle");

return(dim1 * dim2);

}

Class Triangle extends Figure.

{

Triangle(double a, double b)

{

super(a, b);

}

double area()

{

System.out.println("triangle");

return(dim1 * dim2);

}

}

Class AbstractDemo

{

public static void main(String args[])

Rectangle r = new Rectangle(20, 30);

Triangle t = new Triangle(20, 10);

Figure f = new Figure(10, 10) → error;

Figure f; //OK

Phones

Work To Do

Important

Phones

2006

Wed

15

Mar

126 WwW • 975-291

```
f = 5;
System.out.println("Area is "+f*area());
f = 5;
System.out.println("Area is "+f*area());
}
```

Final Keyword
The keyword **final** has three uses:

- When it is used before a variable, that variable is treated as constant.
Ex: final int FILE_OPEN=1;
final int FILE_CLOSE=2;
All final variables must be initialized.
- To disallow a method from being overridden, **final** keyword is used as a modifier at the start of its declaration. i.e. methods declared as **final** cannot be overridden.
Ex:

CLASS A

```
final void meth()
{
    System.out.println("This is a final method");
}
```

Work To Do

Important

Phone

2006

Thu

16

Mar

126 WwW • 975-291

Class B extends A
{

```
void meth() // Error Can't override
{
```

```
    System.out.println("Method");
```

)
iii) To prevent a class to be inherited, **final** keyword is used as an class decoration. Declaring a class as **final** implicitly declare that all of its methods are **final**.

Ex:-
final CLASS A

```
void meth()
```

```
System.out.println("Hello");
```

)
CLASS B extends A // Error, Can't subclass A.
{

```
//....
```



Work To Do

Important

Phone

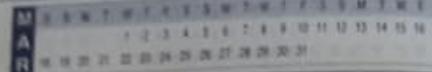
2006

Fri

17

Mar

126-Wins • 877-231

PACKAGE

Package are the containers for classes that are used to keep the classes share compartmentalized. A package is a collection of related classes, interfaces and also exceptions.

Advantages

- provide a convenient means to group the related things.
- Avoid potential naming conflict.

Defining a Package:-

Syntax:- package pkg;

Where package is the key word or command must be mentioned as the first statement in a Java source file. Any class declared within that file will belong to the specified package. The package statement defines the name space in which classes are stored; pkg is the name of the package.

Ex: The following statement creates a package called MyPackage

package MyPackage;

Work To Do

Important

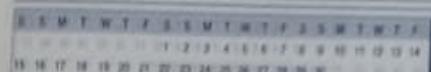
Phone

2006

Sat

18

Mar



APR

Java uses file system directories to store packages. And all the class files for any class declared under MyPackage must be stored in a directory called MyPackage.

Example:-

```
package pack1;
public class Max
{
    int x, y;
    public Max(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
    public int getmax()
    {
        if (x > y)
            return x;
        else
            return y;
    }
}
```

19 Sun

compile this class at C:\>javac -d . Max.java
package pack1;

```
public class Max;
```

```
int x, y;
```

Work To Do

Important

Phone

2006

Mon

20

Mar

13th Week • 279-286

MAR

SUN	MON	TUE	WED	THU	FRI	SAT
		1	2	3	4	5
		6	7	8	9	10
		11	12	13	14	15
		16	17	18	19	20
		21	22	23	24	25
		26	27	28	29	30
						31

public Min (int x, int y)

{
 if (x < y) = x;
 else = y;

public int sum ()

{
 return x + y;

 else

 return y;
}

Compile it as C:\Java\ -d .Min.java

import Paarl.MAX;

import Paarl.MIN;

class MaxMin{}

public static void main (String args [])

{
 MAX m1 = new MAX(10, 20);

 System.out.println(m1.getmax());

 MIN m2 = new MIN(20, 30);

 System.out.println(m2.getmin());