System.in object is known as standard input object.

System.out Object is known as standard output object.

(*) ⇒ java.lang package is implicitly imported into Java program.

/** ... */ : (javadoc comment) :- It enables us to embed program documentation directly in our program. javadoc utility program (part of jdk) reads javadoc comments and uses them to prepare our program's documentation in HTML format.

* Java tool (interpreter) launches JVM, which loads the .class file. The JVM calls the main().

package ⇒ named group of related classes are collectively reffered as Java class library (Java API).

A static method is special, bcz we can call it without creating the object of the class in which it is specified/declared.

* Java is known as extensible language bcz we can declare new class types as needed.

* everything in a java program must be inside a class.

primitive types

primitive ⇒ Appendix

JAVA

- J2SE (Java 2 Standard Edition)
- J2EE (Java 2 Enterprise Edition)
  - ⇓ JSP (Java Server Page)
  - ⇓ To make a dynamic/server web page
- J2ME (Java 2 Mobile Edition)

web-page (A page of website)

- Static
  - ⇓ created with Html (Hyper Text Markup Language)
  - ⇓ understandable by browser
- Dynamic / Server
  - ⇓ created with
  - ASP (Active Server Page)
  - JSP (Java Server Page)
  - PHP (Preprocessor Hypertext) or (Hypertext preprocessor)

JAVA:-

Java is a pure object oriented platform independent programming language which is developed by James Gosling and Patrick Naughton in 1991 in Sun Microsystems. It is mainly designed for electronic goods such as TV, VCRs etc. The initial name of java is "oak" and finally in 1995, it renamed as Java and launch its version Java 1.0. The Sun Microsystem gives WORA (Write Once, Run anywhere) Taglina for java 1.0 version.

History:-

In 1990, Sun Microsystems decided to develop special software that can be used to manipulate consumer electronic devices. A team of Sun MicroSystem programmers headed by James Gosling was formed to undertake this task.

In 1991, after exploring the possibility of using the most popular object oriented language c++, the team announced a new language named "Oak".

In 1992, the team known as Green Project team by Sun, demonstrated the application of their new language to control a list of home appliances using a hand held device with a tiny touch sensitive screen.

In 1993, the World Wide Web (www) appears on the internet & transformed the ten based

07 JANUARY 2008
MONDAY
007-359 • WEEK 02

internet into a graphical-rich environment.

In 1994, the team developed a web browser called "Hot Java" to locate & run applet programs on Internet.

In 1995, "Oak was renamed Java" due to some legal snags.

In 1996, Java established itself as a leader for Internet Programming & also as a general purpose, OOP language.

There are many version that has been released. The version of java are:-

- JDK Alpha and Beta (1995)
- JDK 1.0 (1996)
- JDK 1.1 (1997)
- J2SE 1.2 (1998)
- J2SE 1.3 (2000)
- J2SE 1.4 (2002)
- J2SE 5.0 (2004)
- Java SE 6 (2006)
- Java SE 7 (2011)
- Java SE 8 (2014)

NOTES

JANUARY M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 JANUARY 08
WEEK 02 • 008-358
TUESDAY

* Features / Buzzword of Java:-

Sun Microsystem officially describes java by given Buzzwords & that are as follows:-

1) Compiled & Interpreted:-

Java is both a compiled & an interpreted language. Java translates source code into byte code instructions. As bytecode are not machine instructions. Java interpreter generates machine code that can directly be executed by the particular machine that is running the Java program.

2) Platform independent & Portable:-

The most java programs can be easily moved from one computer to another, anywhere & anytime. Changes & upgrade in OS, processors and system resources will not force any changes in java program.

3) Object Oriented:-

Almost everything in java is an object. All program code & data reside within objects and classes.

4) Robust & Security:-

Java provides many safeguard to ensure reliable code. It has strict compile time and run-time checking for datatypes. Security is used for programming on Internet.

FEBRUARY

Threat of viruses and abuse of resources are everywhere. The absence of pointers in java ensures that programs can't gain access to memory location without proper authorisation.

5) Distributed :-

It is designed as a distributed language for creating application on network. It has ability to share both data and programs.

6) Simple & small :-

Java is a simple & small language. Many features of C & C++ that are either redundant or sources of unreliable code are not part of java.

For ex :- java doesn't use pointers, preprocessor header files, goto statements & many others.

7) Multi-threaded :-

Multi-threaded means handling multiple task simultaneously. It means, we need not wait for the application to finish one task before beginning another.

For ex :- listening sound clip while browsing a page at a time.

NOTES

8) High Performance :-

Use of intermediate bytecode and multi-threading enhances the overall execution speed of Java programs.

9) Dynamic :-

Java is capable of dynamically linking in new class libraries, methods and objects. It can also the determine the type of class through a query, making it possible to either dynamically link or abort the program, depending on the response.

TYPES OF JAVA :-

There are two types of java programs :-

A) Application Program :-

Application programs are stand-alone programs that are written to carry out certain tasks on local computer such as solving equation, reading & writing files etc. The application program can be executed using two steps :-

(i) Compile source code to generate Bytecode using javac compiler.
(ii) Execute the bytecode program using java interpreter.

B) Applet Program :-

NOTES

Applets are small java programs developed for Internet applications. An applet located in distant computer can be downloaded via internet & executed on a local computer.

using Java capable browser. The Java applets can also be executed in the command line using applet-viewer, which is the part of JDK.

Differences b/w Java and C++:-

1. Java is a pure object oriented language while C++ is basically C with object oriented extension.
2. Java doesn't support ~~oopoo~~ operator overloading.
3. Java doesn't have template classes as in C++.
4. Java doesn't support multiple inheritance of classes.
5. Java doesn't support global variable.
6. Java doesn't use pointers.
7. Java has replaced the destructor function with a finalise() function.
8. There are no header files in java.
9. Java has no structures or unions.
10. Java has no individual functions.

Bytecode:-

The java compiler compiles the code for a machine that physically doesn't exist, i.e. for a virtual machine. This compiled code is known as bytecode, and hypothetical machine is called Java Virtual Machine (JVM).

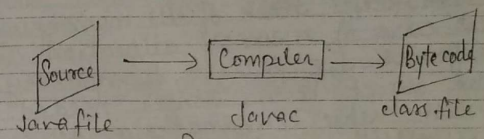


Fig:- Process of Compilation

Java Virtual Machine (JVM):-

Java language is a platform independent, usually referred to as "write once run anywhere (WORA). This is accomplished by the JVM that runs on the local machine, interprets the Java bytecode, and converts it into platform-specific machine code. The JVM is invoked differently, depending on the types of Java program. JVM performs the following functions:-

1. When a class file is executed, JVM loads all required classes automatically from the local disk or from across the network. This is the function of "class loader" utility of JVM.

(2) After loading the required classes, JVM verifies to make sure that the classes do not violate any of the basic rules of Java language. This is the function of "Bytecode verifier".

(3) Then JVM keeps track of all memory usage. It takes care of memory allocation and also performs the release of memory after the object is no longer needed. This process which manages dereferenced objects is called Garbage Collection.

Note:-

Within JVM, Just-In-time (JIT) compilers are used to improve performance. JIT compiler translates bytecode only the first time. If repeated execution of the code is required, it is automatically mapped to the corresponding native machine code. This is especially effective in ~~reiteration~~ repetative code, such as loops or recursive functions.

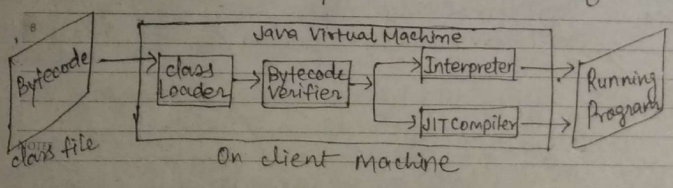


Fig:- The Java Runtime System

Java Development Kit (JDK):-

Java environment includes a large number of development tools & hundreds of classes & methods. The development tools are part of the system known as Java Development Kit (JDK).

The major part of it comprises of Software Development Kit (SDK). Java 2SDK 1.4 includes the following sets of tools:-

- Basic Tools (javac, java, javadoc, appletviewer, jar, jdb, javah, javap, extcheck)
- Remote Method Invocation (RMI) Tools (rmic, rmiregistry, rmid, serialver)
- Internationalisation Tools (native2ascii)
- Security Tools (keytool, jarsinger, policytool)
- Java IDL & RMI-IIOP Tools (tnameserv, idlj, orbd, servertool)
- Java Plug-in TM Tools.

NOTES

16 JANUARY 2008
WEDNESDAY
016-350 • WEEK 03

Application / Uses of Java:-

A/c to Sun, 3 billion devices run java. There are many devices where java is currently used. Some of them are as follows:-

1) Desktop application such as acrobat reader, media player, anti-virus, etc.

2) Web Applications such as irct.co.in, etc.

3) Enterprise Applications such as banking applications.

4) Mobile
5) Embedded System
6) Smart Card
7) Robotis
8) Games, etc.

Types of Java Applications:-

There are mainly four types of applications that can be created using java programming:-

1) Stand-alone Application:-

NOTES It is also known as desktop application or window-based application. An application that we need to install on every machine such as media-player, antivirus, etc. AWT and Swing are used in java for creating stand alone applications.

2) Web Application:-

An application that runs on the server side and creates dynamic page, is called web application. ~~Currently, servlet~~ Currently, servlet, jsp, Struts, jsf, etc. technologies are used for creating web applications in java.

3) Enterprise Application:-

An application that is distributed in nature, such as banking applications, etc. It has the advantage of high level security, load balancing & clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application:-

An application that ~~is~~ is created for mobile devices. Currently, android and javel are used for creating mobile applications.

NOTES

Character set of JAVA:-

The valid / recognised set of characters used in java is known as character set of Java. character set can contain maximum 2^{16} (65536) characters, bcz Java follows unicode.

There are following characters used in character set of Java:-

1. Capital letters (A – Z)
2. Small letters (a – z)
3. Digits (0 – 9)
4. Special symbols (+, –, *, /, $, etc)
5. White space
   - → Blank space
   - → tab
   - → ~~new line~~ Carriage return
   - → Line feed / new line
   - → form feed

NOTES

Keyword ~~words~~ context
backslash
Separator
Comment
Documentation
import
assign
unit
static
compile time

25 JANUARY FRIDAY 2008 025-341 • WEEK 04

20/10/14

TOKEN :- → definition c++ ke copy Be with Lava.

- Keyword
- Identifier
- Literal
- Operator
- Separator
- ~~Punctuator~~

① Keyword :-

Keywords are reserved words that are pre-defined in Java library. They cannot be changed or modified.

There are total 50 keywords in Java & like C & C++, keywords are written in small case letter.

C →	int	char	float	double
	void	signed	unsigned	short
	long	if	else	switch
	case	default	break	continue
	while	do	for	goto
	return	auto	static	extern
	register	struct	union	enum
	constant	volatile	typedef	sizeof()
C++ →	new	delete	inline	class
	private	protected	public	friend
	operator	virtual	try	catch
	throw	template	end	this

JANUARY 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 JANUARY SATURDAY 26 WEEK 04 • 026-340

② Identifier :-

Identifier is the name of variable, label, array, function, class, interface or any user defined data item.

There are certain rules regarding identifier :-

(i) Identifier must be meaningful & unique.

(ii) It is the combination of alphabets letters capital as well as small, digits (0-9), underscore (_) or dollar ($) sign.

(iii) First character of identifier must be an alphabet or underscore or $ sign. It means digits can't be used as first character of identifier.

(iv) No special symbols can be used except underscore (_) and dollar ($).

(v) white Spaces is not allowed.

(vi) It may be 31 or 38 character long but first 8 characters are significant & we have to use short & simple name.

(vii) Pre-defined keywords, functions or we can say pre-defined terms cannot be used as a name/identifier.

(viii) With JDK 8, the use of an underscore by itself as an identifier is not recommended.

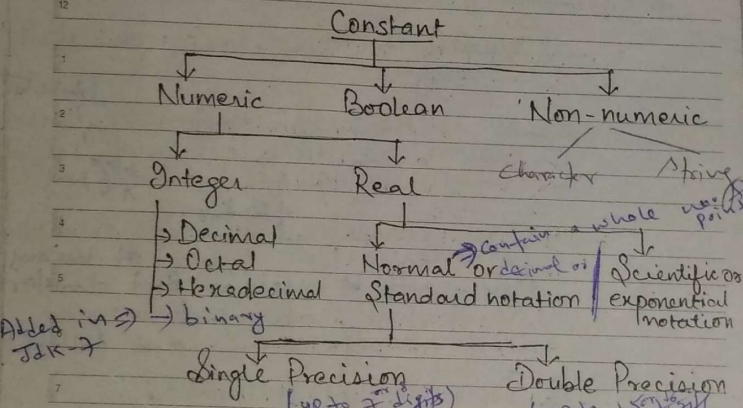
28 JANUARY MONDAY 2008 028-338 • WEEK 05

Mantissa :- It is either a real no. or an integer. Its exponent can have + or minus (-) sign. e/E is used to indicate exponent.

Ex:- 12e-2
1.5e+5
2.19f3

③ Literal / Constant :-

A value which doesn't change during execution of program is called constant or we can say constant is a fixed value during execution.



1. Numeric Constant :-

A constant that contains number or works on numbers is called numeric constant.

There are two types of numeric constant :-

(a) Integer
(b) Real

Real no. expressed in decimal notation is an integer with an optional plus (+) or minus (-) sign followed by a decimal point, which is an integer. Its exponent is used to indicate exponent.

Ex:- 12e-2
1.5e+5
2.19f3

29 JANUARY TUESDAY 2008 WEEK 05 • 029-337

(a) Integer Constant :-

A constant which has no fractional part is called integer constant. Any whole number value is an integer literal.

Integer constant is of four types :-

(i) Decimal constant
(ii) Octal constant
(iii) Hexadecimal constant
(iv) binary constant (Added in JDK-7)

(i) Decimal Constant :-

It is combination of digits from 0 to 9 with an optional leading +ve or -ve sign.

Ex:- +25, 129, -225, etc.

(ii) Octal Constant :-

It is combination of digits from 0 to 7, & preceeded with zero (0).

Ex:- 025, 057, 0225, etc.

(iii) Hexadecimal constant :-

It contains digits from 0 to 9 & letters A to F & also preceeded with zero x (0x).

Ex:- 0x25, 0x129, etc.

Ex:- 45 = 101101
129 = 10000001
111 = 1101111

(iv) binary constant

It contains bits 0, 1 and leaded/preceded with 0b/0B (zero b/B). Binary constant is added in JDK-7.

Ex:- 0b1010, 0B1111 etc.

Note:- Beginning with JDK-7, we can embed one or more underscores (_) in integer/fract/exponent to read large integer literals. During compilation, underscore is discarded.

(b) Real Constant:/ floating point

A constant that contain fractional part.

Real constant is of two types:-

(i) Single Precision (float)
(ii) Double Precision (double)

(i) Single Precision:-

It contains upto ~~six~~ 7 or 8 digits after decimal points.

Ex:- 10.523f, 227.342156F, etc
Ex:- 3.1415f.

(ii) Double Precision:-

It can contain ~~more than~~ upto 15 or 16 digits ~~six digits~~ after decimal point.

Ex:- 203.24361273, 403.456D, 32.360, etc

Note:- ⇒ By default, real constant/floating point constant in standard notation is considered as double (double precision) value. To make it float (single precision) value, we use letter f/F as suffix in constant. However, we can also use d/D optional letter for double constant.

(Margin note:) Standard notation consists of a whole no. followed by a decimal point followed by fractional value.

2. Boolean:-

The constant which works on only two values namely true and false is known as boolean constant. ~~Unlike C and C++, in Java, the values one (1) & zero (0) is represented as true and false respectively, and considered as Boolean constant.~~

The values true and false do not convert into any numeric representation. The true and false in Java does not equal to 1 and 0 respectively.

Back slash Character Constant:-

Ex:- '\ddd' → for octal Ex:- '\141' (→ 'a')

'\udddd' → for Hexa Ex:- '\u0061' (→ 'a') (ISO-latin-1 [illegible])

'\\' → For backslash

'\'' → for single quote

'\"' → for double quote

'\n' → for new line

'\t' → for tab

'\b' → for backspace

'\r' → for carriage return

'\f' → for form feed

3. Non-Numeric Constant:-

A constant which is not related to numbers is called non-numeric constant.

There are two types of non-numeric constant:-

(a) Character constant
(b) String constant

(a) Character Constant:-

It contains a single character surrounded with single quote. Characters in Java follows Unicode character set which

01 FEBRUARY FRIDAY 2008

is 16 bit code, while c & c++ uses ASCII character set which is 7 bit or 8 bit code.

032-334 • WEEK 05

Ex:- 'A', 'S', '*', etc.

(b) String Constant:- It contains one or more characters enclosed within double quote. In c & c++, string are array of characters, while in Java, it is treated as Objects.

Ex:- "A", "abc", "Diksha", "*-*/", etc

④ Operators:-

Operators are symbols that perform operations to their operands.

There are basically three types of operator a/c to the no. of operands found in one expression. An expression is a combination of operator & operand.

(a) Unary opt.

(b) Binary Opt.

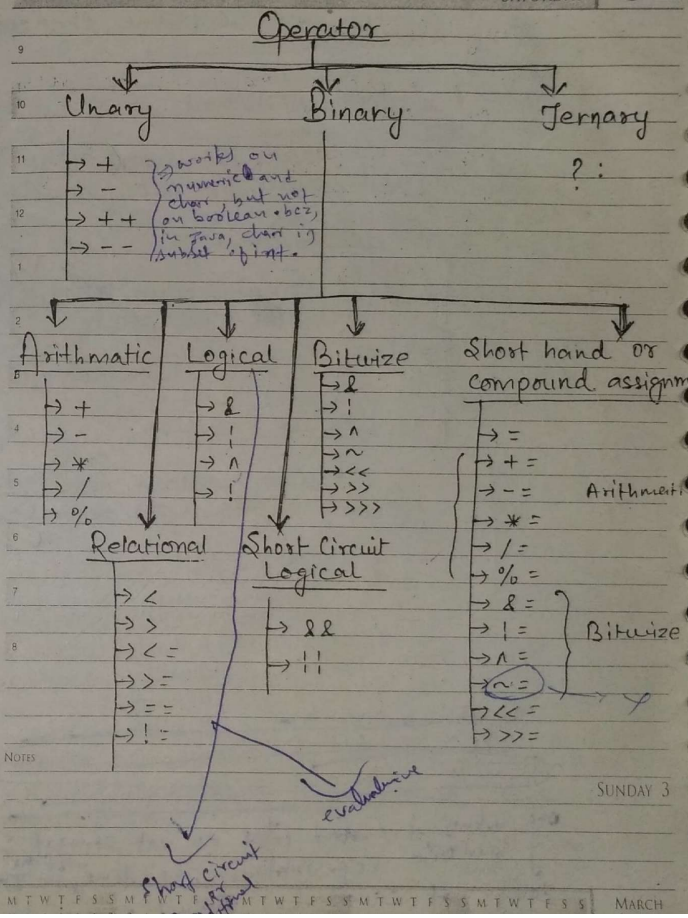
(c) Ternary / Conditional Opt.

NOTES

FEBRUARY 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 FEBRUARY 02 SATURDAY

WEEK 05 • 033-333



NOTES

SUNDAY 3

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S MARCH

Scanned by CamScanner

04 FEBRUARY MONDAY 2008

035-331 • WEEK 06

(a) Unary Operator :-

This operand performs operation on one operand.

There are following types of unary operator :-

(i) + (Unary Plus)
(ii) - (Unary Minus)
(iii) ++ (Increment opt.)
(iv) -- (Decrement opt.)
(v) ! (Unary higation)
(vi) ~ (Tild / complement opt.)

(b) Binary Operator :-

The operator which perform operation on two operands.

Binary operators are classified in :-

(i) Arithmatic opt.
(ii) Relational opt.
(iii) Logical opt [Evaluating]
(iv) Short circuit Logical
(v) Bitwise opt.
(vi) Short-hand / Compound assignment

NOTES

FEBRUARY 2008

2008

WEEK 06 • 036-330

FEBRUARY TUESDAY 05

(i) Arithmatic operator :-

It works on numeric type (byte, short, int, long, float, double) and char type operands.

works on numeric and char type.

+ (For addition)
- (For subtraction)
* (For multiplication)
/ (For quotient)
% (For remainder)

(ii) Relational Operator :-

It returns the result in the form of true and false.
This relational opt. are as follows :

It takes operands of numeric types and char.

→ <
→ >
→ <=
→ >=
→ == (equality)
→ != (incaulity)

can't work on boolean values (true, false) bcz, boolean values are not ordered.

numeric, char and boolean

Ex :- x = 15, y = 10

x < y → False
x > y → True
x <= 15 → True
x >= y → True
x == y → False
x != y → True

true == true → true

NOTES

MARCH 2008

logical opt :-

(iii) Logical ^evaluafice Operator :-

- boolean logical AND → &
- boolean logical OR → |
- boolean logical exclusive OR → ^
- logical negation / logical NOT / logical complement → !

} → will evaluate each operands.

Ex :- x = 10 , y = 15 , z = 12

i) x < y & y > z
   True & True → True

ii) x > y & y > z
   False & True → False

iii) x < y | y > z
   T | T → True

iv) x > y | y > z
   F | T → True

v) x < y ^ y > z
   T ^ T → False

vi) x > y ^ y > z
   F ^ T → True

evaluates the second operand only when necessary.
Java will not evaluate the right hand operand when the outcome/result is determined by left operand.

(iv) Short Circuit Logical :- or conditional

- → && (conditional AND)
- → || (conditional OR)

Ex :- x = 10 , y = 15 , z = 12

i) x < y && y > z
   T , T → True

ii) x > y && y > z
   F → False

iii) x < y || y > z
   T → True

iv) x > y || y > z
   F , T → True

v) x > y ^ y < z
   F , F → False

Note :- ~~The operator that returns the result in form of true or false is known as Boolean operator.~~

08 FEBRUARY FRIDAY 2008
039-327 • WEEK 06

(V) Bitwise operator :- (integer types)

- ~ (Bitwise complement)
- & (Bitwise AND)
- | (Bitwise OR)
- ^ (Bitwise X-OR)
- << (Left shift opt.)
- >> (Right shift opt. / signed right shift)
- >>> (Unsigned Right shift / Right shift with fill zero)

Ex:- x = 25, y = -25, z = 10, s = 13

i) x & z = ?

```
00000000000000000000000000011001 → 25
00000000000000000000000000001010 → 10
00000000000000000000000000001000 ⇒ 8
```

ii) y & s = ?

```
11111111111111111111111111100111 → -25
00000000000000000000000000001101 → 13
00000000000000000000000000000101 ⇒ 5
```

iii) x | z = ?

```
00000000000000000000000000011011 → 27
```

iv) y | s = ?

```
11111111111111111111111111101111
        ↓ 2's comp
00000000000000000000000000010001 ⇒ -17
```

2008 FEBRUARY 09 SATURDAY
WEEK 06 • 040-326

(guid char), not on boolean, float and double

v) x ^ z = ?

```
00000000000000000000000000011001
00000000000000000000000000001010
00000000000000000000000000010011 → 19
```

vi) ~x = ?

```
00000000000000000000000000011001
        ↓
11111111111111111111111111100110
        ↓ 2's comp
00000000000000000000000000011010 ⇒ -26
```

vii) ~y = ?

```
11111111111111111111111111100111
        ↓
00000000000000000000000000011000 → 24
```

viii) x<<1 = ?

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 → x
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 → 50
```

ix) y<<1 = ?

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 → y
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0
        ↓ 2's comp
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 → -50
```

* extended 8 bit character set is ISO-Latin-1, ranging from 0 to 255.
* char is an unsigned 16 bit ranging from 0 to 65535. The standard set of unicode.
* ASCII ranges from 0 to 127. We can perform arithmetic operation on char.

primitive / simple / elemental data type
↳ primitive types are not objects.
↳ Java specifies a range and behavior for each primitive type.

7-bit ASCII character set is a subset

x) $x>>1=?$

0000000000000000000000000011001 → x

0000000000000000000000000001100 → 12

xi) $y>>2=?$

11111111111111111111111111100111 → y

00111111111111111111111111111001

⇓ 2's comp

11000000000000000000000000000111 ⇒ −7

xii) $x>>>1$

0000000000000011001 → x = 25

A/c to MSB → 0 000000000000001100 → x>>1

Always zero → 0 0000000000001100 → x>>>1

xiii) $y>>>1$

0000000000000011001 → x = 25

1111111111111100111 ←(2's) x = −25

A/c to MSB → 1111111111111110011 → x>>1

MSB always zero → 0111111111111110011 → x>>>1

(vi) Ternary Operator:-

This operator is also known Conditional Operator.

The ternary opt. are ? :

⑤ Separator:-

- ( ) ⇒ Parenthesis
- { } ⇒ braces
- [ ] ⇒ Brackets
- , ⇒ Comma
- ; ⇒ semicolon
- . ⇒ period
- colons (::) :- used to create a method or constructor reference. (Added by Jdk 8)

int r = n + y;
↑ dynamic initialize

Each variable has a name, a type, a size (in bytes) and a value.

Note:- Java is strongly typed language. All operations are type-checked by the compiler for type compatibility. Illegal operation will not be compiled. This strong type checking helps to prevent error and enhance reliability.

03 MARCH 2008
MONDAY
063-303 • WEEK 10

Q.④ WAP to print the cube of a number.

Prog:-
```
//prog to print the cube of a number
import java.util.*;
class Cube
{
    public static void main(String args[])
    {
        int x, c;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a no.");
        x = s.nextInt();
        c = x*x*x;
        System.out.println("Cube of no. is "+c);
    }
}
```
(args: an array of objects of type String)

Q.⑤ WAP to print the area of triangle.

Prog:-
```
//prog to print the area of triangle
import java.util.*;
class Area
{
    public static void main(String args[])
    {
        int b, h;
        float a;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter base & height");
        b = s.nextInt();
        h = s.nextInt();
        a = 1/2 * b * h;
        System.out.println("Area = "+a)
    }
}
```

NOTES

Data can come from several sources, such as the user at the keyboard, or a file on disk etc. Here, System.in specify the source of data, which is keyboard. It typed by user and translates these bytes into appropriate types.

2008 WEEK 10 • 064-302 MARCH TUESDAY 04

Q.⑥ WAP to print the circumference of circle.

Prog:-
```
//prog to print the circumference of circle
import java.util.*;
class Circum
{
    public static void main(String args[])
    {
        int r;
        float c;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter radius");
        r = s.nextInt();
        c = 2*3.14*r;
        System.out.println("Circumference="+c);
    }
}
```
use math.PI

NOTES

05 MARCH WEDNESDAY 18|11|14 2008

065-301 • WEEK 10

To compile a Java Program:-

Syntax:- javac filename.java↵

Ex:- javac Demo.java↵

To execute the program of Java:-

Syntax:- java filename↵

Ex:- java Demo↵

Note:- The extension of java program file is .java, while the C++ is .cpp and the C is .c.

The phases involved / included in the execution of java program is as follows:-

- Creation of a new java program file
- Compiling java program file
- Executing java program file.

Creation of a new java program file:-

First of all, we have to create a file having extension: java. To do this, we have to type the following on command prompt:-

2008 MARCH THURSDAY 06

WEEK 10 • 066-300

C:\> notepad filename.java↵

Prompt — changeable

notepad is a editor of windows, while a editor named edit is editor of DOS, while Vi editor is editor of unix. Editor is just like your notebook, where you can write, modify, delete, etc with the program.

Ex:- C:\> notepad Dem.java↵

Note:- The name of java program file will be same as that class name which contains the function/method main.

Compiling Java Program file:-

After saving the program file of java, we perform compilation operation with the help of javac compiler of java & the syntax of compilation is

Syntax:- javac filename.java↵

Ex:- javac Demo.java↵

Executing Java Program file:-

The execution of java program is performed using java and the syntax is:-

Syntax:- java filename↵

Path setting in Java:-

07 MARCH FRIDAY

There is no need to set the path in Java, if the file/source file is saved within JDK/bin directory, because the tools such as javac, java etc. available in the current directory. But, this is not possible for Java source file outside the JDK/bin folder.

To set the path for java:-

After copying the path, follow the following steps:-

1. Right click on MY COMPUTER and choose PROPERTIES.
2. From system property window, choose ADVANCED SYSTEM SETTINGS from left pane/area.
3. From Advanced tab, choose ENVIRONMENT VARIABLES
4. From environment variables window, choose PATH variable from system variable section and finally click on EDIT tab. (or click on new tab of user variables and write Path in variable name and paste the path in variable value.)
5. Paste the copied path at the beginning of VARIABLE VALUE section. → OK.

Note:-

We have to terminate your path using semicolon (;), because each path is separated with semicolon.

To set the temporary path in Java

1. Open Command prompt.
2. Type the command set path = copied path

Command name

Ex:- C:\> set path = C:\Program Files\Java\jdk1.6.0\bin

Other are two ways to set the path in Java:-
1. temporary
2. Permanant

19 | 11 | 14
MARCH SATURDAY 08

In short, to execute program of Java:-

C:\> notepad filename.java ↵

C:\> javac filename.java ↵

C:\> java filename ↵

Note:-

Like Turboc2, Dev.C of C and turboC3, visual studio C++ of C++. In Java, there are several editor exist where we can write, compile, modify and execute the program.

These popular editors are as follows:-

1. Jedit
2. Jcreator
3. BlueJ
4. Netbens, etc.
5. eclipse

Some important points regarding java like naming conventions such as different filename from class name, different formats of main()

SUNDAY 9

Naming Convention in Java

Camel Case

Scanned by CamScanner

note with programs:
1. Uninitialized local variable produces error.
2. initialization of value into a variable
3. Dynamic initialization
4. error when Path not set.
5. error when filename and classname not matched.
6. we can give different name of both.

21/11/14

// Example of identifiers

```
import java.util.Scanner;
class Ddd
{
    public static void main(String args[])
    {
        int $first, _second, r;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter two nos.");
        $first = s.nextInt();
        _second = s.nextInt();
        r = ($first + _second)/2;
        System.out.println("Average is "+r);
    }
```

Path not set:
Error
↓
'javac' is not recognized as an internal or external command, operable program or batch file.

// Example of constant

also take an example of backslash character constant, octal & hexa integers.

```
import java.util.Scanner;
class Ddd
{
    public static void main(String args[])
    {
        byte a = 125;   // error
        short b = 25000;   // error
        int c = 123456;
        long d = 4664645;
        long e = 9223372036854775804L;
```



```
        char x = 'A';
        float m = 12.34f;   // error
        double n = 12345.535;
        boolean y = true;
        String s = "shailesh";
        System.out.println("a is "+a);
        System.out.println("b is "+b);
        System.out.println("c is "+c);
        System.out.println("d is "+d);
        System.out.println("e is "+e);
        System.out.println("x is "+x);
        System.out.println("m is "+m);
        System.out.println("n is "+n);
        System.out.println("y is "+y);
        System.out.println("s is "+s);
    }
}
```

notepad Ram.java

```
class Shyam
{
    public static void main(String args[])
    {
        System.out.println("welcome to java");
    }
}
```

Error: Could not find or load main class Ram.

// Example of Relational Operators

```
import java.util.Scanner;
class RO
{
    public static void main(String args[])
    {
        int x = 50, y = 160, z = 50;
        System.out.println(x < y);
        System.out.println(x > y);
        System.out.println(x <= z);
        System.out.println(x >= y);
        System.out.println(x == y);
        System.out.println(x != y);
        System.out.println(x == z);
    }
```

Output
True
False
True
F
F
T
T

also give the example of float point, exponential and double precision.

12 MARCH 2008
WEDNESDAY
072-294 • WEEK 11

```
// Example of Evaluative Logical Opt.
import java.util.Scanner;
class DS
{
    public static void main(String args[])
    {
        int x=50, y=160, z=50;                    Output
        System.out.println(x<y & y>z);            T
        System.out.println(x>y & y>z);            F
        System.out.println(x<y | y>z);            T
        System.out.println(x>y | y<z);            F
        System.out.println(x>y ^ y>z);            T
        System.out.println(x<y ^ y>z);            F
        System.out.println(!(x<y));               F
    }
}
```

```
// example of uninitialized local variable
class Demo
{
    main()
    {
        int x;
        sopln(x);
    }
}
```

output:
Variable x might not have been initialized

```
// initialized
// of
class Demo
{
    main()
    {
        byte x=120;
        short y=32769;
        sopln(x);
    }
}
```

possible loss of precision
required: byte found: int

2008 MARCH 13
WEEK 11 • 073-293
THURSDAY

```
// Example of Bitwise operator
import java.util.Scanner;
class DS
{
    public static void main(String args[])
    {
        int x=27, y=19, z=13, n=20, m=25;
        System.out.println(x&y);          Output
        System.out.println(x|y);          19
        System.out.println(x^y);          27
        System.out.println(~z);           8
        System.out.println(n<<1);         -14
        System.out.println(n<<2);         40
        System.out.println(n>>1);         80
        System.out.println(n>>>2);        10
        System.out.println(m>>>1);        5
        m=-25;                            12
        System.out.println(m>>>1);        2147483635
    }
}
```

```
// overflow error
class Demo
{
    main()
    {
        byte x=127+5;
        short y=32767+9;
        sopln(x);
    }
}
```

possible loss of precision
required: byte found: int
required: short found: int

```
//Example of Short Circuit Logical
import java.util.Scanner;
class SCL
{
    public static void main(String args[])
    {
        int x=10, y=15, z=12;
        System.out.println(x<y && y>z);   T
        System.out.println(x>y && y>z);   F
        System.out.println(x<y || y>z);   T
        System.out.println(x>y || y>z);   T
        System.out.println(x>y ^ y<z);    F
    }
}
```

Output

← Take the example of arithmetic on characters.

Operator Precedence :-

The process of specifying the way of execution of operators in an expression is considered as precedence.

Operator Precedence is shown below from higher value to lowest value:-

1. ( )
2. ++, --, ~, !
3. *, /, %
4. +, -
5. >, >=, <=, ==, !==

Q. What will be the value of X and Y after execution of following codes?

```
Y = 5;  X = ++Y + 4 * Y;
```

Soln:-

```
X = ++5 + 4 * 5
X = 6 + 4 * 5
X = 6 + 20
X = 26   Ans
```

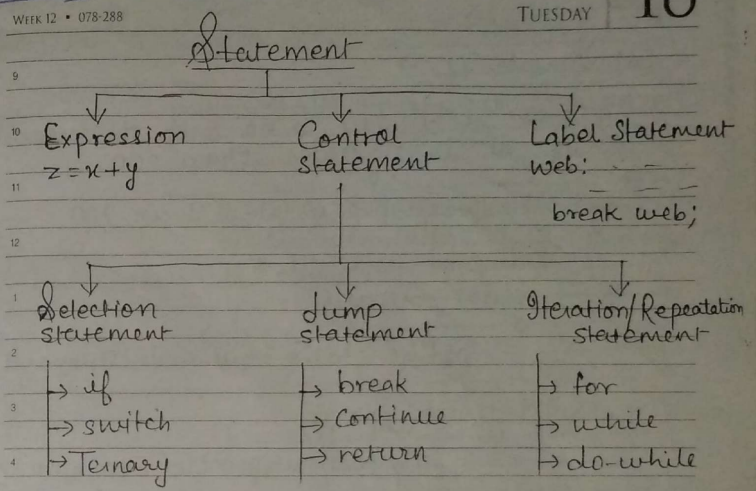
17 MARCH MONDAY 2008 077-289 WEEK 12

we can also refer Scanner class as java.util.Scanner, if we not import this full package name and class name.
java.util Package. it includes only this is known as fully qualified class name.

modifier (from Thang)

scope
Typecasting

18 MARCH TUESDAY 2008 WEEK 12 • 078-288



SELECTION STATEMENT :-

'if' statement :-

'if' statement are categorised in four :-

(a) 'simple if' statement
(b) 'if else' statement
(c) 'Nested if' statement
(d) 'else if ladder' statement

NOTES

(a) Simple if :-

(1) WAP to display a no. onto the screen if it is greater than 200.

Prog:-
```
// display nos. greater than 200
import java.util.*;
class Great
{
    public static void main(String args[])
    {
        int n;
        System.out.println("Enter a no.");
        Scanner s = new Scanner(System.in);
        n = s.nextInt();
        if(n>200)
        {
            System.out.println(n);
        }
    }
}
```

(2) WAP to print the perimeter of rectangle if length is less than its breadth.

Prog:-
```
// print the perimeter of rectangle if
// l<b
import java.util.*;
class Peri
```



```
{
    public static void main(String args[])
    {
        int l, b, p;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter l and b");
        l = s.nextInt();
        b = s.nextInt();
        if(l<b)
        {
            p = 2*(l+b);
            System.out.println(p);
        }
    }
}
```

(3) WAP to print the area of circle if radius<10

Prog:-
```
//print the area of circle if radius<10.
import java.util.*;
class Area
{
    public static void main(String args[])
    {
        float r, a;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter radius");
        r = s.nextFloat();
        if(r<10)
        {
            a = 3.14*r*r;
            System.out.println(a);
        }
    }
}
```

(b) 'if else' :-

① WAP to check the given no. is odd or even.

Prog:-
```
//prog to check no. is odd or even
import java.util.*;
class OE
{
    public static void main(String args[])
    {
        int x;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a no.");
        x = s.nextInt();
        if(x%2==0)
            System.out.println("even is"+x);
        else
            System.out.println("Odd is"+x);
    }
}
```

② WAP to print the product of two nos. if first no. > second no, otherwise print the modulus of both nos.

Prog:-
```
//prog to print the product or modulus
// of two nos.
import java.util.*;
```



```
class Pro
{
    public static void main(String args[])
    {
        int x,y,z;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter two nos.");
        x = s.nextInt();
        y = s.nextInt();
        if(x>y)
        {
            z = x*y;
            System.out.println("Product is"+z);
        }
        else
        {
            z = x%y;
            System.out.println("Modulus is"+z);
        }
    }
}
```

③ WAP to print the cube of a given no. if the no. is less than 10, otherwise print the square of that no.

Prog:-
```
//prog to print the cube or square of a no.
import java.util.*;
class CS
{
    public static void main(String args[])
```

24 MARCH 2008
MONDAY
084-282 • WEEK 13

```
    {
        int x, c;
        Scanner s=new Scanner(System.in);
        System.out.println("Enter a no.");
        x = s.nextInt();
        if(x<10)
        {
            c= x*x*x;
            System.out.println("Cube is"+c);
        }
        else
        {
            c = x*x;
            System.out.println("Square is"+c);
        }
    }
}
```

NOTES

MARCH 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S
31 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

2008 MARCH 25
WEEK 13 • 085-281 TUESDAY

(c) 'Nested if':-

① WAP to print the greatest no. among three nos.

Prog:-

```
//prog to print the greatest no. among
//three nos.
import java.util.*;
class Great
{
    public static void main(String args[])
    {
        int x,y,z;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter three nos.");
        x = s.nextInt();
        y = s.nextInt();
        z = s.nextInt();
        if(x>y)
        {
            if(x>z)
                System.out.println("Greatest="+x);
            else
                System.out.println("Greatest="+z);
        }
        else
        {
            if(y>z)
                System.out.println("Greatest="+y);
            else
                System.out.println("Greatest="+z);
        }
```

NOTES

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

(2) WAP to print the smallest no. among three nos.

Prog:- //prog to print the smallest no. among 3 nos.

```
import java.util.*;
class Small
{
    public static void main(String args[])
    {
        int x, y, z;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter 3 nos.");
        x = s.nextInt();
        y = s.nextInt();
        z = s.nextInt();
        if(x<y)
        {
            if(x<z)
                System.out.println("Smallest="+x);
            else
                System.out.println("Smallest="+z);
        }
        else
        {
            if(y<z)
                System.out.println("Smallest="+y);
            else
                System.out.println("Smallest="+z);
        }
    }
}
```


(3) WAP to calculate amount, discount & net_amount after taking rate & quantity from keyboard on following condition.

Item no.	Amount	Discount
1	>25000	25%
1	<=25000	15%
Any other item	—	10%

Prog:- //prog to calculate amount, discount & net_amount

```
import java.util.*;
class Demo
{
    public static void main(String args[])
    {
        int quan, item;
        float rate, amt, disc, netamt;
        Scanner s = new Scanner(System.in);
        quan = s.nextInt();
        item = s.nextInt();
        rate = s.nextfloat();
        amt = s.nextfloat();
        disc = s.nextfloat();
        netamt = s.nextfloat();
        amt = rate * quan;
        if(item==1)
        {
            if(amt>25000)
```

```
        disc = amt * 25/100;
    else
        disc = amt * 15/100;
}
else
{
    disc = amt * 10/100;
}
netamt = amt - disc;
System.out.println("Amount is " +amt);
System.out.println("Discount is " + disc);
System.out.println("netamt is " + netamt);
}
}
```


(d) 'else if Ladder' :-

① WAP to display the name of month after taking month no. from keyboard.

Prog:-

```
// prog to display the name of month
import java.util.*;
class Month
{
    public static void main(String args[])
    {
        int x;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter month no.");
        x = s.nextInt();
        if(x==1)
            System.out.println("January");
        else if(x==2)
            System.out.println("February");
        else if(x==3)
            System.out.println("March");
        else if(x==4)
            System.out.println("April");
        else if(x==5)
            System.out.println("May");
        else if(x==6)
            System.out.println("June");
```

```
else if(x==7)
    System.out.println("July");
else if(x==8)
    System.out.println("August");
else if(x==9)
    System.out.println("September");
else if(x==10)
    System.out.println("October");
else if(x==11)
    System.out.println("November");
else if(x==12)
    System.out.println("December");
else
    System.out.println("Invalid month no.");
}
}
```

NOTES

② WAP to print the no. of days of a particular month after taking a month no. from keyboard.

Prog:-

```
//prog to print the no. of days of a particular month
import java.util.*;
class Days
{
  public static void main(String args[])
  {
    int mno;
    Scanner s = new Scanner(System.in);
    System.out.println("Enter month no.");
    mno = s.nextInt();
    if(mno==1||mno==3||mno==5||
       mno==7||mno==8||mno==10||mno==12)
      System.out.println("31 days");
    else if(mno==4||mno==6||mno==9||mno==11)
      System.out.println("30 days");
    else if(mno==2)
      System.out.println("28 or 29 days");
    else
      System.out.println("Invalid month no.");
  }
}
```

NOTES

02 APRIL WEDNESDAY 2008

093-273 • WEEK 14

③ WAP to print the greatest among three nos. using 'else if ladder'.

Prog:-
```
//prog to print the greatest no.
import java.util.*;
class Great
{
    public static void main(String args[])
    {
        int x,y,z;
        Scanner s=new Scanner(System.in);
        System.out.println("Enter 3 nos.");
        x=s.nextInt();
        y=s.nextInt();
        z=s.nextInt();
        if(x>y && x>z)
            System.out.println("Greatest is"+x);
        else if(y>z)
            System.out.println("Greatest is"+y);
        else
            System.out.println("Greatest is"+z);
    }
}
```

2008

WEEK 14 • 094-272

APRIL THURSDAY 03

'Switch' Statement:-

① WAP to check the character is vowel or consonent.

Prog:-
```
//prog to check the character
import java.util.*;
class character
{
    public static void main(String args[])
    {
        char x;
        Scanner s=new Scanner(System.in);
        System.out.println("Enter a character");
        x=s.nextchar();
        switch(x)
        {
            case 'A':
            case 'a':
            case 'E':
            case 'e':
            case 'I':
            case 'i':
            case 'O':
            case 'o':
            case 'U':
            case 'u': System.out.println("vowel");
                      break;
            default: System.out.println("Consonent");
        }
    }
}
```

Scanned by CamScanner

(2) WAP to display the name of days of a week after taking a no. from keyboard

Prog: //using switch case

```
import java.util.*;
class Name
{
    public static void main(String args[])
    {
        int num;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a no.");
        num = s.nextInt();
        switch(num)
        {
            case 1: System.out.println("Sunday");
                    break;

            case 2: System.out.println("Monday");
                    break;

            case 3: System.out.println("Tuesday");
                    break;

            case 4: System.out.println("Wednesday");
                    break;

            case 5: System.out.println("Thursday");
                    break;
```



```
            case 6: System.out.println("Friday");
                    break;

            case 7: System.out.println("Saturday");
                    break;

            default: System.out.println("Invalid no");
        }
    }
}
```

(3) WAP to display the no. of days of a particular month after taking month no. from keyboard using switch case.

Prog:- // display no. of days using switch case

```
import java.util.*;
class Month
{
    public static void main(String args[])
    {
        int num;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a month no.");
        num = s.nextInt();
        switch(num)
        {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
```

07 APRIL MONDAY 2008

```
case 10:
case 12: System.out.println("31 days");
         break;
case 4:
case 6:
case 9:
case 11: System.out.println("30 days");
         break;
case 2: System.out.println("28 or 29 days");
default: System.out.println("Invalid no.");
}
}
```

// pairing String in switch expression

```
import java.util.Scanner;
class Demo
{
public static void main(String[] args)
{
String s;
Scanner ss = new Scanner(System.in);
System.out.println("Enter a month no. to get no. of days in that month");
s = ss.nextLine();
switch(s)
{
case "one": System.out.println("Jan"); break;
case "two": System.out.println("Feb"); break;
case "twelve": System.out.println("Dec"); break;
default: System.out.println("Invalid");
}
}
}
```

2008 APRIL 08 TUESDAY

Ternary statement:-

This operator works on the same line as if statement.

Syntax:- Condition ? Expression1 : Expression2;
(Expression1 ↓ true condition, Expression2 ↓ if False)

① WAP to print the greater no. b/w two nos. using ternary statement.

Prog:-

```
// prog to print the greater no. b/w 2 nos.
import java.util.*;
class Ternary
{
public static void main(String args[])
{
int x, y;
Scanner s = new Scanner(System.in);
System.out.println("Enter two nos.");
x = s.nextInt();
y = s.nextInt();
(x>y) ? System.out.println(x) : System.out.println(y);
}
}
```

09 APRIL 2008
WEDNESDAY
100-266 • WEEK 15

Loop :-

- ENTRY CONTROLLED or PRE-TESTED → while, for
- EXIT CONTROLLED or POST TESTED → do-while

① WAP to display no. from 1 to 5 using while, for and do-while loop.

Prog:
```
//using while loop
import java.util.*;
class Num
{
    public static void main(String args[])
    {
        int i=1;
        while(i<=5)
        {
            System.out.print(i);
            i=i+1;
            System.out.println();
        }
    }
}
```



```
//using for loop
import java.util.*;
class Num
{
    public static void main(String args[])
    {
        int i;
        for(i=1;i<=5;++i)
        {
            System.out.print(i);
            System.out.println();
        }
    }
}

//using do while loop
import java.util.*;
class Num
{
    public static void main(String args[])
    {
        int i;
        i=1;
        do
        {
            System.out.print(i);
            i=i+1;
            System.out.println();
        }while(i<=5);
    }
}
```

11 APRIL 2008
FRIDAY
102-264 • WEEK 15

② WAP to print the following series:-
1, 4, 9, 16, 25, - - - , 100

Prog:- //using while loop
```
import java.util.*;
class Pat
{
  public static void main(String args[])
  {
    int i;
    i=1;
    while(i<=10)
    {
      System.out.print(i*i);
      i=i+1;
      System.out.println();
    }
  }
}
```

//using for loop
```
import java.util.*;
class Pat
{
  public static void main(String args[])
  {
    int i;
    for(i=1; i<=10; ++i)
    {
      System.out.print(i*i);
      System.out.println();
    }
  }
}
```

NOTES

APRIL 2008
M T W T F S S M T W T F S S M T W T F S S M T W T F S 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

3

2008 APRIL 12
WEEK 15 • 103-263 SATURDAY

//using do while loop
```
import java.util.Scanner;
class Pat
{
  public static void main(String args[])
  {
    int i;
    i=1;
    do
    {
      System.out.print(i*i);
      i=i+1;
      System.out.println();
    }while(i<=10);
  }
}
```

③ WAP to generate lucas series:-
1, 3, 4, 7, 11, 18, - - - -

Prog:- //using while loop
```
import java.util.*;
class Lucas
{
  public static void main(String args[])
  {
    int x=2, y=1, z=1, term, i;
    System.out.println("Enter no. of term");
    Scanner s=new Scanner(System.in);
```

NOTES

SUNDAY 13

M T W T F S S M T W T F S S M T W T F S S M T W T F S S 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
2008

```
        term = s.nextInt();
        i=1;
        while(i<=term)
        {
          System.out.print(z+"\t");
          z = x+y;
          x = y;
          y = z;
          i=i+1;
        }
    }
}
// using do-while
import java.util.*;
class Lucas
{
    public static void main(String args[])
    {
        int x=2, y=1, z=1, term, i;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter no. of term");
        term = s.nextInt();
        i=1;
        do
        {
            System.out.print(z+"\t");
            z = x+y;
            x = y;
```

NOTES


```
            y = z;
            i = i+1;
        }while(i<=term);
    }
}
// using for loop
import java.util.*;
class Lucas
{
    public static void main(String args[])
    {
        int x=2, y=1, z=1, term, i;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter no. of term");
        term = s.nextInt();
        for(i=1; i<=term; ++i)
        {
            System.out.print(z+"\t");
            z = x+y;
            x = y;
            y = z;
        }
    }
}
```

NOTES

16 APRIL 2008
WEDNESDAY
107-259 • WEEK 16

④ WAP to check the given no. is dracula or not. [1247 ⇒ 1+2+4=7 , 369 ⇒ 3+6=9]

Prog:- //using while loop

```
import java.util.*;
class Drac
{
  public static void main(String args[])
  {
    int r, t, n, s=0;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter a no.");
    n = s.nextInt();
    t = n%10;
    n = n/10;
    while(n>0)
    {
      r = n%10;
      s = s+r;
      n = n/10;
    }
    if(t==s)
      System.out.println("Dracula is"+n);
    else
      System.out.println("Not Dracula is"+n);
  }
}
```

NOTES

APRIL 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 APRIL 17
WEEK 16 • 108-258 THURSDAY

//using for loop

```
import java.util.*;
class Drac
{
  public static void main(String args[])
  {
    int r, t, n, s=0;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter a no.");
    n = s.nextInt();
    t = n%10;
    n = n/10;
    for( ; n>0 ; n=n/10)
    {
      r = n%10;
      s = s+r;
    }
    if(t==s)
      System.out.println("Dracula is"+n);
    else
      System.out.println("Not dracula is"+n);
  }
}
```

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S MAY

(5) WAP to check the no. is perfect or not.

Prog:-

```
//prog using while loop
import java.util.*;
class Perf
{
    int n, i, s=0;
    Scanner s = new Scanner(System.in);
    System.out.println("Enter a no.");
    n = s.nextInt();
    i=1;
    while(i<=n/2)
    {
        if(n%i==0)
        {
            s=s+i;
        }
        i=i+1;
    }
    if (n==s)
        System.out.println("Perfect no. is"+n);
    else
        System.out.println("Not perfect no."+n);
}
```

Notes


```
//prog using for loop
import java.util.*;
class Perf
{
    int n, i, s=0;
    Scanner s = new Scanner(System.in);
    System.out.println("Enter a no.");
    n = s.nextInt();
    for(i=1; i<=n/2; ++i)
    {
        if(n%i==0)
        {
            s=s+i;
        }
    }
    if(n==s)
        System.out.println("Perfect no.="+n);
    else
        System.out.println("Not Perfect no.="+n);
}
```

Notes

⑥ WAP to print the factorial of a given no. using for, while, do-while.

Prog: //using for loop

```
import java.util.*;
class fact
{
    public static void main(String args[])
    {
        int n, f=1;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a no.");
        n = s.nextInt();
        for( ; n>=1; --n)
        {
            f = f * n;
        }
        System.out.println("Factorial = "+f);
        System.out.println( );
    }
}
```

NOTES


```
//using while loop
import java.util.*;
class fact
{
    public static void main(String args[])
    {
        int n, f=1;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a no.");
        n = s.nextInt();
        while(n>=1)
        {
            f = f*n;
            n = n-1;
        }
        System.out.println("factorial="+f);
        System.out.println( );
    }
}
```

NOTES

29/11/14

30 APRIL 2008 WEDNESDAY 121-245 • WEEK 18

Array :- It is collection of similar types of data elements that are accessed through a common name called array name. Array is a list which holds multiple elements of same type and ide elements are processed / accessed using arrayname & index no.

Note :- In array, whenever we cross the boundary of index no., it produces an error; while in C and C++, it does not produce error.

Types of Array :-

- Single / One dimension array
- Multi-dimension array → Two-d, Three-d
- Jagged

① **Single Dimension :-** It is collection of elements arranged row wise.

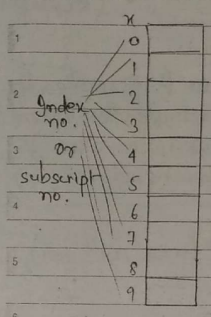
Syntax :-

```
datatype arrayname[] = new datatype[size];
                or
datatype []arrayname = new datatype[size];
                or
datatype arrayname[];
arrayname = new datatype[size];
```

2008 WEEK 18 • 122-244 MAY 01 THURSDAY

Ex :-

```
int x[] = new int[10];
          or
int []x = new int[10];
          or
int []x;
x = new int[10];
```



x[2] ⇒ correct

x[8] ⇒ correct

x[14] ⇒ error in java but not in C and C++.

Ex :-

```
float m[] = new float[5];
            or
float m[];
m = new float[5];
            or
float[]m = new float[5];
```

Q.① WAP to enter elements in array and display it.

Prog:-
```
//prog to enter elements in array & display it
import java.util.*;
class Demo
{
    public static void main(String []args)
    {
        int x[] = {5,12,9,7,25};  <- array initialized
        int y[] = new int[5];
        float m[];
        m = new float[10];
        y[0] = 500;
        y[1] = 700;
        y[2] = 5000;
        y[3] = 7000;
```
program will automatically take y[4] = 0

```
        Scanner s = new Scanner(System.in);
        System.out.println("Enter 10 float elements");
        for(int i=0; i<10; ++i)
        {
            m[i] = s.nextFloat();
        }
        System.out.println("Elements of x is");
        for(int i=0; i<5; ++i)
        {
            System.out.print(x[i] + "\t");
        }
        System.out.println("\n Elements of y is");
        for(int j=0; j<5; ++j)
        {
            System.out.print(y[j] + "\t");
        }
        System.out.println("\n Elements of m are");
        for(int i=0; i<10; ++i)
        {
            System.out.print(m[i] + "\t");
        }
    }
}
```

Q.② WAP to print the greatest no. among ten nos.

Prog:-
```
//prog to print the greatest no. among 10 nos.
import java.util.*;
class Demo
{
    public static void main(String args[])
    {
        int m[] = new int[10], i, g;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter 10 elements");
```

```
        for(i=0; i<10; ++i)
        {
            m[i] = s.nextInt();
        }
        g = m[0];
        for(i=1; i<10; ++i)
        {
            if(g < m[i])
                g = m[i];
        }
        System.out.println("Greatest value is "+g);
    }
}
```

Q.③ WAP to explain the concept of array index out of bound in java.

Prog:- //concept of array index out of bound

```
import java.util.*;
class Demo
{
    public static void main(String args[])
    {
        int x[] = {15, 100, 200, 30, 19};
        System.out.println(x[7]);
    }
}
```

Output:- Error

2/12/14

New Version of For Loop:-

In Java, there is another format of for loop which is specially used on list.

Syntax:- for(datatype variable : arrayname) (same as the type of array)
{

}

Q. WAP to display the elements of an array using the new version of for loop as well as normal for loop.

Prog:- //Example to display elements of array

```
class Demo
{
    public static void main(String args[])
    {
        int x[] = {25, 100, 500, 200, 900, 10};
        System.out.println("Elements of x are:-");
        for(int i:x)
        {
            System.out.println(i);
        }
        System.out.println("Elements of x are:-");
        for(int j=0; j<x.length; ++j)
        {
            System.out.println(x[j]);
        }
    }
}
```

07 MAY 2008
WEDNESDAY
128-238 • WEEK 19

Note:- In Java, length property is used to return/evaluate the length of array.

Syntax:- arrayname.length;

② Two-dimension array:-

It is collection of elements arranged in the form of row & column.

Syntax:-

datatype arrayname[][]=new datatype[size1][size2];
or
datatype [][]arrayname=new datatype[size1][size2];
or
datatype arrayname[][];
arrayname=new datatype[size1][size2];

Ex:- int x[][]=new int[3][4];
or
int [][]x=new int[3][4];
or
int x[][];
x=new int[3][4];

NOTES

MAY 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S 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

2008 MAY 08
WEEK 19 • 129-237 THURSDAY

Q.① WAP to enter elements in 2-D array and display it.

Prog:- //prog to enter & display elements in 2-D

```
import java.util.*;
class Example
{
    public static void main(String args[])
    {
        int x[][]={{15,20},{20,40},{100,500}};
        int y[][]=new int[2][2];
        y[0][0]=100;
        y[0][1]=300;
        y[1][0]=900;
        int z[][];
        int z[][]=new int[3][4];
        Scanner s=new Scanner(System.in);
        int i,j;
        System.out.println("Enter elements in z");
        for(i=0;i<3;++i)
        {
            for(j=0;j<4;++j)
            {
                z[i][j]=s.nextInt();
            }
        }
        System.out.println("Elements of x are:");
        for(i=0;i<3;++i)
        {
```

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S 30 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 JUNE 2008

```
        for(j=0;j<2;++j)
        {
            System.out.print(x[i][j]+"\t");
        }
        System.out.println();
    }
    System.out.println("Elements of y are:-");
    for(i=0;i<2;++i)
    {
        for(j=0;j<2;++j)
        {
            System.out.print(y[i][j]+"\t");
        }
        System.out.println();
    }
    System.out.println("Elements of z are:-");
    for(i=0;i<3;++i)
    {
        for(j=0;j<4;++j)
        {
            System.out.print(z[i][j]+"\t");
        }
        System.out.println();
    }
}
```

2) Array & String

③ Three Dimension Array:-

```
//prog of three dimension array
import java.util.Scanner;
class Array
{
    public static void main(String args[])
    {
        int x[][][] = new int[5][3][2];
        int i,j;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the marks of 5 students in 3 subjects");
        for(i=0;i<5;++i)
        {
            for(j=0;j<3;++j)
            {
                x[i][j][0] = s.nextInt();
                x[i][j][1] = s.nextInt();
            }
        }
        System.out.println("Marks details are:-");
        for(i=0;i<5;++i)
        {
            for(j=0;j<3;++j)
            {
                System.out.print(x[i][j][0]+"\t"+x[i][j][1]+"\t\t");
            }
            System.out.println();
        }
    }
}
```

12 MAY 2008
MONDAY 133-233 • WEEK 20

	0		1		2	
	0	1	0	1	0	1
0	70	32	68	21	79	19
1						
2						
3						
4						

Jagged Array/variable length array:-

An array which has variable column length is known as jagged array or variable length array. A variable length array can hold different length of column in each row.

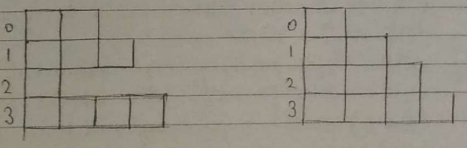
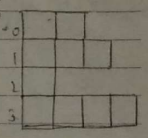


fig: Jagged Array

2008 MAY 13
WEEK 20 • 134-232 TUESDAY

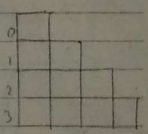
Syntax:-
datatype arrayname[][]=new datatype[size][];
arrayname[row index]=new datatype[size];
(row, col)

Ex:- int x[][]=new int[4][];
x[0]=new int[2];
x[1]=new int[3];
x[2]=new int[1];
x[3]=new int[4];



or

int x[][]=new int[4][];
x[0]=new int[1];
x[1]=new int[2];
x[2]=new int[3];
x[3]=new int[4];



```
//Explain the example of jagged array
import java.util.Scanner;
class Deem
{
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        int x[][]=new int[4][];
        x[0]=new int[2];
        x[1]=new int[3];
        x[2]=new int[1];
```

```
x[3] = new int[5];
int i;
System.out.println("Enter two elements");
for(i=0; i<2; ++i)
{
    x[0][i] = s.nextInt();
}
System.out.println("Enter three elements");
for(i=0; i<3; ++i)
{
    x[i][i] = s.nextInt();
}
System.out.println("Enter a no.");
x[2][0] = s.nextInt();
System.out.println("Enter five elements");
for(i=0; i<5; ++i)
{
    x[3][i] = s.nextInt();
}
System.out.println("Elements are :-");
for(i=0; i<2; ++i)
{
    System.out.print(x[0][i] + '\t');
}
System.out.println();
for(i=0; i<3; ++i)
{
    System.out.print(x[i][i]+'\t');
}
System.out.println();
```



```
System.out.println(x[2][0]);
for(i=0; i<5; ++i)
{
    System.out.print(x[3][i]+'\t');
}
System.out.println();
}
}
```

Q (1) WAP to swap two arrays.

Prog:-

```
// prog to swap two arrays
import java.util.*;
class Swap
{
    public static void main(String args[])
    {
        int x[] = new int[5];
        int y[] = new int[5];
        int t; Scanner s = new Scanner(System.in);
        System.out.print("Enter values in array");
        for(int i=0; i<5; ++i)
        {
            x[i] = s.nextInt();
            y[i] = s.nextInt();
        }
        for(int i=0; i<5; ++i)
        {
            t = x[i];
            x[i] = y[i];
```

```
        System.out.println("Swapped elements are:");
        for(i=0; i<5; ++i)
        {
            System.out.println(x[i]);
        }
        for(i=0; i<5; ++i)
        {
            System.out.println(y[i]);
        }
    }
}
```

Q.② WAP to print square of values in array.

Prog:-

```
//prog to print square of values in array
import java.util.*;
class Square
{
    public static void main(String args[])
    {
        int n[] = new int[5];
        int i;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter values in array");
        for(i=0; i<5; ++i)
        {
            n[i] = s.nextInt();
        }
```



```
        for(i=0; i<5; ++i)
        {
            System.out.print(n[i]*n[i]);
        }
    }
}
```

Q.③ WAP to print the greatest no. in matrix

Prog:-

```
//prog to print the greatest no. in matrix
import java.util.*;
class Great
{
    public static void main(String... args)
    {
        int n[][] = new int[3][3];
        int i, j, g;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter values in matrix");
        for(i=0; i<3; ++i)
        {
            for(j=0; j<3; ++j)
            {
                n[i][j] = s.nextInt();
            }
        }
        g = n[0][0];
        for(i=0; i<3; ++i)
        {
            for(j=0; j<3; ++j)
```

```
        {
            if(n[i][j] > g)
                g = n[i][j];
        }
    }
    System.out.print("Greatest no.="+g);
  }
}
```

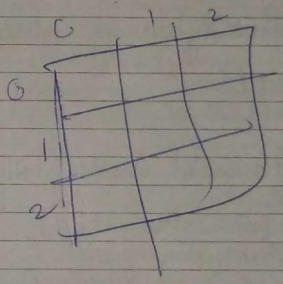
Q.④ WAP to print the sum of diagonal.

Prog:-

```
//prog to print the sum of diagonal
import java.util.*;
class Sum
{
  public static void main(String... args)
  {
    int n[][] = new int[3][3];
    int i,j;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter values in matrix");
    for(i=0; i<3; ++i)
    {
      for(j=0; j<3; ++j)
      {
        n[i][j] = s.nextInt();
      }
```



```
    for(i=0; i<3; ++i)
    {
      s = 0;
      for(j=0; j<3; ++j)
      {
        if((i+j)%2==0)
          s = s+n[i][j];
      }
    }
    System.out.print("Sum of diagonal="+s);
  }
}
```



LABELLED BREAK & LABELLED CONTINUE :-

Labelled break :-

In Java, break keyword has more flexibility regarding of jumping from C and C++. Labelled break of Java exits the control from a particular section of program which is denoted as a label. Like C and C++; in java, label is an identifier followed by colon(:) and used to represent a particular section of program.

In C and C++, break keyword exit the control only from that loop where break keyword is specified. Same thing is also applied in Java, but java has additional facility of labelled break that exit the controls from any section of loop program.

Syntax to specify label :- labelname:

Ex :- shai:
Rahi:

Syntax of labelled break :- break labelname;

Ex :- break shai;
break rahi;

NOTES

```
//prog to explain the example of break
//keyword in Java
class Diksha
{
    public static void main(String args[])
    {
        int i;
        for(i=1; i<=10; ++i)
        {
            if(i==5)
                break;
            System.out.println(i);
        }
    }
}
```

Output:- 1234

```
//Another example of break
class Brea
{
    public static void main(String args[])
    {
        int i,j;
        for(i=1; i<=5; ++i)
        {
            for(j=1; j<=5; ++j)
            {
                if(i==3)
                    break;
```



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

Output:-

```
11
12
13
14
15
21
22
23
24
25
41
42
43
44
45
51
52
53
54
55
```

Output:-

```
11
12
21
22
31
32
41
42
51
52
```

```
class Brea
{
    public static void main(String args[])
    {
        int i,j;
        for(i=1; i<=5; ++i)
```

```
            {
                for(j=1; j<=5; ++j)
                {
                    if(j==3)
                        break;
                    System.out.println(i+" "+j);
                }
            }
        }
    }
```

//Example of break

Output:

```
class Diksha
{
    public static void main(String... args)
    {
        int i, j;
        for(i=1; i<=5; ++i)
        {   if(i==3)
                break;
            for(j=1; j<=5; ++j)
            {
                System.out.println(i+" "+j);
            }
        }
    }
}
```

Output:

```
11
12
15
21
22
25
31
32
35
41
42
45
51
52
55
```

NOTES

//Another Example

Output:

```
class Demo
{
    public static void main(String... args)
    {
        int i, j;
        for(i=1; i<=5; ++i)
        {
            for(j=1; j<=5; ++j)
            {
                System.out.println(i+" "+j);
            }
            if(i==3)
                break;
        }
    }
}
```

Output:

```
11
12
13
14
15
21
22
23
24
25
31
32
33
34
35
```

//Another Example

```
class Teach
{
    public static void main(String args[])
    {
        int i,j;
        for(i=1; i<=5; ++i)
        {
            if(i==3)
                break;
            for(j=1; j<=5; ++j)
```

NOTES

28 MAY WEDNESDAY 2008 149-217 • WEEK 22

```
            {
                if(J==3)
                  break;
                System.out.println(i+" "+j);
            }
        }
    }
}

//prog of labelled break
class Sheet
{
    public static void main(String args[])
    {
        int i,j;
        Diksha: for(i=1;i<=5;++i)
                {
                    Santu: for(j=1;j<=5;++j)
                           {
                               if(j==3)
                                 break Diksha;
                               System.out.println(i+" "+j);
                           }
                }
    }
}
```

Output:-

```
1 1
1 2
```

29 MAY THURSDAY 2008 WEEK 22 • 150-216

Labelled Continue:-

In Java, there is a concept of labelled continue in which continue keyword can be used with label-name. With the help of labelled continue, we can use the continue keyword to skip statement of java in flexible way according to label.

Syntax:- continue labelname;

Ex:- continue demo;

```
//prog to explain the example of labelled continue
class Diksha
{
    public static void main(String args[])
    {
        int i;
        for(i=1;i<=10;++i)
        {
            if(i==5)
              continue;
            System.out.println(i);
        }
    }
}
```

Output:- 1 2 3 4 6 7 8 9 10

30 MAY 2008 FRIDAY 151-215 • WEEK 22

```
//Another example of continue
class Diksha
{
    public static void main(String args[])
    {
        int i,j;
        for(i=1; i<=5; ++i)
        {
            for(j=1; j<=5; ++j)
            {
                if(i==3)
                    continue;
                System.out.println(i+""+j);
            }
        }
    }
}
```

output
11
12
14
15
21
22
24
25
31
32
34
35
41
42
44
45
51
52
54
55

```
//Another example
class Diksha
{
    public static void main(String... args)
    {
        int i,j;
        for(i=1; i<=5; ++i)
        for(j=1; j<=5; ++j)
        {
            if(j==3)
                continue;
            System.out.println(i+""+j);
        }
    }
}
```

NOTES

MAY 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 MAY 31 SATURDAY WEEK 22 • 152-214

```
//Another example
class Diksha
{
    public static void main(String... args)
    {
        int i,j;
        for(i=1; i<=5; ++i)
        {
            if(i==3)
                continue;
            for(j=1; j<=5; ++j)
            {
                if(j==3)
                    continue;
                System.out.println(i+""+j);
            }
        }
    }
}
```

output
11
12
14
15
21
22
24
25
41
42
44
45
51
52
54
55

```
//Example of labelled continue
class Diksha
{
    public static void main(String args[])
    {
        int i,j;
        Diksha: for(i=1; i<=5; ++i)
        {
            Santu: for(j=1; j<=5; ++j)
            {
                if(j==3)
                    continue Diksha;
```

NOTES

SUNDAY 1

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S JUNE 2008

```
                System.out.println(i+" "+j);
            }
        }
    }
}
//Another example
class Demo
{
    public static void main(String... args)
    {
        int i, j;
        Ashu: for(i=1; i<=5; ++i)
            { if(i==3)
                continue Ashu;
              Sam: for(j=1; j<=5; ++j)
                  {
                    if(j==3)
                      continue Ashu;
                    System.out.println(i+" "+j);
                  }
            }
    }
}
```

output-

```
1 1
1 2
2 1
2 2
3 1
3 2
4 1
4 2
5 1
5 2
```

Output:-

```
11
12
21
22
41
42
51
52
```

STRING:-

A sequence of characters are called string. It means, a string is made up of several characters, i.e collection of characters.

In Java, strings are represented in Java, strings are represented in form of object. These are two built-in classes specified in java.lang package which are used to create as well as to access string, these classes are as follows:-

(i) String
(ii) StringBuffer } classes
(iii) StringBuilder

(i) String:-

This class is a pre-defined class specified in java.lang package. This class is used to create a fixed length string. The string is created with string class cannot be changed/modified/edited.

Syntax to create a string using string class

(a) Using reference of string class
(b) using string constructor

NOTES

(a) using reference of string class:-

classname ↑

String reference name;

Ex:- String s;
s = "Manish";

Ex:- String m = "Shailesh";

(b) using string constructor:-

String objectname = new String();
or
String objectname = new String("Stringvalue/constant");

Ex:- String s = new String();
s = "Ramesh";

Ex:- String m = new String("Shailesh");

Q. WAP to create string & display it.

Prog:- //prog to create string & display it

```
class Demo
{
    public static void main(String args[])
    {
        String s = "Prashant Trivedi";
        String s1 = new String("Shailesh Kumar");
        System.out.println("First String is" + s);
        System.out.println("Second String is" + s1);
    }
}
```


or

```
class Demo
{
    public static void main(String args[])
    {
        String s;
        String s1 = new String();
        s = "Prashant Trivedi";
        s1 = "Shailesh Kumar";
        System.out.println("First string is" + s);
        System.out.println("Second string is" + s1);
    }
}
```

Q. WAP to enter a string from keyboard and display it.

Prog:- //prog to enter a string & display it.

```
import java.util.*;
class Demo
{
    public static void main(String args[])
    {
        String s;
        String s1 = new String();
        Scanner m = new Scanner(System.in);
        System.out.println("Enter two String");
        s = m.nextLine();
        s1 = m.nextLine();
```

09 JUNE 2008
MONDAY
161-205 • WEEK 24

```
        System.out.println("First string is"+s);
        System.out.println("Second string is"+s1);
    }
}
```

Q. WAP to enter a string & store into another string.

Prog:-
```
//prog to enter a string & store into another
import java.util.*;
class Demo
{
    public static void main(String args[])
    {
        String s;
        System.out.println("Enter a string");
        Scanner m = new Scanner(System.in);
        s = m.nextLine();
        String s1 = s;
        String s2 = new String(s);
        System.out.println("First string is"+s);
        System.out.println("Second string is"+s1);
        System.out.println("Third string is"+s2);
    }
}
```

NOTES

2008 WEEK 24 • 162-204 JUNE TUESDAY 10

Methods / Functions of String class:-

- toLowerCase()
- toUpperCase()
- replace()
- trim()
- equals()
- equalsIgnoreCase()
- Length()
- charAt()
- compareTo()
- concat()
- substring()
- indexOf()
- startsWith()
- endsWith()

① toLowerCase():- This function converts a string into lower case.

Syntax:- stringconstant/reference/object.toLowerCase();

Ex:- "MANOJ".toLowerCase();

Ex:- String s = "RAMESH";
String s1 = s.toLowerCase();

```
//Explain the example of toLowerCase()
class Demo
{
    public static void main(String args[])
    {
        String s = "RAMESH";
        String s1 = s.toLowerCase();
        String s2 = "KUMUD".toLowerCase();
        System.out.println("First string is "+s1);
```

NOTES

JULY 2008

```
        System.out.println("Second string is"+s2);
        System.out.println("Third string is"+
                           "RITIKA".toLowerCase());
    }
}
```

② toUpperCase():- This function converts a string into upper case.

Syntax:- constant/reference/object.toUpperCase();

Ex:- "MANOJ".toUpperCase();

Ex:-
```
String s = "ramesh";
String s1 = s.toUpperCase();
```

//Explain the example of toUpperCase()

```
class Demo
{
    public static void main(String args[])
    {
        String s = "ramesh";
        String s1 = s.toUpperCase();
        String s2 = "kumud".toUpperCase();
        System.out.println("First string is"+s1);
        System.out.println("Second string is"+s2);
        System.out.println("Third String is"+"ritika".
                           toUpperCase());
    }
}
```

NOTES

③ replace():- This function replaces an existing character of a string to a new character

Syntax:- String.replace(old char, new char)

Ex:-
```
String s = "Kamal";
s.replace('a','o');
```

Output:- Komol

//Explain the example of replace()

```
class Demo
{
    public static void main(String args[])
    {
        String s = "Kamal";
        String s1 = s.replace('a','o');
        String s2 = "Malyalam";
        s2.replace('o','i');
        System.out.println("First string is"+s);
        System.out.println("Second string is"+s1);
        System.out.println("Third string is"+s2);
    }
}
```

NOTES

Output:-
```
Kamal
Komol
Malyalam
```

13 JUNE 2008
FRIDAY
165-201 • WEEK 24

④ length():- This function returns the length of a string. The length of string is total no. of characters of string.

Syntax:- String.length();

Ex:- String s="Komal";
int l = s.length();
int l1 = "Ramesh".length();

//prog to explain the example of length()

```
class Demo
{
    public static void main(String... args)
    {
        String s = "Kamal";
        int l = s.length();
        int l1 = "UJJAWAL".length();
        System.out.println("First string length"+l);
        System.out.println("Second string length"+l1);
    }
}
```

Note:- replace():-
Since, we know that the string created with string class is not modified. So, the replace() function cant replace the original string, while we can store the result of replace() in another string.

JUNE 2008 | M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S | JULY 2008

16/12/114

2008 JUNE 14 SATURDAY
WEEK 24 • 166-200

⑤ trim():- This function removes all leading and trailing spaces of string. It means, this function removes the beginning & end space of a string.

Syntax:- String.trim();

Ex:- "----Diksha-----";
leading space, trailing space

//prog to explain the example of trim()

```
class Demo
{
    public static void main(String args[])
    {
        String s = "----Kamal-----";
        System.out.println(s.length());
        String s1 = s.trim();
        System.out.println(s1.length());
        System.out.println(s);
        System.out.println(s1);
        System.out.println(s.trim());
    }
}
```

Output:- 18
5
----Kamal
Kamal
Kamal

SUNDAY 15

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S | JULY 2008

16 JUNE 2008
MONDAY
168-198 • WEEK 25

(6) equals():- This function compares two strings & return boolean values "True" or "False", if both strings are same or not same respectively.

Syntax:- String1.equals(string2);

```
//prog to explain the example of equals
class Demo
{
    public static void main(String args[])
    {
        String s = "ram";
        String s2 = "ram";
        String s1 = "ramesh";
        boolean r1, r2;
        r1=s.equals(s2);
        r2 = s.equals(s1);
        System.out.println(r1);
        System.out.println(r2);
        System.out.println("Ram".equals(s));
    }
}
```

NOTES

Output:-

```
True
False
False
```

2008 JUNE 17
WEEK 25 • 169-197 TUESDAY

(7) equalsIgnoreCase():- This function compare two strings character by character & returns boolean value. This function ignore the case while compairing.

Syntax:- String1.equalsIgnoreCase(String 2);

```
//Example of equalIgnoreCase
class Demo
{
    public static void main(String args[])
    {
        String s = "ram";
        String s2 = "ram";
        String s1 = "ramesh";
        boolean r1, r2;
        r1=s.equalsIgnoreCase(s2);
        r2= s.equalsIgnoreCase(s1);
        System.out.println(r1);
        System.out.println(r2);
        System.out.println("Ram".equalsIgnoreCase(s));
    }
}
```

NOTES

Output:-

```
True
false
True
```

18 JUNE 2008
WEDNESDAY
170-196 • WEEK 25

⑧ charAt():- This function returns a character from a string at a particular index.

Syntax:- String.charAt(index no.);

//Explain the example of charAt() function

```
class Demo
{
    public static void main(String args[])
    {
        String s1 = "ramesh";
        System.out.println(s1.charAt(2));
        for(int i=0; i<s1.length(); ++i)
        {
            System.out.println(s1.charAt(i));
        }
    }
}
```

Output:
m
r
a
m
e
s
h

NOTES

JUNE 2008

2008
WEEK 25 • 171-195

17 | 12 | 14
JUNE 19
THURSDAY

⑨ compareTo():- This function compare two strings and return zero(0) if both strings are same, returns positive value if first string is greater than second, returns negative value if first string is less than second.

Syntax:- string1.compareTo(string2);

//Example of compareTo() function

```
class Demo
{
    public static void main(String args[])
    {
        String s = "Ramesh";
        String s1 = "Ramesh";
        String s2 = "Manish";
        int r1, r2;
        r1 = s.compareTo(s1);
        r2 = s.compareTo(s2);
        System.out.println(r1);
        System.out.println(r2);
        System.out.println("Sohan".compareTo(s));
    }
}
```

NOTES

Output:-
0
-27
1

JULY 2008

(10) concat() :- This function concates two strings.

Syntax:- String1.concat(string2);

```
//Example of concat()
class Demo
{
  public static void main(String... args)
  {
    String s = "Ramesh";
    String s2 = "Manish";
    String s3 = s.concat(s2);
    System.out.println(s);
    System.out.println(s3);
    System.out.println("Sohan".concat(s2));
  }
}
```

Output:-

```
Ramesh
RameshManish
SohanManish
```


(11) substring() :- This function extracts character from a string.

Syntax:- string.substring(start-position);
or
string.substring(start, end-position);
↓ excludes end position character

Ex:- String s = "Manmohan"; (0 1 2 3 4 5 6 7)
String s1 = s.substring(3); → Mohan
String s2 = s.substring(3,6); → Moh

```
//Example of substring() function
class Demo
{
  public static void main(String args[])
  {
    String s = "Ramesh";
    String s3 = s.substring(2);
    System.out.println(s);
    System.out.println(s3);
    System.out.println("Sohan".substring(1,3));
  }
}
```

Output:-

```
Ramesh
mesh
oh
```

(12) indexOf() :- This function returns the index no. of a particular character of string.

Syntax:- string.indexOf(character); (returns index from position)
or
string.indexOf(character, position);

```
//Example of IndexOf()
class Diksha
{
  public static void main(String args[])
  {
    String s="Saras";
    int x=s.indexOf('a');
    System.out.println(x);
    System.out.println(s.indexOf('a',2));
    System.out.println("malyalam".indexOf('y'));
    System.out.println("malyalam".indexOf('a',5));
  }
}
```

NOTES

(13) startsWith() :- This function returns true if a string starts with a particular text/ word. Otherwise, returns false.

Syntax:- string.startsWith("Text");

```
//Example of startsWith()
class Diksha
{
  public static void main(String... args)
  {
    String s="He is a good boy";
    System.out.println(s.startsWith("He"));
    System.out.println(s.startsWith("good"));
  }
}
```

Output:- True
False

NOTES

(14) endsWith():- This function true if a string ends with a particular text/word. Otherwise, return false.

Syntax:- string.endsWith("Text");

```
//Example of endsWith()
class Diksha
{
    public static void main(String... args)
    {
        String s = "He is a good boy";
        System.out.println(s.endsWith("boy"));
        System.out.println(s.endsWith("good"));
    }
}
```

Output:-

```
True
False
```

Q. WAP to explain the example of length().

Prog:-

```
//example of length()
class Demo
{
    public static void main(String args[])
    {
        String s = "Manjusa";
        int i, j;
        for(i=1; i<=s.length(); ++i)
        {
            for(j=0; j<i; ++j)
            {
                System.out.print(s.charAt(j));
            }
            System.out.println();
        }
    }
}
```

Output:-

```
M
Ma
Man
Manj
Manju
Manjus
Manjusa
```


NOTES

StringBuffer():- This function is used to create a string of variable/changeable length. The string created with StringBuffer class can be modified/edited. We can insert text in a string created with StringBuffer.

Syntax to create a string using stringBuffer class

StringBuffer objectname = new stringBuffer("string");

Ex:- StringBuffer s = new stringBuffer("welcome");

StringBuffer s1 = "welcome"; -> error

Q. WAP to enter a string using stringBufferclass

Prog:-
```
//prog to enter a string using stringBuffer class
class Diksha
{
    public static void main(String args[])
    {
        stringBuffer s = new stringBuffer("He is a good boy");
        System.out.println(s);
    }
}
```

NOTES

SUNDAY 29

24/12/14

30 JUNE 2008
MONDAY
182-184 • WEEK 27

Functions of stringBuffer class :-

- setCharAt()
- append()
- insert()
- reverse()
- length()
- substring()
- charAt()
- capacity()
- delete()
- replace()

(1) setCharAt() :-

This function adds/sets a character on a particular position.

Ex Syntax :- stringBuffer s = new stringBuffer("Manisha");
s.setCharAt(3,'o');
System.out.println(s); → Manosha

Syntax :- string.setCharAt(Position, character);

//prog to explain the example of setCharAt()

```
class Demo
{
    public static void main(String... args)
    {
        StringBuffer s = new StringBuffer("Manisha");
```

NOTES

JUNE 2008 | AUGUST 2008

2008 JULY 01
WEEK 27 • 183-183
TUESDAY

```
        s.setCharAt(3,'u');
        System.out.println(s);
    }
}
```

Output :- Manusha

(2) append() :-

This function adds/appends string at the end of another string.

Syntax :- string1.append(string2);

```
// Example of append()
class Demo
{
    public static void main(String... args)
    {
        StringBuffer s = new StringBuffer("Manisha");
        StringBuffer s1 = new StringBuffer("Manjari");
        s.append(s1);
        System.out.println(s);
    }
}
```

Output :- Manishamanjari

NOTES

02 JULY 2008
WEDNESDAY 184-182 • WEEK 27

(3) insert():-

This function inserts/adds a string within another string at a particular position.

Syntax:- string1.insert(pos, string2);

```
//prog to explain the example of insert()
class Demo
{
    public static void main(String... args)
    {
        StringBuffer s = new StringBuffer("Manisha is a gud");
        StringBuffer s1 = new StringBuffer("good");
        s.insert(13, s1);
        System.out.println(s);
    }
}
```

Output:- Manisha is a good gud.

(4) reverse():-

This function is used to reverse a string.

Syntax:- String.reverse();

26/12/14
2008 JULY 03
WEEK 27 • 185-181 THURSDAY

```
//prog to explain the example of reverse().
class Demo
{
    public static void main(String args[])
    {
        StringBuffer s = new StringBuffer("Radhika is a good girl");
        s.reverse();
        System.out.println(s);
    }
}
```

Output:- lrig doog a si akihdaR

(5) capacity():-

This function returns the capacity of a string.

Syntax:- String.capacity();

```
// Explain the example of capacity()
class Demo
{
    public static void main(String args[])
    {
        StringBuffer s = new StringBuffer("Komal is a good girl");
        int x;
        x = s.capacity();
        System.out.println("capacity is"+x);
    }
}
```

04 JULY 2008
FRIDAY 186-180 • WEEK 27

⑥ delete():-

This function removes/deletes characters of a string.

Syntax:- string.delete(startindex, endindex); (endindex: before end)

//prog to explain the example of delete()

```
class Demo
{
    public static void main(String args[])
    {
        StringBuffer s = new StringBuffer("Malyalam");
        s.delete(1,6);
        System.out.println(s);
    }
}
```

Output:- Mam

⑦ replace():-

This function replaces a string into another string.

Syntax:- string1.replace(startindex, endindex, string);

//prog to explain the example of replace()

```
class Demo
{
    public static void main(String... args)
    {
        StringBuffer s = new StringBuffer("Malyalam");
        s.replace(1,4," y name is");
        System.out.println(s);
    }
}
```

Character Input using Scanner class:-

We can receive character input using a scanner class by ~~with the help of~~ retrieving character at zero(0) index with charAt() function of string class by receiving string using next() or nextline() function of ~~String~~ scanner class.

Syntax:- s.next().charAt(0); ← take a string
'or'
s.nextline().charAt(0); ← take a string

scanner object

Q. WAP to take a character from keyboard & display it.

Prog:-
```
//prog to display a character
import java.util.Scanner;
class Deee
{
    public static void main(String... args)
    {
        Scanner s = new Scanner(System.in);
        char x;
        System.out.println("Enter a character");
        x = s.next().charAt(0);
        System.out.println("Character is: "+x);
    }
```

11 JULY FRIDAY 2008

```
import java.util.*;
class Demo
{
    public static void main(String...args)
    {
        Scanner s=new Scanner(System.in);
        byte a=s.nextByte();
        short b=s.nextShort();
        int c=s.nextInt();
        long d=s.nextLong();
        float e=s.nextFloat();
        double f=s.nextDouble();
        boolean g=s.nextBoolean();
        String h=s.next();
        String i=s.nextLine();
        char j=s.nextLine().charAt(0);
        s.close();
    }
}
```

29/12/14 JULY SATURDAY 05

```
public static void main(String... args)
{
    StringBuffer s=new StringBuffer("malyalam");
    s.replace(1,4,"y name is ");
    System.out.println(s);
}
```

Output:- My name is alam

Receiving input through Command line :-

In Java, we can receive input from command-line during execution of program. A command line is the line where commands are given & arguments specified on command line is known as command line arguments.

The argument given on command line is considered as string and received by main() function. The string name args of main() function receive the argument specified on command line & stored within it in the form of array. It means, the first command line argument is stored on index no. zero(0), second argument on index no. one(1), and so on.

SUNDAY 6

Ex:- String args;

'or'

String s;

s/args is string name/string reference that can hold a single string.

07 JULY 2008
MONDAY
189-177 • WEEK 28

Ex:- String args[] → specified as argument of main()
↓
array of type String that can hold one or more strings.

Ex:- C:\users\Mukesh > ——
↓
command line

Ex:- C:\users\mukesh > java Deee Hi Hello Bye
0 1 2
Command line arguments which is received by String args[] of main().

args
0	Hi
1	Hello
2	Bye

Ex:- C:\users\mukesh > java Deee welcome 12 15 10.5f 20.92f

Here, total five arguments are specified, so the array of size 5 is created for args of type string.

args
0	welcome
1	12
2	15
3	10.5f
4	20.92f

Ex:- C:\users\mukesh > java Deee

no argument is specified, so no array will create.

Note:- Arguments on command-line is separated by space.

NOTES

JULY 2008 | AUGUST 2008

2008 JULY 08
WEEK 28 • 190-176
TUESDAY

Parse means change
Parsetype() → changeable

Q. WAP to explain the example of command line arguments.

Prog:-
```
// example of command line arguments
class Deee
{
    public static void main (String args[])
    {
        System.out.println ("Command Line Arguments are:-");
        for (int i=0 ; i<args.length ; ++i)
        {
            System.out.println (arg[i]);
        }
    }
}
```

Q. WAP to print the sum of two integers nos. using command line argument. Nos. must be supplied on command line.

datatype	wrapper class	
byte	⇒ Byte	⇒ parseByte()
short	⇒ Short	⇒ parseShort()
int	⇒ Integer	⇒ parseInt()
long	⇒ long	⇒ parseLong()
float	⇒ Float	⇒ parseFloat()
double	⇒ Double	⇒ parseDouble()
char	⇒ Character	
boolean	⇒ Boolean	

NOTES

JULY 2008 | AUGUST 2008

09 JULY 2008
WEDNESDAY
191-175 • WEEK 28

```
//prog to print the sum of two integers
//using command line argument
class Deee
{
    public static void main(String args[])
    {
        int x, y, z;
        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);
        z = x+y;
        System.out.println("sum is "+z);
    }
}
```

Q. WAP to print the product of an integer & two double values using command line argument.

Prog:-

```
class Deee
{
    public static void main(String args[])
    {
        int x;
        double y,z,r;
        x = Integer.parseInt(args[0]);
        y = Double.parseDouble(args[1]);
        z = Double.parseDouble(args[2]);
        r = x*y*z;
        System.out.println("Product is"+r);
    }
}
```

JULY 2008

2008
WEEK 28 • 194-172
JULY 12
SATURDAY

Q. WAP to print a reverse of a no. using command line argument.

Prog:-

Q. WAP to enter a string & display it using command line argument.

Prog:-

```
class Deee
{
    public static void main(String args[])
    {
        String s;
        S = args[0];
        System.out.println("String is"+s);
    }
}
```

SUNDAY 13

AUGUST 2008

8/10/15

14 JULY MONDAY 2008

196-170 • WEEK 29

Q. WAP to enter a character & display it using command line argument.

Prog:-

```
//prog to enter a character & display it.
class Demo
{
    public static void main(String args[])
    {
        char x;
        x = args[0].charAt(0);
        System.out.println("Character is" + x);
    }
}
```

BufferedReader:-

Receiving/reading input using BufferedReader class:-

NOTES

JULY 2008

M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	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	•	•	•

16 JULY 2008
WEDNESDAY 198-168 • WEEK 29

Q. WAP to print the sum of two nos.

Prog:- //prog to print the sum of two nos.

```
import java.io.*;
class Sum
{
    public static void main(String args[])throws
    {
        int x, y, r;
        InputStreamReader m = new InputStreamReader
        BufferedReader br=new BufferedReader(m);
        System.out.println("Enter two nos.");
        x=Integer.parseInt(br.readline());
        y=Integer.parseInt(br.readline());
        r=x+y;
        System.out.println("Sum is "+r);
    }
}
```

or

```
import java.io.*;
class Sum
{
    public static void main(String args[])
    {
        int x,y,r;
        InputStreamReader m = new InputStreamReader
                              (System.in);
        BufferedReader br = new BufferedReader(m);
        try
```

2008 WEEK 29 • 199-167 JULY THURSDAY 17

```
        {
            System.out.println("Enter two nos");
            x = Integer.parseInt(br.readline());
            y = Integer.parseInt(br.readline());
            r = x+y;
            System.out.println("Sum is "+r);
        }
IOException }
        catch(IOException e) / Catch(Exception e)
(System.in);  {
            System.out.println(e);
        }
    }
}
```

Q. WAP to print the product of an integer and a double no.

Prog:- //prog to print the product of an integer
//& a double no.

```
import java.io.*;
class Prod
{
    public static void main(String args[])
    {
        int x;
        double y, p;
        BufferedReader br = new BufferedReader(new
                            InputStreamReader(System.in
```

```
        try
        {
            System.out.println("Enter an Integer & a double value");
            x=Integer.parseInt(br.readline());
            y=Integer.parseInt(br.readline());
            p=x*y;
            System.out.println("Product="+p);
        }
        catch(Exception e)
        {
        }System.out.println(e);
    }
}
```

Q. WAP to print the area of circle

Prog:-
```
//prog to print the area of circle
import java.io.*;
class Area
{
    public static void main(String args[])throws IOException
    {
        int r;
        double a;
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter radius");
```



```
        r=Integer.parseInt(br.readline());
        a=Math.PI*r*r;
        System.out.println("Area is"+a);
    }
}
```

Q. WAP to enter a string & display it.

Prog:-
```
import java.io.*;
class Diksha
{
    public static void main(String args[])throws IOException
    {
        String s;
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter a string");
        s=br.readline();
        System.out.println("String is"+s);
    }
}
```

Q. WAP to enter a character & display it.

Prog:-
```
//prog to enter a character & display it
import java.io.*;
class Diksha
{
    public static void main(String args[])throws IOException
    {
        char s;
```

```
BufferedReader br = new BufferedReader
                    (new InputStreamReader(System.in));
System.out.println("Enter a character");
s = (char)br.read();
System.out.println("Character is"+s);
}
}
```

'or'

```
import java.io.*;
class Diksha
{
public static void main(String args[])
{
char s;
BufferedReader br = new BufferedReader
                    (new InputStreamReader(System.in));
try
{
System.out.println("Enter a character");
s = br.readLine().charAt(0);
System.out.println("Character is"+s);
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```


Receiving/Reading input using DataInputStream

Q. WAP to add two nos. using DataInputStream

Prog: //prog to add two nos.

```
import java.io.*;
class Demo
{
public static void main(String args[]) throws I
                                              Except
{
int x, y, z;
DataInputStream d = new DataInputStream(S
System.out.println("Enter two nos.");
x = Integer.parseInt(d.readline());
y = Integer.parseInt(d.readline());
z = x+y;
System.out.println("Sum is"+z);
}
```

Q. WAP to enter a string & display it.

Prog:-
```
//prog to enter a string & display it
import java.io.*;
class Demo
{
    public static void main(String args[])throws
                                        IOException
    {
        String s=new String();
        DataInputStream d=new DataInputStream
                                    (System.in);
        System.out.println("Enter a string");
        s=d.readline();
        System.out.println("String is "+s);
    }
}
```

Q. WAP to enter a character & display it.

Prog:-
```
//prog to enter a character & display it
import java.io.*;
class Demo
{
    public static void main(String args[])throws
                                        IOException
    {
        char x;
        DataInputStream d=new DataInputStream(System.in);
        System.out.println("Enter a character");
        x=(char)d.read(); //x=d.readline().charAt(0)
        System.out.println("character is"+x);
    }
```


Q. WAP to enter elements in one dimensional (1-d) array & display it.

Prog:-
```
//prog to enter elements in 1-d array
import java.io.*;
class Demo
{
    public static void main(String args[])throws
                                        IOException
    {
        int x[]=new int[5];
        DataInputStream d=new DataInputStream
                                    (System.in);
        System.out.println("Enter 5 elements");
        for(int i=0; i<5; ++i)
        {
            x[i]=Integer.parseInt(d.readline());
        }
        System.out.println("Elements are :-");
        for(int i=0; i<5; ++i)
        {
            System.out.print(x[i]+"\t");
        }
    }
}
```

25 | JULY FRIDAY 2008

207/159 • Week 30

Receiving / Reading input using console class

Q. WAP to add two nos. using console class

Prog:-

```
//prog to add two nos.
import java.io.*;
class Demo
{
    public static void main(String args[])
    {
        int x, y, r;
        Console m = System.console();
        System.out.println("Enter two nos.");
        x = Integer.parseInt(m.readline());
        y = Integer.parseInt(m.readline());
        r = x + y;
        System.out.println("Sum is "+r);
    }
}
```

28 JULY 2008
MONDAY 210-156 • WEEK 31

Executing Java Program on Netbeans Editor:-

Step 1:- Create a new project using File → new project... ↵

Step 2:- Choose Java from categories pane/area & then select java application from projects which is placed at right pane & finally, click on Next.

Step 3:- Type a suitable Project name in PROJECT NAME section, and change the class name within create main class section, and then finally, click on FINISH.

[v] create main class [Rani.Rani]
Package name — Classname [good practice to change]

Step 4:- WAP logic within main() function & execute the program using run(▶) placed on toolbar.

NOTE:-

① We can also run the program by right clicking on program & choose run file.
or
We can also run the program by right clicking on Program filename placed within project section & choose run file.

JULY 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S AUGUST 2008

2008 JULY 29
WEEK 31 • 211-155 TUESDAY

② We can store multiple java files within a single project.

A file within project is created by right clicking on package name within project section & then selecting new.

After selecting new, choose java class... ↵

Type Suitable classname & click on FINISH.

Write the main() function within class & specify the logic of program within main().

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S AUGUST 2008

FUNCTION :- (Non-static function)

Case I :- User defined function as well as main() function is specified within same class.

```
//prog of case I
class Exammm
{
    void sum(int a, int b)
    {
        int r;
        r = a + b;
        System.out.println("sum is" +r);
    }
    public static void main(String args[])
    {
        Exammm m = new Exammm();
        m.sum(15, 70);
    }
}
```

```
// create function sum() & product().
import java.io.*;
class Exam
{
    void product(int a, int b)
    {
        System.out.println("Product is" +(a*b));
    }
    void sum(int a, int b)
    {
```

```
        int r;
        r=a+b;
        System.out.println("sum is"+r);
    }
    public static void main(String args[])throws
                                        IOException
    {
        BufferedReader br=new BufferedReader(new
        Exam m= new Exam();
        System.out.println("Enter two nos.");
        int x,y;
        x=Integer.parseInt(br.readLine());
        y=Integer.parseInt(br.readLine());
        m.sum(x,y);
        m.product(x,y);
    }
}
```

Case II :-
User defined function are specified in separate class & main() function in different class.

```
// Example of non-static function
class Garg
{
    void product(int a,int b)
    {
        System.out.println("Product is"+(a*b));
    }
```



```
    void sum(int a,int b)
    {
        int r;
        r=a+b;
        System.out.println("sum is"+r);
    }
}
InputStreamReader(System.in));
class Exam
{
    public static void main(String args[])throws IO
                                        Exception
    {
        BufferedReader br=new BufferedReader(new Input
        Garg m = new Garg();          StreamReader(System.in))
        System.out.println("Enter two nos.");
        int x,y;
        x=Integer.parseInt(br.readLine());
        y=Integer.parseInt(br.readLine());
        m.sum(x,y);
        m.product(x,y);
    }
}
```

NOTES

13/1/15

STATIC FUNCTION:-

A function which is preceeded with keyword 'static' is termed as static function & it is accessed using class name.

Syntax:- classname.functionname();

Case I:- Static function & main() function are specified within same class.

```
// Example of static function
import java.io.*;
class Exam
{
    static void product(int a, int b)
    {
        System.out.println("Product is" +(a*b));
    }
    static void sum(int a, int b)
    {
        int r;
        r = a+b;
        System.out.println("sum is" +r);
    }
public static void main(String args[]) throws
                                    IOException
{
    BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
```



```
    System.out.println("Enter two nos.");
    int x, y;
    x = Integer.parseInt(br.readLine());
    y = Integer.parseInt(br.readLine());
    sum(x,y);          //Exam.sum(x,y);
    product(x,y); //Exam.product(x,y);
    }
}
```

Case II:-

Static function is specified in separate class, while main() function is specified in different class.

```
//Example of static function
import java.io.*;
class Garg
{
    static void product(int a, int b)
    {
        System.out.println("Product is" +(a*b));
    }
    static void sum(int a, int b)
    {
        int r;
        r = a+b;
        System.out.println("sum is" +r);
    }
}
class Exam
```

```
{
    public static void main(String args[]) throws IOException
    {
        BufferedReader br = new BufferedReader(new
                            InputStreamReader(System.in));
        System.out.println("Enter two nos.");
        int x, y;
        x = Integer.parseInt(br.readLine());
        y = Integer.parseInt(br.readLine());
        Garg.sum(x, y);
        Garg.product(x, y);
    }
}
```

Notes

Some examples of function :-

Q. WAP to pass a string into function

Prog :-

```
//prog to pass a string into function
import java.io.*;
class Happy
{
    void display(String s)
    {
        System.out.println("String is "+s);
    }
}
class Exam
{
    public static void main(String args[]) throw
                                        IOException
    {
        BufferedReader br = new BufferedReader(new
                            InputStreamReader(System.in));
        System.out.println("Enter a string");
        String ss;
        ss = br.readline();
        Happy gg = new Happy();
        gg.display(ss);
    }
}
```

Notes

OR

```
//using static function
```

08 AUGUST 2008
FRIDAY 221-145 • WEEK 32

```
import java.io.*;
class Happy
{
    static display(String s)
    {
        System.out.println("String is"+s);
    }
}
class Exam
{
    public static void main(String args[])throws
                                        IOException
    {
        BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
        System.out.println("Enter a String");
        String ss;
        ss= br.readline();
        Happy.display(ss);
    }
}
```

NOTES

AUGUST M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008
WEEK 32 • 222-144 AUGUST SATURDAY 09

Q. WAP to pass one dimensional array to function.

Prog:- //prog to pass 1-D array to function

```
import java.io.*;
class Happy
{
    int greatest(int n[])
    {
        int g = n[0];
        for(int i=1; i<n.length; ++i)
        {
            if(g<n[i])
            {
                g= n[i];
            }
        }
        return(g);
    }
}
class Exam
{
    public static void main(String args[])throws
                                        Exception
    {
        BufferedReader br= new BufferedReader(new
                        InputStreamReader(System.in));
        int m[]= new int[10];
        System.out.println("Enter 10 elements");
        for(int c=0; c<10; ++c)
        {
            m[c]=IntegerparseInt(br.readline());
        }
```

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S SEPTEMBER

11 AUGUST 2008
MONDAY 224-142 • WEEK 33

```
        Happy gg = new Happy();
        int gr = gg.greatest(n);
        System.out.println("greatest value="+gr);
    }
}
```

Or

```
//using static
import java.io.*;
class Happy
{
    static int greatest(int n[])
    {
        int g = n[0];
        for(int i=1; i<n.length; ++i)
        {
            if(g<n[i])
                g=n[i];
        }
        return(g);
    }
}
class Exam
{
    public static void main(String args[]) throws IOExcept
    {
        BufferedReader br = new BufferedReader(new
                                InputStreamReader(System.in));
```

AUGUST 2008 | M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S | SEPTEMBER 2008

2008
WEEK 33 • 225-141 AUGUST TUESDAY 12

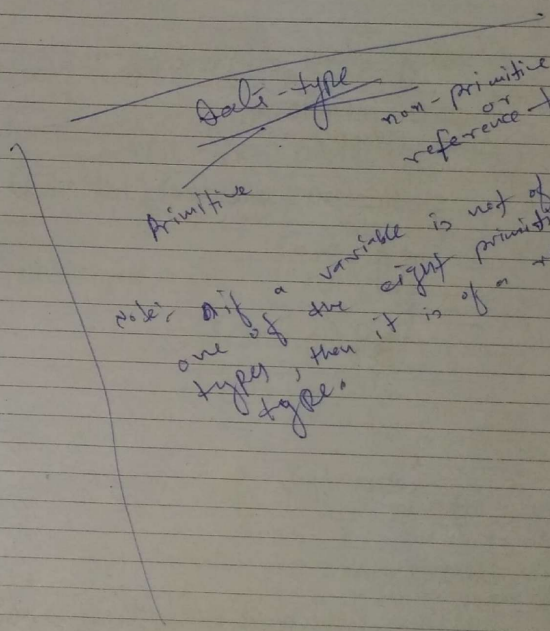
```
        int n[] = new int[10];
        System.out.println("Enter 10 elements");
        for(int c=0; c<10; ++c)
        {
            n[c] = Integer.parseInt(br.readline());
        }
        int gr = happy.greatest(n);
        System.out.println("Greatest value="+gr);
    }
}
```

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S | SEPTEMBER 2008

Q. WAP to pass 2-D array using non-static function.

Prog:
```
//prog to pass 2-D array
import java.io.*;
class Happy
{
  void display(int n[][],int siz1,int siz2)
  {
    for(int i=0;i<siz1;++i)
    {
      for(int j=0;j<siz2;++j)
      {
        System.out.print(n[i][j]+"\t");
      }
      System.out.println();
    }
  }
}
class Exam
{
  public static void main(String args[]) throws IOException
  {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int n[][] = new int[3][4];
    System.out.println("Enter elements in n");
    for(int c=0;c<3;++c)
    {
      for(int d=0;d<4;++d)
      {
        n[c][d]=Integer.parseInt(br.readline());
      }
    }
    Happy gg = new Happy();
    gg.display(n,3,4);
  }
}
```

Note: (1) Unlike local variables, which are not automatically initialized, every field has a default initial value. An attempt to use an uninitialized local variable causes compilation error.

(2) set (i.e. assign values to) or get (i.e. obtain the value of) methods can access the private instance variables. The name of the methods not need to begin with set and get but this naming convention is recommended and is convention for special Java components called JavaBeans.

16/1/15

CLASS:-

It is a user-defined datatype that contains two types of members namely datamember / field / instance variable and member function / method. Class is an important concept of java through which java language achieves encapsulation as well as data hiding (or information hiding) features of OOPS (Object Oriented Programming Language).

In Java, a class wraps its members namely field and method together in a single unit. The process of wrapping of members in a single unit is termed as Encapsulation.

The two members specified within class in Java is as follows:-

(1) Data member / field / Instance variable
(2) Member function / Method

(1) Data member / field / Instance variable:-

The variable specified within class is termed as field / instance variable / data member of class.

Notes

18 AUGUST 2008 MONDAY
231-135 • WEEK 34

② Member function/ Method :-

The function specified within class is termed as member function/method.

Syntax :-

```
access-specifier class classname
{
    access-specifier field1;
    access-specifier field2;
    :
    access-specifier fieldn;
    access-specifier method1;
    access-specifier method2;
    :
    access-specifier methodn;
}
```

NOTE :-

① Unlike C++, the class of Java is not terminated with semicolon(;).

② In Java, access-specifier represents the accessibility of class, field, method, etc. from outside the class or program. It determines how other parts of the program can access the members of the class.
There are following four access-specifier or access modifiers in Java :-

(i) private
(ii) protected
(iii) public
(iv) friendly / default → not keyword

(keywords)

AUGUST 2008 | SEPTEMBER 2008

Note: each object of a class contains its own copy of an attribute/field

2008 | 19 AUGUST TUESDAY
WEEK 34 • 232-134

③ When no access specifier is specified, default friendly access specifier is considered.

Q. WAP to explain the example of class.

Prog :-

```
//prog to explain the example of class
class Demo
{
    int x, y; //two fields x & y have friendly access specifier
}
class MyDemo
{
    public static void main(String args[])
    {
        Demo d = new Demo();
        d.x = 25;
        d.y = 40;
        System.out.println("x is "+d.x+"\ny is "+d.y);
    }
}
```

//Another example of class

```
class Demo
{
    private int x, y; //x & y are instance variable which has private access
    void input(int a, int b)
    {
        x = a;
        y = b;
    }
```

SEPTEMBER 2008

```
    void display()
    {
        System.out.println("x is"+x+"\n y is"+y);
    }
}
class MyDemo
{
    public static void main(String args[])
    {
        Demo d = new Demo();
        d.input(50, 60);
        d.display();
    }
}
```

Q. WAP to create a class rectangle to calculate & print the area & perimeter of rectangle.

Prog:-
```
//prog to create a class rectangle
import java.util.Scanner;
class Rectangle
{
    private int len, bre;
    void input(int l, int b)
    {
        len = l;
        bre = b;
    }
    public void display()
    {
        System.out.println("Length is"+len+"\nbreadth is"+bre);
    }
```



```
    public void getArea()
    {                    → camelcase
        int a;
        a = len * bre;
        System.out.println("Area is"+a);
    }
    public void getPeri()
    {
        System.out.println("Perimeter is"+(2*(len+bre
    }
}
class DemoRect
{
    public static void main(String args[])
    {
        Rectangle r = new Rectangle();
        Scanner s = new Scanner(System.in);
        int x, y;
        System.out.println("Enter length and breadth");
        x = s.nextInt();
        y = s.nextInt();
        r.input(x, y);
        r.display();
        r.getArea();
        r.getPeri();
    }
}
```

27 AUGUST 2008
WEDNESDAY

=) Private member function

=) Nesting of member function

=) array of object

=) static data member and member function within class

2008 AUGUST 28
WEEK 35 • 241-125 THURSDAY

PASSING OBJECT AS ARGUMENT IN JAVA:-

Q. WAP to add two complex no.

Prog: //prog to add two complex nos.

```
import java.util.Scanner;
class Complex
{
    private int real, imag;
    void input(int re, int im)
    {
        real = re;
        imag = im;
    }
    public void display()
    {
        System.out.println(real+"+"+imag+"i");
    }
    public void add(Complex c1, Complex c2)
    {
        real = c1.real + c2.real;
        imag = c1.imag + c2.imag;
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Scanner s = new Scanner(System.in);
        int x, y, m, n;
        System.out.println("Enter four values");
```

```
x = s.nextInt();
y = s.nextInt();
m = s.nextInt();
n = s.nextInt();
Complex cc1 = new Complex();
Complex cc2 = new Complex();
Complex cc3 = new Complex();
cc1.input(x, y);
cc2.input(m, n);
cc1.display();
cc2.display();
cc3.add(cc1, cc2);
cc3.display();
}
}
```

Q1. WAP to create a class

NOTES

returning object

NOTES

CONSTRUCTOR :-

It is a special type of member function whose name is same as class name. The main work of constructor is to initialise the object of the class. A constructor has no return type, not even void, but can take argument/parameter. It is invoked/ call during object creation of object.

There are following types of constructor in Java :-

1. Default / no argument
2. Argumented / Parameterised constructor
3. Copy constructor

(1) Default / No argument :-

A constructor having no argument is called default constructor or no argument constructor. It means, a constructor that does not take any argument / parameter is called default constructor. A default constructor is automatically invoked when the object of a class is created.

Ex:-
```
class Example
{
    Example() // default const.
    {
    }
}
```

10 SEPTEMBER 2008
WEDNESDAY 254-112 • WEEK 37

Q. WAP to explain the example of default constructor.

Prog:-
```
//example of default constructor
class Demo
{
    private int x, y;
    Demo() //default const.
    {
        x=15;
        y=25;
    }
    void display()
    {
        System.out.println("x is"+x+"\n y is"+y);
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Demo d = new Demo();
        d.display();
    }
}
```

Note:- If we declare any Constructors for a class, the java compiler will not create a default constructor for that class.

2008
WEEK 37 • 255-111 SEPTEMBER THURSDAY 11

② Parameterised Constructor:-

A constructor having arguments is called parameterised / argumented constructor. It means, a constructor that takes arguments/ parameters is considered as parameterised/ argumented constructor. A parameterised constructor will be also invoked at the time of object creation.

Q. WAP to explain the example of parameterised constructor.

Prog:-
```
//example of parameterised constructor
class Demo
{
    private int x, y;
    Demo(int a, int b)//parameterised const.
    {
        x=a;
        y=b;
    }
    void display()
    {
        System.out.println("x is"+x+"\n y is"+y);
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Demo d = new Demo(50, 60);
        d.display();
    }
}
```

(3) Copy Constructor:-

The process of initialising an object with another object through constructor is called copy constructor.

Q. WAP to explain the example of copy const.

Prog:- //example of copy constructor

```
class Demo
{
    private int x,y;
    Demo(int a, int b) //parameterised const.
    {
        x=a;
        y=b;
    }
    void display()
    {
        System.out.println("x is "+x+"\n y is "+y);
    }
    Demo(Demo dd) //copy const.
    {
        x=dd.x;
        y=dd.y;
    }
}
class MyDem
{
    public static void main(String args[])
```



```
    {
        Demo d = new Demo(50,60);
        d.display();
        Demo d1 = new Demo(d);
        d1.display();
    }
}
```

CONSTRUCTOR OVERLOADING / MULTIPLE CONSTRUCTORS IN A CLASS:-

The process of specifying multiple constructors (more than one constructor / zero or more constructor) within a single class is called Constructor Overloading.

Multiple constructors are identified a/c to no. of arguments & their types.

Q. WAP to explain the example of constructor overloading.

Prog: //example of Constructor overloading.

```
class Example
{
    private int x,y,z;
    Example(int a, int b)
    {
        x=a;
        y=a;
        z=b;
    }
```

```
Example(int m)
{
    x = m;
    y = m;
    z = m;
}
Example()
{
    x = 100;
    y = 200;
    z = 300;
}
Example(int a, int b, int c)
{
    x = a;
    y = b;
    z = c;
}
void display()
{
    System.out.println("x is"+x+"\ny is"+y+"\nz is"+z);
}
}
class MyExam
{
public static void main(String args[])
{
    Example e1 = new Example();
    Example e2 = new Example(50, 60);
    Example e3 = new Example(500, 1500, 1000);
```



```
    Example e4 = new Example(5000);
    e1.display();
    e2.display();
    e3.display();
    e4.display();
}
}
```

this :-

It is a keyword which can be used inside any method/function to refer the current object. It is a reference variable that refers to the current object.

There are following uses of 'this' keyword.

1. This keyword can be used to refer current class object.

```
//Example of 'this' keyword
class Mukt
{
    private int x, y;
    void input(int a, int b)
    {
        this.x = a;
        this.y = b;
    }
    void display()
    {
        System.out.println("x is"+x+"\n Y is"+y);
    }
```

```
..atic void main(String args[])

.. m= new Mukt();
..nput(12,50);
..display();

}
```

② 'this' keyword can be used to resolve instance variable hiding.

Within a method, when instance variable & local variable has same name, then local variable hides instance variable, it means only local variable will be activated/enabled.

```
//hiding of instance variable
class Mukt
{
    private int x,y;
    void input(int x, int y)
    {
        x=x;
        y=y;
    }
    void display()
    {
        System.out.println("X is"+x+"\n Y is"+y);
    }
    public static void main(String args[])
    {
        Mukt m= new Mukt();
        m.input(12,50);
        m.display();
    }
```

Output
x is 0
y is 0

Note:- We can resolve the hiding of instance variable with local variable by using 'this' keyword.

```
//Resolving hiding of instance variable
class Mukt
{
    private int x,y;
    void input(int x, int y)
    {
        this.x=x;
        this.y=y;
    }
    void display()
    {
        System.out.println("X is"+x+"\n Y is"+y);
    }
    public static void main(String args[])
    {
        Mukt m=new Mukt();
        m.input(12,50);
        m.display();
    }
}
```

③ this() can be used to invoke current class constructor.

```
//invoking current class constructor
class Mukt
{
    private int x,y;
    Mukt()
    {
        System.out.println("Hello..");
```

28 MON NOV '05

```
Mukt(int x,int y)
{
    this();
    this.x = x;
    this.y = y;
}
void display()
{
    System.out.println("X is"+x+"\nY is"+y);
}
public static void main(String args[])
{
    Mukt m = new Mukt(12,70);
    m.display();
}
}
```

Output
Hello...
x is 12
y is 70

```
//Another example
class Mukt
{
    private int x,y,z;
    Mukt(int a,int b)
    {
        x = a;
        y = b;
    }
    Mukt(int x,int y,int z)
    {
        this(x,y);
        this.z = z;
    }
    void display()
    {
        System.out.println("X is"+x+"\nY is"+y+"\
```

IMP. NOTES

29 TUE

```
public static void main(String args[])
{
    Mukt m = new Mukt(12,70,50);
    m.display();
}
}
```

④ 'this' keyword can be used to invoke current class method.

```
//Invoking current class method
class Mukt
{
    void display()
    {
        System.out.println("Hi");
    }
    void show()
    {
        this.display();
    }
    void disp()
    {
        show();
    }
    public static void main(String args[])
    {
        Mukt m = new Mukt();
        m.disp();
    }
}
```

IMP. NOTES

:)

30 WED NOV '05

⑤ 'this' keyword can be passed as an argument in method. It is mainly seen in event handling.

```
//
class Mukt
{
    private int x, y;
    Mukt (int a, int b)
    {
        x=a;
        y=b;
    }
    void display(Mukt m)
    {
        Sopln("X is"+m.x+"\n y is"+m.y);
    }
    void show(Mukt t)
    {
        display(this);
    }
    public static void main(String args[])
    {
        Mukt m1=new Mukt(50,70);
        m1.show(m1);
    }
}
```

IMP. NOTES

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

2008

21/1/15

WEEK 38 • 262-104

SEPTEMBER THURSDAY 18

INHERITANCE :-

The process of acquiring/taking the property of one class by another class is known as inheritance/derivation. The class which acquires the property is known as child class/ derived class/ sub class & the class which gives the property, i.e the class whose property is acquired is termed as parent class/ base class/ super class.

Inheritance is an important concept of Java ~~CPP~~, which ~~acquires~~ achieves reusability. With the help of inheritance, we can reuse the all or partial property of an existing class.

There are following types of inheritance in Java :-

1. Single Inheritance
2. Multi-Level inheritance
3. Hierarchical inheritance

NOTES :-

(1) Java does not support multiple inheritance but the structure of multiple inheritance can be achieved with the help of interface ~~inheritance~~ concepts of Java.

(2) A child class can acquire all or partial property of a parent class & can also contains its own property.

T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S | OCTOBER 2008
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

19 SEPTEMBER 2008
FRIDAY 263-103 • WEEK 38

(S) An existing class can be considered as base class & the syntax of base class will be same as normal class.

(A) In Java, the keyword extends is used to derive/inherit a base class into derived class & the syntax for derive class is as follows:-

Syntax:-
access specifier class derive classname extends base classname
{
}

Note:-
The member preceded with private access specifier of base class is not inherited/derived while the member with public, protected & friendly access specifier can inherit/derive & hold the same access-specifier as in base class.

NOTES

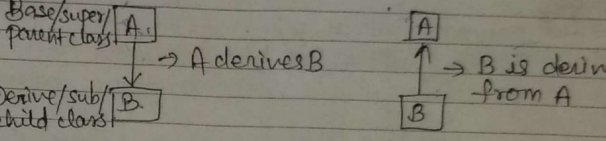
SEPTEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

e used

2008 22/1/15
WEEK 38 • 264-102 SEPTEMBER SATURDAY 20

① SINGLE INHERITANCE:-

The process of deriving/inherit a class with a single base class is known as single inheritance.



Q. WAP to explain the example of single inheritance.

Prog:-
```
//example of single inheritance in Java
class Example
{
    Private int x;
    int y;
    void input(int a, int b)
    {
        x=a;
        y=b;
    }
    void display()
    {
        System.out.println("x is"+x+"\ny is"+y);
    }
    int getX()
    {
        return(x);
    }
}
```

NOTES

SUNDAY 21

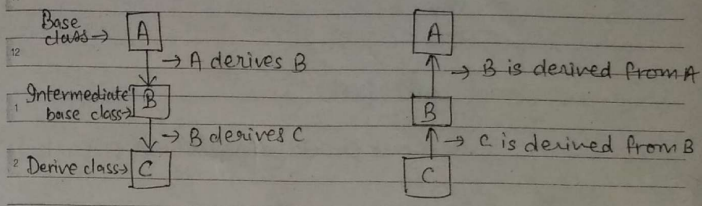
M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S OCTOBER 2008

```
class Sum extends Example
{
    private int r;
    void getSum()
    {
        r = y + getX();
        System.out.println("sum is"+r);
    }
}
class MyDemo
{
    public static void main(String args[])
    {
        Sum s = new Sum();
        s.input(25, 15);
        s.display();
        s.getSum();
    }
}
```

NOTES

(2) MULTI-LEVEL INHERITANCE:-

The process of deriving a class with another derived class is known as multi-level inheritance.



Q. WAP to explain the example of multilevel inheritance.

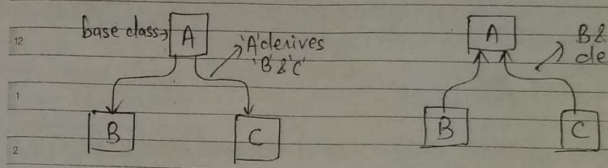
Prog:-

```
//explain the example of multilevel inheritance
class Student
{
    protected int roll;
    void getRoll(int r)
    {
        roll = r;
    }
    void putRoll()
    {
        System.out.println("Roll is" + roll);
    }
}
class Marks extends Student
```

```
}
    protected int m1, m2, m3;
    void getMarks(int a, int b, int c)
    {
        m1 = a;
        m2 = b;
        m3 = c;
    }
    void putMarks()
    {
        System.out.println("M1 is" + M1 + "\nM2 is" + m2
                           + "\nM3 is" + m3);
    }
}
class Result extends Marks
{
    private int tot;
    void getTotal()
    {
        tot = m1 + m2 + m3;
        System.out.println("Total marks = " + tot);
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Result rr = new Result();
        rr.getRoll(2);
        rr.putRoll();
        rr.getMarks(10, 15, 20);
        rr.putMarks();
        rr.getTotal();
    }
}
```


(3) Hierarchical Inheritance:-

The process of deriving multiple classes (two or more classes) with a single base class is known as hierarchical inheritance.



Q. WAP to explain the example of hierarchical inheritance.

Prog:-

```
//example of hierarchical inheritance in Java
class Example
{
    protected int x, y;
    void input(int a, int b)
    {
        x = a;
        y = b;
    }
    void display()
    {
        System.out.println("x is" + x + "\ny is" + y);
    }
```

```
class Sum extends Example
{
    private int s;
    void getSum()
    {
        s = x + y;
        System.out.println("Sum is" +s);
    }
}
class Product extends Example
{
    private int p;
    void getProduct()
    {
        p = x * y;
        System.out.println("Product is" + p);
    }
}
class Dem
{
    public static void main(String args[])
    {
        Sum ss = new Sum();
        ss.input(50, 60);
        ss.display();
        ss.getSum();
        Product pp = new Product();
        pp.input(5, 10);
        pp.display();
        pp.getProduct();
    }
}
```

NOTES

METHOD OVERRIDING :-

NOTES

29 SEPTEMBER 2008
MONDAY 273-093 • WEEK 40

Q. WAP to explain the example of method over-riding.

Prog:-
```
//example of method over-riding
class Demo
{
    void display()
    {
        System.out.println("Welcome Demo");
    }
}
class Example extends Demo
{
    void display()
    {
        System.out.println("Welcome Example");
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Example e = new Example();
        e.display();
    }
}
```

NOTES

SEPTEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 SEPTEMBER 30
WEEK 40 • 274-092 TUESDAY

CONSTRUCTOR IN INHERITANCE:-

Unlike member function, constructor cannot be inherited, i.e. a constructor specified within base class is not inherited in derived class.

In inheritance, constructor are invoked/accessed/call a/c to the Hierarchy/derivation of inheritance. It means, the constructor specified in base class is invoked before the constructor specified in derived class.

```
//Example of Constructor in inheritance
class Demo
{
    Demo()
    {
        System.out.println("Hi Demo");
    }
}
class Example extends Demo
{
    Example()
    {
        System.out.println("Hi Example");
    }
}
class Diksha extends Example
{
    Diksha()
```

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S OCTOBER 2008

```
    {
        System.out.println("Hi Diksha");
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Diksha h = new Diksha();
    }
}
```

'Super' Keyword :-

It is an important keyword of java which is used for following purposes :-

1. calling / invoking base class constructor
2. accessing base class method in the case of method over riding.
3. accessing the data member / instance variable of base class when both base class & derived class datamember has same name

(1) calling / invoking base class constructor :-

The first use of 'super' keyword is to invoke / call base / super class constructor from derive class constructor.

Syntax :-

```
    super(arguments);
```

```
//example of super keyword
class Demo
{
    private int x,y;
    Demo(int a, int b)
    {
        x=a;
        y=b;
    }
    void display()
    {
        System.out.println("x is "+x+"\n y is "+y);
    }
}
class Example extends Demo
{
    private int z;
    Example(int m, int n, int o)
    {
        super(m,n);
        z=o;
    }
    void show()
    {
        System.out.println("z is "+z);
    }
}
class MyDem
```

```
{
    public static void main(String args[])
    {
        Example e = new Example(50,90,40);
        e.display();
        e.show();
    }
}
```

② accessing base class method in the case of method overriding:-

The second use of 'super' keyword is to access the method of base class when method over-riding occurs. As we know, in method overriding, we can only access the method of derive class.

Syntax:- super.methodname();

```
// example of super class by second method
class Demo
{
    void display()
    {
        System.out.println("Hello Demo");
    }
}
class Example extends Demo
```



```
{
    void display()
    {
        super.display();
        System.out.println("Hello Example");
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Example e = new Example();
        e.display();
    }
}
```

Output:- Hello Demo
Hello Example

③ accessing the data member/instance variable of base class when both base class & derived class data member has same name:-

'super' keyword is also used to access the data member of base class when the data member of derive class is same as base class.

SUNDAY 5

Syntax:- super.datamembername;

06 OCTOBER 2008
MONDAY 280-086 • WEEK 41

```
//example of super keyword by third method
class Demo
{
    int x;
}
class Example extends Demo
{
    int x;
    void input(int a, int b)
    {
        super.x = a;
        x = b;
    }
    void display()
    {
        System.out.println("Base x is" + super.x + "\n
                                Derive x is" + x);
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Example e = new Example();
        e.input(50,100);
        e.display();
    }
}
```


27/1/15

2008 OCTOBER 07
WEEK 41 • 281-085 TUESDAY

'final' Keyword :-

This keyword is an important keyword of java which is used for following purposes :-

1. To specify constant/final value.
2. To prevent a method from being overrider
3. To prevent a class from being inherited.

1. To specify constant/final value :-

The first use of 'final' keyword is to specify constant/final values within java program. The value specified using 'final' keyword can't be changed or modified during execution of program.

Syntax :- final datatype variable-name = value;
(variable-name written in capital letter to differentiate from normal variable.)

Ex:- final int NUM = 25;

```
// prog to explain the example of final keyword
// to specify constant/final value
class Demo
{
    final int NUM = 25;
```

```
public static void main(String args[])
{
    final int NUM = 25;
    System.out.println("Num is" +NUM);
    NUM = 100; //error
}
}
```

(2) To prevent a method from being overriden:-

The second use of 'final' keyword is to prevent a method of base class from being overriden. With the help of 'final' keyword, we can prevent the method of base class to be overriden in derived class. We have to simply preceed 'final' keyword in the method of base class.

```
//prog to explain the example of final
//keyword to prevent from method overriding
class Demo
{
    final void display()
    {
        System.out.println("welcome Demo");
    }
}
class Example extends Demo
{
    void display()
    {
        System.out.println("welcome example");
    }
}
class DemExam
{
    public static void main(String args[])
    {
        Example e = new Example();
        e.display();
    }
}
```

(3) To prevent a class from being inherited :-

The third use of 'final' keyword is to prevent a class from being inherited. With the help of 'final' keyword, we can prevent a delive class to inherit the base class. We have to simply specify the keyword 'final' before the base class.

```
//prog to explain the example of final keyword
//to prevent a class from being inherited.
final class Demo
{
    void display()
    {
        System.out.println("welcome Demo");
    }
}
```

10 OCTOBER 2008
FRIDAY 284-082 • WEEK 41

```
class Example extends Demo
{
    void display()
    {
        System.out.println("Welcome Example");
    }
}
class DemExam
{
    public static void main(String args[])
    {
        Example e = new Example();
        e.display();
    }
}
```

NOTES

OCTOBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 OCTOBER 11
WEEK 41 • 285-081 SATURDAY

OBJECT / INSTANCE :-

The variable of type class is called object / instance of class.

or

An instance of class is called object.

Syntax to create object in java :-

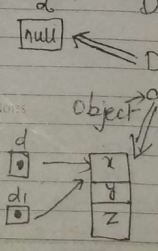
```
classname objectname = new classname();
        'or'
classname objectname;
obj-name = new classname();
```

Syntax to create reference of class :-

```
classname reference-name;
```

Ex :-

```
class Demo
{
    int x,y,z;          → object
}
Demo d = new Demo();
        'or'
Demo d;                 → reference
d = new Demo();
```



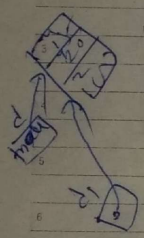
SUNDAY 12

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

NOVEMBER 2008

Q. WAP to explain the example of reference in Java.

Prog:-
```
//example of reference in java
class Demo
{
    int x,y,z;
}
class MyDem
{
    public static void main(String args[])
    {
        Demo d = new Demo();
        d.x = 15;
        d.y = 25;
        d.z = 100;
        Demo d1 = d;
        System.out.println("x is"+d.x+"\ny is"+d.y
                +"\nz is"+d.z);
        System.out.println("x is"+d1.x+"\ny is"+d1.y+
                "\nz is"+d1.z);
    }
}
```



Note:- A base class reference can store object of its own class as well as the object of successor class within it. But, whenever we store the object of successor/sub/derive classes in base class reference, then we are only

permitted to access those member of derive class which is inherited from base class. It means, the base class reference cannot access the original member of derived class.

This problem is dissolved with the help of DMD (Dynamic Method Dispatch).

Q. WAP to explain the example of base class reference which holds base class object as well as derive class object.

Prog:-
```
// base class reference which holds base class
// object as well as derive class object.
class Demo
{
    void show()
    {
        Sopln("Demo show");
    }
}
class Exam extends Demo
{
    void show()
    {
        sopln("Exam show");
    }
    void display()
```

```
        }
        Sopln("Exam display");
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Demo dd;
        Demo d = new Demo();
        Exam e = new Exam();
        dd = d;
        dd.show();  → Demo show
        dd = e;
        dd.show();  → Exam show
        dd.display(); // error
    }
}
```

Notes

DMD (Dynamic Method Dispatch):-

It is an important feature of java through which run-time polymorphism is achieved. Dynamic Method Dispatch is a technique through which java resolve the calling of overriden method at run-time rather than compile-time. In compile time, calling of overriden method is resolved using 'super' keyword

```
//prog to explain the example of DMD
class Demo
{
    void show()
    {
        Sopln("Demo show");
    }
}
class Exam extends Demo
{
    void show()
    {
        Sopln("Exam show");
    }
}
class Example extends Exam
{
    void show()
    {
```

Notes

```
        SopIn("Example show");
    }
}
class MyDem
{
public static void main(String args[])
{
    Demo dd;
    Demo d = new Demo();
    Exam e = new Exam();
    Example e1 = new Example();
    dd = d;
    dd.show();
    dd = e;
    dd.show();
    dd = e1;
    dd.show();
}
}
```

NOTES

ABSTRACT CLASS :-

The process of specifying generalised form of a class without providing implementation of some or all members of a class is called Abstract class. 'abstract' keyword is used to specify/define an abstract class.

Abstract class must contain atleast one abstract method. An abstract method is a method that contain only name and method signature, and preceeded with keyword 'abstract'. An abstract method does not have method body, it contains only declaration/prototype of method.

In Java, a class which contains atleast one abstract method is considered as abstract class / abstract super class / dummy class. An abstract class can also contain instance method, instance variable/data member, constructor as well as final method.

The abstract method is implemented in derive class/sub-class. It means, a sub class has full responsibility to implement all abstract method or defining the body of abstract method. The subclass which implements/uses the abstract class is called concrete class, and the method which

NOTES SUNDAY 19

specify the body for abstract method is called concrete method.

A derive class can either implement all abstract method or declare itself as abstract. The abstract method which is not implemented by sub-class will be implemented by next level sub-class in class hierarchy.

NOTE:-

(1) We cannot make a constructor, static method & final method as abstract.

(2) We cannot create the object of an abstract class. However, we can create the reference of abstract class.

Q. WAP to explain the example of abstract class.

Prog:- //example of abstract class.

```
abstract class Shape
{
    Protected int x, y;
    Shape(int a, int b)
    {
        x = a;
        y = b;
    }
    abstract void getArea();
}
class Rectangle extends Shape
{
    Rectangle(int m, int n)
    {
        super(m, n);
    }
    void getArea()
    {
        System.out.println("Area of rectangle=" +(x*y));
    }
}
class circle extends Shape
{
    Circle(int m)
    {
        super(m, 0);   // zero
    }
```

```
    void getArea()
    {
        Sopln("Area of Circle is "+(Math.PI*x*x));
    }
}
class Triangle extends Shape
{
    Triangle(int m, int n)
    {
        super(m,n);
    }
    void getArea()
    {
        Sopln("Area of Triangle="+((x*y)/2));
    }
}
class ShapeDemo
{
    public static void main(String args[])
    {
        Rectangle rr = new Rectangle(5,10);
        Circle cc = new Circle(8);
        Triangle tt = new Triangle(4,6);
        rr.getArea();
        cc.getArea();
        tt.getArea();
    }
}
```

or


```
class ShapeDemo
{
    psvm(String args[])
    {
        Rectangle rr = new Rectangle(5,10);
        Circle cc = new Circle(8);
        Triangle tt = new Triangle(4,6);
        Shape s;
        s = rr;
        s.getArea();
        s = cc;
        s.getArea();
        s = tt;
        s.getArea();
    }
}
```

or

```
class ShapeDemo
{
    psvm(String args[])
    {
        Shape s;
        s = new Rectangle(5,10);
        s = new Circle(8);
        s = new Triangle(4,6);
    }
}
```

INTERFACE :-

The process of specifying / representing full abstract form / generalise form is called interface. Interface is the way / form through which Java represents the concept of fully abstract class.

With the help of interface, Java achieves multiple inheritance. In Java, interface keyword is used to create an interface. We can specify instance variable and method / function prototype within interface as member.

The interface variable / data member of instance is by default public, static & final, while the method prototype within the interface is considered as abstract and public defaultly.

Note 1 :- An interface can be utilised / implemented by any no. of classes and a single class can also utilised / implement multiple (more than one) interfaces.

Note 2 :- In Java, implements keyword is used to utilise / implement the interface.

NOTES

27 OCTOBER 2008
MONDAY
301-065 • WEEK 44

Syntax to create an Interface:-

```
interface interface-name
{
    instance variable/field-1;
    instance variable/field-2;      } By default, public, static & final
    instance
    instance variable/field-n;
    method prototype-1;
    method prototype-2;             } By default public & abstract
    method prototype-n;
}
```

Note:- By default, access-specifier for interface is public, while the class is friendly / defaultly.

Ex:-
```
interface Demo
{
    void display();
    void show();
}
```

Ex:-
```
interface Example
{
    float x = 3.14f;
    void display();
    void getArea();
}
```

OCTOBER 2008 | M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 OCTOBER 28
WEEK 44 • 302-064 TUESDAY

IMPLEMENTING/UTILISING AN INTERFACE:-

An interface can be utilised / implemented by one or more classes using implements keyword. The class who implements/utilises the interface must define the body of all abstract methods specified within interface. If a class performs partial utilisation of abstract method, then that class must be declared as abstract otherwise compiler produce an error.

NOTE:- Like abstract class, we cannot create the object of an interface. However, we can create the reference of an interface.

Syntax to implement/utilise an interface:-
```
class classname implements interfacename
{
    ...
}
```

Ex:-
```
class XYZ implements Demo
{
    ...
}
```

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S NOVEMBER 2008

29 OCTOBER 2008
WEDNESDAY
303-063 • WEEK 44

Q. WAP to explain the example of interface.

Ans:
```
//example of interface
interface Demo
{
    void show();
    void display();
}
class XYZ implements Demo
{
    public void show()
    {
        Sopln("welcome show");
    }
    public void display()
    {
        Sopln("Bye Display");
    }
}
class DemoXYZ
{
    Public static void main(String args[])
    {
        XYZ m = new XYZ();
        m.show();
        m.display();
    }
}
```

Notes


```
class DemoXYZ
{
    psvm(String args[])
    {
        Demo d;
        XYZ m = new XYZ();
        d = m;
        d.show();
        d.display();
    }
}
```

OR

```
class DemoXYZ
{
    psvm(String args[])
    {
        Demo d = new XYZ();
        d.show();
        d.display();
    }
}
```

Notes

31 OCTOBER FRIDAY 2008 — 305-061 • WEEK 44

3/2/15

Specifying main() function within that class which implements an interface:-

```
// prog to specifying main() function within
// that class which implements an interface
interface Demo
{
    void show();
    void display();
}
class Sruti implements Demo
{
    public void show()
    {
        System.out.println("xyz show");
    }
    public void display()
    {
        System.out.println("xyz display");
    }
    public static void main(String args[])
    {
        Sruti s = new Sruti();
        s.show();
        s.display();
    }
}
```

NOTES

OCTOBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 NOVEMBER 01 SATURDAY — WEEK 44 • 306-060

Q. WAP to explain the example of interface specifying both instance variable & met

'or'

WAP to explain the example of implemen an interface by multiple classes.

Prog:-

```
// example of implementing an interface b
// multiple classes
interface Shape
{
    float PIE = 3.14f;
    void getArea(int x, int y);
}
class Rectangle implements Shape
{
    public void getArea(int a, int b)
    {
        System.out.println("Area of Rectangle is" (l*b));
    }
}
class Circle implements Shape
{
    public void getArea(int x, int m)
    {
        System.out.println("Area of circle is"+(PIE
    }
}
```

SUNDAY 2

```
class Sruti
{
    public static void main(String args[]
```

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S DECEMBE 2008

```
    {
        Shape s;
        Rectangle rr = new Rectangle();
        Circle cc = new Circle();
        s=rr;
        s.getArea(12,8);
        s=cc;
        s.getArea(5,7);
    }
}
```

NOTE: (related to above program):-

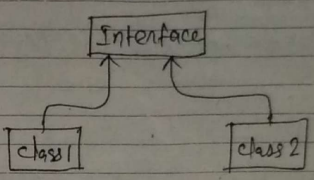
(1) Data member / instance variable / field within interface must be initialised during declaration & written in capital letter, because the instance variable within interface is by default final, static and public, otherwise compiler produces an error.

(2) The variable-name must be specified as argument in method prototype within interface, otherwise compiler produces an error.

NOTES

Implementing an Interface by multiple classes:

In Java, a single interface can be utilised implemented by multiple (two or more) classes.




```
// implementing an interface by multiple classes
interface Demo
{
    void show();
    void display();
}
class Xyz implements Demo
{
    public void show()
    {
        System.out.println("Xyz show");
    }
    public void display()
    {
        System.out.println("Xyz display");
    }
}
class Mno implements Demo
{
```

NOTES

```
    public void show()
    {
        System.out.println("Mno show");
    }
    public void display()
    {
        System.out.println("Mno display");
    }
}
class Soulti
{
    public static void main(String args[])
    {
        Demo d;
        Xyz mm = new Xyz();
        Mno nn = new Mno();
        d=mm;
        d.show();
        d.display();
        d=nn;
        d.show;
        d.display();
    }
}
```

Output: Xyz show
Xyz display
Mno show
Mno display

Implementing multiple interfaces in a single class:-

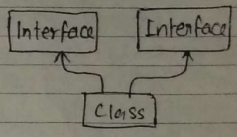
We can utilise/implement more than one interfaces by a single class by separating interfaces with comma(,).

Syntax:-

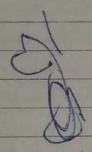
```
class classname implements interface-1, interface-2, ... interface-n
{
}
```

Ex:-

```
//prog to implement multiple interfaces in a
//single class
interface Demo
{
    void display();
}
interface Examp
{
    void show();
    void disp();
}
class Xyz implements Demo,Examp
{
    public void show()
    {
        System.out.println("xyz show");
    }
```



```
    public void display()
    {
        System.out.println("Xyz disp");
    }
}
class Srutl
{
    public static void main(String args[])
    {
        Xyz d = new Xyz();
        d.show();
        d.display();
        d.disp();
    }
}
```



NOTES

Inheriting an Interface:-

In Java, we can inherit/derive the member of one interface into another interface.

```
// prog to show the example of inheriting an interface
// using single inheritance hierarchy.
interface Demo
{
    void display();
}
interface Examp extends Demo
{
    void show();
    void disp();
}
class Xyz implements Examp
{
    public void show()
    {
        System.out.println("Xyz sohow");
    }
    public void display()
    {
        System.out.println("Xyz display");
    }
    public void disp()
    {
        System.out.println("Xyz disp");
    }
}
```

NOTES

SUNDAY 9

10 NOVEMBER 2008
MONDAY
315-051 • WEEK 46

```
class Sruti
{
    public static void main(String args[])
    {
        Xyz d = new Xyz();
        d.show();
        d.display();
        d.disp();
    }
}

// Inheriting an interface in another interface
// using multi-level hierarchy
interface Demo
{
    void display();
}
interface Example extends Demo
{
    void show();
}
interface Examp extends Example
{
    void disp();
}
class Xyz implements Examp
{
    public void show()
    {
        System.out.println("Xyz show");
    }
    public void display()
    {
        System.out.println("Xyz display");
    }
    public void disp()
    {
        System.out.println("Xyz disp");
    }
}
class Sruti
{
    public static void main(String args[])
    {
        Xyz d = new Xyz();
        d.show();
        d.display();
        d.disp();
    }
}
```

NOTES

NOVEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S 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

2008 NOVEMBER 11
WEEK 46 • 316-050
TUESDAY

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S 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 DECEMBER 2008

```
// Inheriting an interface with two or more interfaced
// Example of multiple inheritance using interface
interface Demo
{
    void display();
}
interface Example
{
    void show();
}
interface Examp extends Example, Demo
{
    void disp();
}
class Xyz implements Examp
{
    public void show()
    {
        Sopln("Xyz Helloshow");
    }
    public void display()
    {
        Sopln("Xyz display");
    }
    public void disp()
    {
        Sopln("Xyz disp");
    }
}
```



```
class Sruti
{
    public static void main(String args[])
    {
        Xyz d = new Xyz();
        d.show();
        d.display();
        d.disp();
    }
}
```

NOTE:- In Java, the structure/format for multiple inheritance with class is not possible, but that structure is possible with interface.

```
//example of inheriting an interface using hierarchical (inheritance) format
interface Demo
{
    void display();
}
interface Example extends Demo
{
    void show();
}
interface Examp extends Demo
{
    void disp();
}
```

```
class Xyz implements Examp
{
    public void display()
    {
        Sopln("Xyz display");
    }
    public void disp()
    {
        Sopln("Xyz disp");
    }
}
class Mno implements Example
{
    public void display()
    {
        Sopln("Mno display");
    }
    public void show()
    {
        Sopln("Mno show");
    }
}
class Sruti
{
    public static void main(String args[])
    {
        Xyz d = new Xyz();
        d.display();
        d.disp();
        Mno m = new Mno();
        m.display();
        m.show();
```


or

```
interface Demo
{
    void display();
}
interface Example extends Demo
{
    void show();
}
interface Examp extends Demo
{
    void disp();
}
class Xyz implements Examp, Example
{
    public void display()
    {
        Sopln("Xyz display");
    }
    public void disp()
    {
        Sopln("Xyz disp");
    }
    public void show()
    {
        Sopln("Xyz show");
    }
}
class Sruti
{
    public static void main(String args[])
    {
```

```
    Xyz d = new Xyz();
    d.display();
    d.disp();
    d.show();
  }
}
```

Partial utilisation of abstract method of interface:-

As we know, a class utilises/implements all abstract method of an interface. But, in java, it is also provision to utilise abstract method according to need. For this, we have to preceed the class with abstract keyword which implement an interface.

```
//partial utilisation of abstract method of interface
interface Demo
{
    void display();
    void show();
    void disp();
}
abstract class Xyz implements Demo
{
    public void display()
    {
        System.out.println("Xyz display");
    }
```



```
    public void show()
    {
        System.out.println("Xyz show");
    }
}
class Sruti extends Xyz
{
    public void disp()
    {
        System.out.println("Bye Bye");
    }
    public static void main(String args[])
    {
        Sruti d = new Sruti();
        d.display();
        d.show();
        d.disp();
    }
}
```

'or'

```
interface Demo
{
    void display();
    void show();
    void disp();
}
abstract class Xyz implements Demo
{
    public void display()
    {
        System.out.println("Xyz display");
    }
```

```
public void show()
{
    System.out.println("Xyz show");
}
}
class Mno extends Xyz
{
public void disp()
{
    System.out.println("Bye Bye See u Never");
}
}
class Sruti
{
public static void main(String args[])
{
    Mno d = new Mno();
    d.display();
    d.show();
    d.disp();
}
}
```

Process:- A running program is called process.
Or, A program in execution is called process & a light weighted process is called thread.

Note:- A single program can have a single process or multiple process.

```
main()
{ ---> one process
}
```

```
main()
{ sum();
  product();
  cube();
}
sum()
{
}
product()
{
}
```

⇒ 4 processes, one main and 3 child processes

MULTI-THREADING :-

The process of creating as well as execut more than one threads simultaneously/ parallely / concurrently is called multi-thread Multi-threading allows multiple task to execute concurrently within a single program By using multi-threading, we can create programs showing animation, playing music, display document & download environmen simultaneously.

In multi-threading concept, the main important elcment to achieve multi-threading is termed / called as thread. A thread is a light weighted process & in java, it is specified as a ~~MAIN~~ class.

The advantage of multi-threaded program is that, it utilises system resources (like CPU, memory, input/output devices, etc) better because other thread can grab the CPU time when one thread execution is blocked. A typical program of java is executed sequentially and is said to be a single thread program. If single thread program execution is blocked while waiting for input/output operation, then no other process of program execution is proceeded.

In java, we can create multiple threads that may or may not share memory. The

switching b/w threads are automatically carried out by JVM (Java Virtual Machine)

When a standalone application is run, a thread is automatically created to execute main() function. This thread is called main thread. The program terminates when the main thread finishes execution & all other thread called child thread are spawned with/from the main thread.

Unlike C and C++, java provides facility for multi-threading. It means, java is a Object oriented programming language that provides facility to create thread & process them.

In Java, there are two ways to create a thread :-

(1) using Thread class
(2) using Runable interface

(1) Using Thread class :-

In Java, the first way to create a thread by deriving/inheriting a class from Thread class and over-riding method run() of Thread class. Thread class is specified in java.lang package. As we know, in java, a user defined thread is a user-defined class.


```
//prog to explain the example of multi thread
//using Thread class
class Xyz extends Thread
{
    public void run()
    {
        for(int i=1; i<=5; ++i)
        {
            System.out.println("I is"+i);
        }
    }
}
class Mno extends Thread
{
    public void run()
    {
        for(int j=1; j<=5; ++j)
        {
            System.out.println("J is"+j);
        }
    }
}
class Pqr extends Thread
{
    public void run()
    {
        for(int k=1; k<=5; ++k)
        {
            System.out.println("K is"+k);
        }
    }
}
```

```
class Diksha
{
    public static void main(String args[])
    {
        Xyz m=new Xyz();
        Mno n=new Mno();
        Pqr o=new Pqr();
        m.start();
        n.start();
        o.start();
    }
}
```

NOTE:- start() function is used to start/initialise thread. It means, this function switches the control to run() method.

(2) Using Runable interface:-

It is the second way to create a thread in java. In this way, a class implements/utilises Runable interface, and used the method run() of Runnable interface.

In this method/way, a thread is invoked/called using the object of Thread class, and the object of user-defined thread class is passed as argument in Thread class constructor. It means, start() function is invoked/call with object of Thread class, and the object of user defined thread is passed as argument of Thread class constructors.


```
// prog to explain the example of multi-threading
// using Runnable interface.
class Xyz implements Runnable
{
    public void run()
    {
        for(int i=1;i<=5;++i)
        {
            System.out.println("I is"+i);
        }
    }
}
class Mno implements Runnable
{
    public void run()
    {
        for(int j=1;j<=5;++j)
        {
            System.out.println("J is"+j);
        }
    }
}
class Pqr implements Runnable
{
    Public void run()
    {
        for(int k=1;k<=5;++k)
        {
            System.out.println("K is"+k);
        }
    }
}
```

```
class Diksha
{
    public static void main(String args[])
    {
        Xyz m = new Xyz();
        Mno n = new Mno();
        Pqr o = new Pqr();
        Thread t1 = new Thread(m);
        Thread t2 = new Thread(n);
        Thread t3 = new Thread(o);
        t1.start();
        t2.start();
        t3.start();
    }
}
```

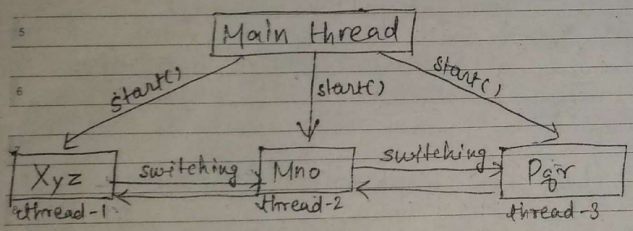


fig:- Thread Examples

Life Cycle of Thread:-

The stages/states from the creation of thread to termination of thread is called the life cycle of thread.

There are following states/stages are considered in life cycle of thread:-

(i) new born
(ii) runnable
(iii) running
(iv) Blocked
(v) Dead / terminate

(i) new :-
A process/thread is considered in new state when process creates/borns.

(ii) runnable:-
A process/thread is considered in runnable state when it is not selected by thread schedular.

(iii) running:-
A thread is considered in running state when it is selected by thread schedular. In this state, thread is executing.

(iv) blocked :-
A thread is considered in blocked state when a thread performs an input/output operation.

(V) terminate :-

A thread is considered in terminate/dead state when a thread completes its whole work.

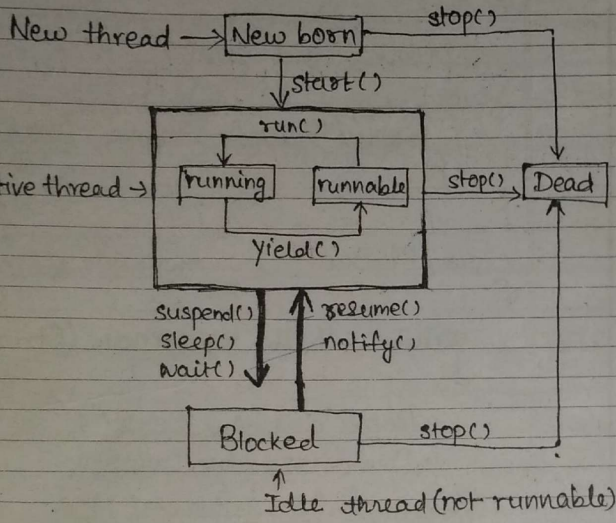


Fig:- State transition diagram of thread lifecycle

or

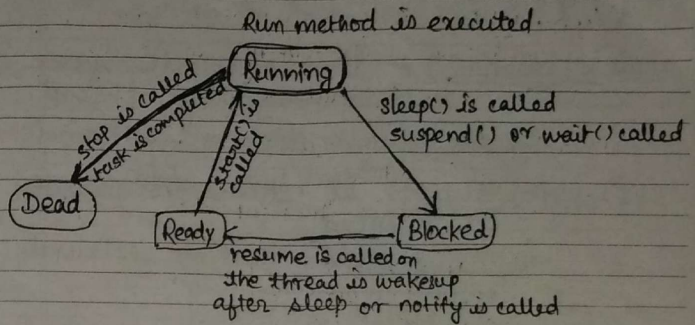


Fig:- Lifecycle of Thread

Thread Scheduling:-

- Preemptive Scheduling
- Time-slice Scheduling

Methods of Thread class:-

① start():- It is used to start execution of thread. JVM invokes/calls run method on execution of start() method.

Syntax:- public void start();

② run():- It is used to perform action for thread.

Syntax:- public void run();

③ getName():- It returns the name of thread.

Syntax:- public final String getName();

1000 milisec = 1 sec

03 DECEMBER WEDNESDAY 2008

338-028 • WEEK 49

④ setName():- It changes the name of current executing thread.

Syntax:- public final void setName(String);

⑤ getPriority():- It returns the priority of the thread.

Syntax:- public final int getPriority();

MIN_PRIORITY ⇒ 1
NORM_PRIORITY ⇒ 5
MAX_PRIORITY ⇒ 10

⑥ setPriority():- It is used to set the new priority for thread.

Syntax:- public final void setPriority(int priority);

⑦ getId():- It returns the id of the thread.

Syntax:- public int getId();

⑧ currentThread():- It returns the reference of current executing thread.

Syntax:- public static Thread currentThread();

⑨ sleep():- It causes the currently running thread to sleep for specified period of time.

Syntax:- public static void sleep(long milisecond);
public static void sleep(long milisecond, int nanosecond);

⑩ join():- It waits for a thread to die.

Syntax:- public void join();

DECEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

Syntax:- public void join();

Page:
Date: 13/3/15

```
//prog to explain the methods of thread
//class
class Demo extends Thread
{
    public void run()
    {
        Sopln("Hello Amrita, dont make noise");
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Demo d1 = new Demo();
        Demo d2 = new Demo();
        Sopln(d1.getName());
        Sopln(d2.getName());
        Sopln(d1.getId());
        Sopln(d2.getId());
        Sopln(d1.getPriority());
        Sopln(d2.getPriority());
        d1.start();
        d2.start();
        d1.setName("Nidhi didi");
        Sopln(d1.getName());
        Sopln(d2.getName());
```

```
//prog to explain the example of current
//thread
class Demo extends Thread
{
    public void run()
    {
        Sopln(Thread.currentThread().getName());
        Sopln(Thread.currentThread().getPriority());
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Demo d = new Demo();
        d.start();
    }
}
```

Note:- Both sleep() & join(), throws ~~Interrupted Exception~~ InterruptedException.


```
//prog to explain the example of sleep & join.
class Demo extends Thread
{
    public void run()
    {
        for(int i=1; i<=5; ++i)
        {
            try
            {
                sleep(2000);
            }
            catch(InterruptedException e)
            {
                System.out.println(e);
            }
            System.out.println("i is"+i);
        }
    }
}
class MyDem
{
    public static void main(String args[])
    {
        Demo d = new Demo();
        Demo d1 = new Demo();
        Demo d2 = new Demo();
        d.start();
        try
        {
            d.join();
        }
```

```
        catch(InterruptedException e)
        {
            System.out.println(e);
        }
        d1.start();
        d2.start();
    }
}
```

isAlive():- This function tests if the thread is alive or not. It returns boolean value.

Syntax:- public boolean isAlive();

```
//
//prog to explain the example of isAlive
//
class Demo extends Thread
{
    public void run()
    {
        Sopln(Thread.currentThread().getPriority());
    }
}
class MyDemo
{
    public static void main(String args[])
    {
        Demo d1 = new Demo();
        Demo d2 = new Demo();
        d1.setPriority(Thread.MIN_PRIORITY);
        d2.setPriority(Thread.MAX_PRIORITY);
        d1.start();
        d2.start();
        Sopln(d1.isAlive());
```

2008 DECEMBER 04 THURSDAY

WEEK 49 • 339-027

9/2/

APPLET:-

Applet are small special kind of java programs that run within a web document/web-page. They are accessed on an Internet server and transported over the network.

An applet is a small program which is embedded in an HTML page that can be automatically downloaded from web server and run in any java enabled browser or not java browser of java. Applet are automatically installed & run as part of HTML document.

The current use of applet includes following:-

(1) Animated graphics
(2) Video games
(3) On-line tests
(4) Databases, Reports, etc.

Difference between Java Applet & Java Appli

Java Applet	Java Application
(A) Java Applets are used for creating dynamic & interactive web application.	(1) Java application are used to write commercial application.

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S JANUARY 2009

Java Applet	Java Application
② Applets are embedded within web page/HTML page & run under the control of browser.	② Java application runs stand-alone with support of Java virtual machine (JVM).
③ Applet application has no main() method.	③ Java applications support main() method.
④ Java Applet is only compiled. It is not executed by interpreter, because it has no main() method. It is only executed by browser.	④ Since, java application has main method, so it is both compiled as well as interpreted.
⑤ Java Applet does not use the library of other languages such as C and C++.	⑤ Java application has facility to use the library of other languages.

TYPES OF APPLET:-

NOTES There are basically two types of applet:-

1. Local Applet
2. Remote Applet

① LOCAL APPLET:-

The applet which resides on local computer & also executed on same computer is called local applet.

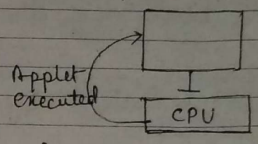
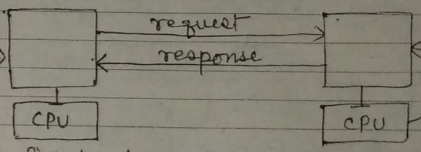


fig:- Local applet executed on local computer

② REMOTE APPLET:-

The applet which resides on remote computer such as web-server and executed on that computer is called remote applet.



output is displayed ⇒ on brower of client computer, because browser acts as a client & interacted with user.

fig:- Local computer

acts as a client

fig:- web Server or remote computer

acts as a web server

NOTES

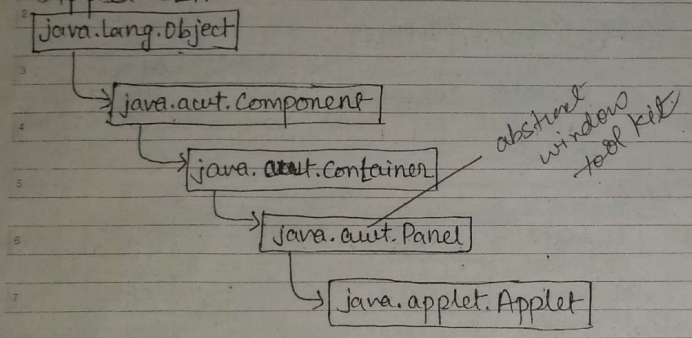
java.awt → Abstract Windows Toolkit

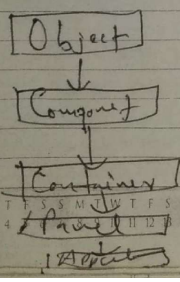
08 DECEMBER 2008
MONDAY 343-023 • WEEK 50

Creating user-defined applet:-

In Java, a user-defined applet is a class which is derived from a built-in class Applet and overrides the method of Applet class.

The Applet class hierarchy in java is as follows:-





10/2

2008 DECEMBER TUESDAY 09
WEEK 50 • 344-022

Applet LifeCycle:-

There are following stages in lifecycle of applet:-

1. Initialisation state
2. Running state
3. Blocked state
4. Display state
5. Dead state

(1) Initialisation state:-

In this state, the applet is first loaded into the memory. It means initialisation state comes only once in applet lifecycle.

In this state, the public void init() method is automatically triggered/invoked call.

Syntax:-
```
public void init()
{
  ——— ———
}
```

(2) Running state:-

After initialisation, the applet enters into running state. This state shows that an applet is executing its work.

In this state, the public void start() method is automatically invoked.

10 DECEMBER WEDNESDAY 2008
345-021 • WEEK 50

Syntax:- public void start()
{
}

(3) Blocked State / Idle state:-

In this state, applet stops its own work temporarily.

In this state, the public void stop() method is automatically invoked.

Syntax:- public void stop()
{
}

(4) Display state:-

It is a part of running state. This state shows that the applet is displaying something on the web page.

In this state, the public void paint() method is automatically invoked.

Syntax:- public void paint(Graphics g)
{
}

NOTES

DECEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 DECEMBER THURSDAY 12/2/15 11
WEEK 50 • 346-020

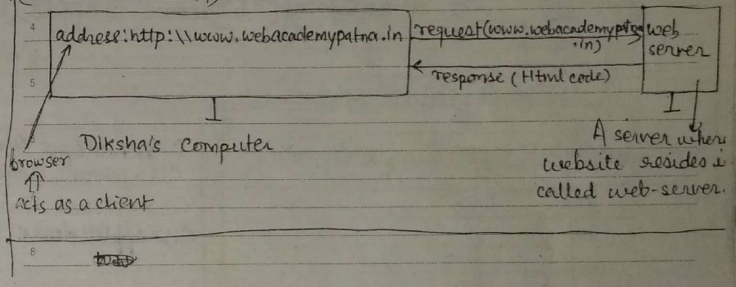
(5) Dead state:-

It happened only once in a applet lifecycle. It has come when either the applet has completed its work, or we exit from the browser or closing the web page.

In this state, the public void destroy method is automatically invoked.

Syntax:- public void destroy()
{
}

(At the back of copy)



NOTES

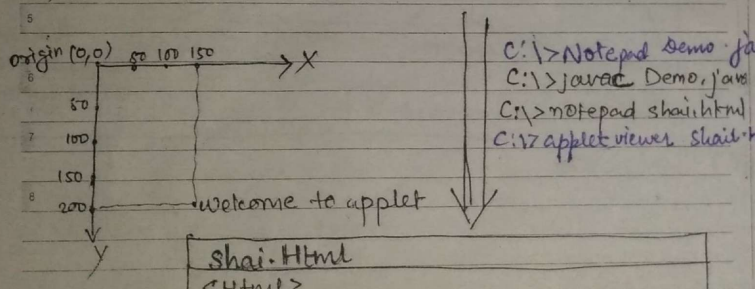
M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S JANUARY 2009

12 DECEMBER FRIDAY 2008 347-019 • WEEK 50

Graphics
- drawString()
- drawLine()
- drawRect()

Q. WAP to explain the example of applet to display a message, & also write Html code to execute applet.

Prog:-
```
//Example of applet
import java.applet.*;
import java.awt.*;
public class Demo extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Welcome to applet",150,200);
    }
}
```
(x coordinate, y coordinate)




```
C:\>Notepad Demo.java
C:\>javac Demo.java
C:\>notepad shai.html
C:\>appletviewer shai.html
```

shai.Html
```
<Html>
  <Head>
  </Head>
  <Body>
    <Applet code="Demo" width="400" Ht="500">
    </Applet>
  </Body>
</Html>
```

NOTES

DECEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 DECEMBER SATURDAY 13 WEEK 50 • 348-018

Note:-
(1) Applet tag of Html is used to hold as well as to execute applet of java.

Syntax for applet tag is as follows:-
```
<applet code="  " width="  " Height="  ">
</applet>
```
tag; property; used to specify applet code file name

(2) drawString() is a non-static function/method of Graphics class which is used to display text/string on applet screen window.

Syntax:-
```
graphics-object.drawString(msg, x, y);
```
(x, y: coordinates)

Ex:- g.drawString("Hello",500,300);

(3) Appletviewer command/tool of Java development Kit (JDK) is used to execute Html file which holds applet code.

NOTES

SUNDAY 14

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

13/2/15

Q. WAP to explain the example of different state of applet.

```
//example of different state of applet
import java.applet.*;
import java.awt.*;
public class Mast extends Applet
{
    String s;
    public void start()
    {
        s+="valentine";
    }
    public void init()
    {
        s="Happy";
    }
    public void paint(Graphics g)
    {
        s+="Day";
        g.drawString(s,150,200);
    }
}
```

↓

```
<Html>
    <Body>
        <applet code="Mast" width="300" Height="400">
        </applet>
    </Body>
</Html>
```


Q. WAP to print the sum of two nos. using applet lifecycle.

```
//prog to print the sum of two nos.
import java.applet.*;
import java.awt.*;
public class Mast extends Applet
{
    int x,y,r;
    public void init()
    {
        x=15;
        y=30;
    }
    public void start()
    {
        r=x+y;
    }
    public void paint(Graphics g)
    {
        g.drawString(String.valueOf(r),150,200);
    }
}
```

converting r to string

Notes

17 DECEMBER 2008
WEDNESDAY
352-014 • WEEK 51

Passing Parameter to Java Applet from Html page

Syntax:- <param Name=" " value=" ">

- used to send parameter
- specify parameter name
- specify value of parameter

Q. WAP to print the product of two nos. using applet. (No. must be sent from Html file)

Prog:-
```
import java.applet.*;
import java.awt.*;
public class Mast extends Applet
{
    int x, y, p;
    public void init()
    {
        x=Integer.parseInt(getParameter("num1"));
        y=Integer.parseInt(getParameter("num2"));
    }
    public void start()
    {
        p = x*y;
    }
    public void paint(Graphics g)
    {
        g.drawString(String.valueOf(p),150,200);
    }
}
```

NOTES

DECEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S


```
<Html>
    <Head>
    </Head>
    <Body>
        <Applet code="Mast" width="300" height="
            <param name="num1" value="70">
            <param name="num2" value="5">
        </applet>
    </Body>
</Html>
```

Note:-
Java applet uses getParameter() function method to receive parameter from Html file.

Syntax:- getParameter("Parameter-name");

Ex:- getParameter("num1");

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S JANUARY 2009

19 DECEMBER FRIDAY 16/12/15 2008

Q. Prog to create an applet to display details of student. The information must be sent from Html file.

```
// prog to create an applet to display details of student
import java.applet.*;
import java.awt.*;
public class Student extends Applet
{
    String nm;
    int roll;
    float fee;
    public void paint(Graphics g)
    {
        nm = getParameter("name");
        roll = Integer.parseInt(getParameter("rl"));
        fee = Float.parseFloat(getParameter("fish"));
        g.drawString(nm, 70, 100);
        g.drawString(String.valueOf(roll), 70, 110);
        g.drawString(String.valueOf(fee), 70, 120);
    }
}
```

<Html>
<Body>
<Applet code="Student" width="600" height="500">
<param name="name" value="Tushar">
<param name="rl" value="87">
<param name="fish" value="12000.50">
</Applet>
</Body>
</Html>

NOTES

DECEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S

EXCEPTION HANDLING :-

Exception handling is a mechanism to identify & detect an exception during execution of program & resolved that exception. An exception is an event which occurs during execution of program & disturb the normal flow of program.

An exception is a condition that is caused by a runtime error. In program, a java exception is an object that describes an exceptional condition that is found in a piece of code. In java, exception can be generated by java runtime system or by coding manually.

There are five keywords used in exception handling of java. These keywords are as follows:-

(1) try
(2) catch
(3) throw
(4) throws
(5) finally

Note:- All built-in exception of java are derived from Throwable class which is further inherited from object class.

The Throwable class has two derive class/ subclass. These sub-class are as follows:-

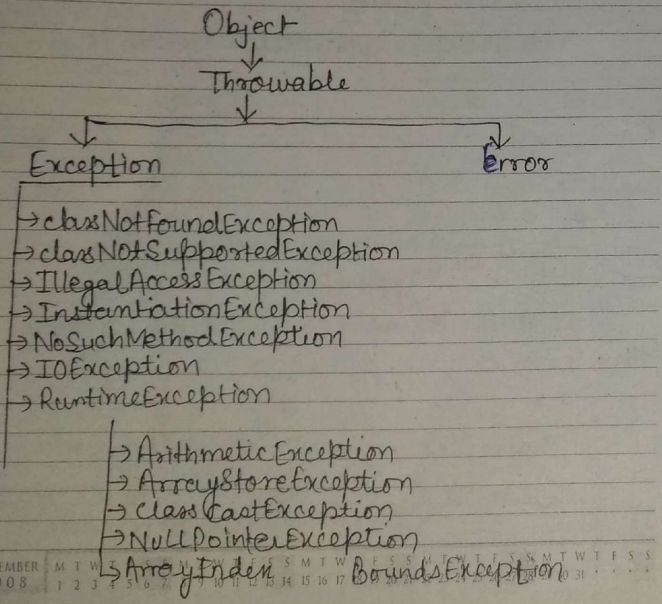
(1) Error (2) Exception

(1) Error:- This class is used by java runtime system to indicate errors.

Ex:- Stack overflow is an example of exception of Error class.

(2) Exception:- This class is used to specify exceptional conditions that user programs should catch.

Ex:- divide by zero, array index out of bounds etc. are example of exceptions of Exception class.




```
try
{
}
catch(Exception e)
{
}
finally
{
}
```

Q. WAP to explain the example of exception handling
OR
WAP to explain the example of arithmetic exception in java.

Prog:-
```
// Example of Arithmetic exception
class Demo
{
    public static void main(String args[])
    {
        int x=15, y=0;
        int r;
        try
        {
            r = x/y;
            System.out.println(r);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Exception caught"+e.getMessage());
        }
    }
}
```

Q. WAP to explain the example of finally keyword.

Prog:
```
//Example of finally keyword
class Demo
{
    public static void main(String args[])
    {
        int x=15, y=0;
        int r;
        try
        {
            r=x/y;
            Sopln(r);
        }
        catch(ArithmeticException e)
        {
            Sopln("Exception caught"+e.getmessage);
        }
        finally e
        {
            Sopln("Hi---See U");
        }
    }
}
```

Q. WAP to explain the example of ArrayIndexOutOfBoundsException.

```
//Example of ArrayIndexOutOfBoundsException
class Demo
{
    public static void main(String args[])
    {
        int x[]={12,15,20,5,8};
        int r;
        try
        {
            r=x[12];
            Sopln(r);
        }
        Catch(ArrayIndexOutOfBoundsException e)
        {
            Sopln("Exception caught"+e.getmessag
        }
        finally
        {
            Sopln("Hi---See u");
        }
    }
}
```


	C	C++
Source file	ram.c ← source code	ram.cpp ← source c
	↓↓ Alt+F9 (compile)	↓↓ Alt+F9 (comp
Objectfile	ram.obj ← object code	ram.obj ← object cod

java
↓
ram.java ← source code
↓↓ javac ram.java
ram.class ← Byte code

NOTE: source file
class file

29 DECEMBER 2008
MONDAY 364-002 • WEEK 53

```
//Example of exception handling
class Demo
{
    public static void main(String args[])
    {
        String s="ramesh";
        Sopln(s);
        try
        {
            s=null;
            Sopln(s.length());
        }
        catch(NullPointerException e)
        {
            Sopln(e);
        }
    }
}

//example of exception handling
class Demo
{
    public static void main(String args[])
    {
        String s="123";
        int x;
        x=Integer.parseInt(s);
        Sopln(x+20);
        try
        {
            s="abc";
```

NOTES

DECEMBER 2008 M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S

2008 DECEMBER 30
WEEK 53 • 365-001 TUESDAY

```
            x=Integer.parseInt(s);
            System.out.println(x+20);
        }
        catch(NumberFormatException e)
        {
            System.out.println(e);
        }
    }
}

//Example of multiple catch Statements
import java.util.*;
class Demo
{
    public static void main(String args[])
    {
        Scanner ss=new Scanner(System.in);
        String s;
        try
        {
            System.out.println("Enter a string");
            s=ss.nextLine();
            int r=s.length();
            System.out.println(r);
            int x=Integer.parseInt(s);
            System.out.println(x);
        }
        catch(NumberFormatException e)
        {
            System.out.println(e);
        }
        catch(NullPointerException e)
        {
            System.out.println(e);
```

NOTES

M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S JANUARY 2009

Topics of Java

- Character set of Java
- Token
- Structure of java program
- Compile & execute java program
- Operator Precedance
- Statement
- Loop
- Array
- New Version of for loop
- Jagged Array
- Labelled Break & Labelled continue
- String
- functions of String class
- String Buffer
- Functions of String buffer
- Character input using Scanner class
- Receiving input through command line
- " " using DataInputStream
- " " using console class
- Executing java program on Netbeans editor
- Function ← (Non-Static)
  (Static)
- Class ← Data Member
  Member function
- Note - access specifier

- Constructor
- Constructor Overloading
- this keyword
- Inheritance
- Method overriding
- constructor in inheritance
- super keyword
- final keyword
- Object/Instance
- DMD
- Abstract class
- Interface
- Process
- Multi-threading
- Life cycle of thread
- Methods of thread
- APPLET
- Life cycle of Applet
- Exception Handling
- Graphics

Website
⇓
Webpage (A single page of website)

Static
⇓
Made with Html
Hypertext Markup language
(A language which is understandable by browser).

Dynamic/Server
⇓
made with
→ ASP (Active Server Page)
→ JSP (Java Server Page)
→ PHP (Preprocessor Hypertext)
or
(HyperText preprocessor)

~~Html :- A page having~~

Html :- A page having extension .html/.htm is called html file/ html script/ html program. It is made up of tags. Tags are specified within angle brackets (< >).

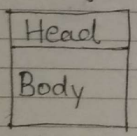


fig: Html page Architecture

opening tag → <Html>
<Head>
</Head>
<Body> → opening tag

</Body> → closing tag

closing tag → </Html>

11/2/15

Note :-

① Any Html program starts with opening tag <Html> and ends with closing tag </Html>.

② A closing tag is preceeded with forward slash (/).

③ An Html page is divided into two parts namely Head and Body and they are represented with <Head> and <body> tags respectively.

<title></title> :- used to display title of web page

Ex :- <title> welcome to Html... </title>

<h1></h1> to <h6></h6> ⇒ used to specify headings

 :- To specify a paragraph.

 :- for line break
<hr/> :- for horizontal line

```
//Program

<Html>
    <head>
        <title>welcome to html...</title>
    </head>
    <body>
    <h1>welcome</h1>
    <h2>welcome</h2>
    <h3>welcome</h3>
    <h4>welcome</h4>
    <h5>welcome</h5>
    <h6>welcome</h6>
    This is a pen.<br/><hr/>This is a book.
    <p> ---------------------- </p>
    <p> ---------------------- </p>
    <p> ---------------------- </p>
    </body>
</Html>
```

Note:- The property regarding a tag is specified in opening tag.

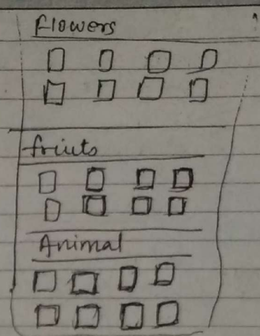
Ex:- <body bgcolor="Red"> (property name, value)
</Body>

Ex:- <body background="xyz.jpg">

<img> :- To display an image on web page.

Ex:- <img src="a.jpg" width="300" Height="400"> (property name; src = Source of image)
</img>

Project



www.wips.co.in

Domain (The name of website is considered as do...
Domain must be unique.

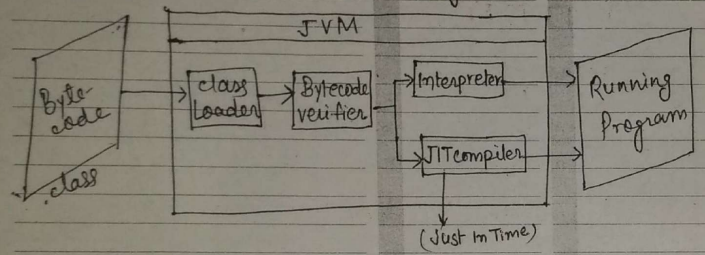
IIS → Internet Information Server.

Q① WAP to create a class circle to print the area & circumference of circle.

② WAP to create a class reverse to print the reverse of a given function.

③ WAP to create a class Great to print the greatest no. among three nos.

Syntax:- public void join();
public void join(long milisecond);



Q. Explain the feature of effective DBMS with pros and cons.

super keyword
with the help of named instance

frame :- It is a top level window that is not contained inside another window. frame class of AWT & JFrame class of Swing is used to create a frame.

```
import javax.swing.*;
class SimpleFrame extends JFrame
{ public static final int DEFAULT_WIDTH = 300;
  public static final int DEFAULT_HEIGHT = 200;
  public SimpleFrame()
  {
    setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
  }
}
class Demo
{
  main(--)
  {
    SimpleFrame fr = new SimpleFrame();
    fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fr.setVisible(true);
  }
}
```

boolean isVisible() → tells or sets the visible property.
void setVisible(boolean)
void setSize(width, height) :- resizes the component to specified width and height.
setLocation(x, y) → moves the component to a new location.
setBounds(x, y, width, Height) → moves and resizes the component.
public String getTitle() → getter fun.
Public void setTitle(String) → setter function

Note:- if we don't explicitly size a frame, then default size is 0 by 0 pixels.

To find out screen size

```
Toolkit kit = Toolkit.getDefaultToolkit();
Dimension scrsiz = kit.getScreenSize();
int scrwidth = scrsiz.width;
int scrheight = scrsiz.width;
```

we should use 50% of these values for frame size.

```
setSize(scrwidth/2, scrheight/2);
setLocationByPlatform(true);
Image img = kit.getImage("icon.gif");
setIconImage(img);
```

17/2/15

January 2015

06

WK 02 (006-359) • Tuesday

Graphics in Java (2-d):-

In Java, a built-in class called Graphics is used to create graphical object on applet screen.

The method/function of Graphics class which is used for designing 2-d Graphics is as follows:-

Graphics class:-

- drawString()
- drawLine()
- drawRect()
- fillRect()
- drawRoundRect()
- fillRoundRect()
- draw3DRect()
- fill3DRect()
- drawOval()
- fillOval()
- drawArc()
- fillArc()
- drawPolygon()
- fillPolygon()
- drawPolyline()

JANUARY 2015

WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	

FEB MAR APR MAY JUN

07 January 2015
Wednesday • WK 02 (007-358)

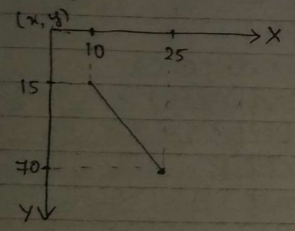
① drawString():-
This function is used to draw string on applet screen.

Syntax:- graphicsobject.drawString(msg,x,y);

② drawLine():-
This function is used to draw a line on the applet screen.

Syntax:- graphicsobject.drawLine(x_1, y_1, x_2, y_2);
$x_1 y_1$ ——— $x_2 y_2$ (Integer)

Ex:- g.drawLine(10,15, 25, 70);

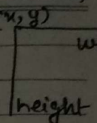


③ drawRect():-
This function is used to draw a rectangle on applet screen.

Syntax:- graphicsobject.drawRect(x, y, width, height);

08 January 2015
WK 02 (008-357) • Thursday

Ex:- g.drawRect(15, 70, 60, 30);



④ fillRect():-
This function is used to draw a filled Rectangle on applet screen.

Syntax:- graphics.fillRect(x,y,width,height);

Ex:- g.fillRect(20,70,60,30);

⑤ drawRoundRect():-
This function is used to draw a rectangle with rounded corner on applet screen.

Syntax:- graphicsobject.drawRoundRect(x,y, width, height, arc width, arc height);

⑥ fillRoundRect():-
This function is used to draw a filled rectangle with rounded corner on applet screen.

Syntax:- graphicsobject.fillRoundRect(x, y, width, height, arc width, arc height);

JANUARY 2015

WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	

FEB MAR APR MAY

09 January 2015
Friday • WK 02 (009-356)

```
//Example of Graphics functions
import java.applet.*;
import java.awt.*;
public class Demo extends Applet
{
    public void paint(Graphics g)
    {
        g.setColor(Color.red);
        g.drawRect(25,10,50,75);
        g.drawRoundRect(100,10,50,75,30,150);
        g.drawLine(20,150,400,40);
        g.fillRect(25,100,50,50);
    }
}
```

(7) setColor():-
It is used to set colour.

Syntax:- graphicsobject.setColor(color-name);

Ex:- g.setColor(Color.red);
classname

(8) draw3DRect():-
This function is used to draw

10 January 2015
WK 02 (010-355) • Saturday

Syntax:- graphicsobj.draw3DRect(x,y,width,height, raised);
Boolean

Ex:- g.draw3DRect(175,10,50,75,false);
or
g.draw3DRect(175,10,50,75,True);

Q. WAP to explain the example of setBackground() & setForeground() function.

```
//Example of setBackground & foreground
import java.applet.*;
import java.awt.*;
public class App1 extends Applet
{
    public void init()
    {
        setBackground(Color.red);
        Color c = new Color(127,255,212);
        setForeground(c);
    }
    public void paint(Graphics g)
    {
        g.drawString("Welcome Diksha",70,50);
        g.setColor(Color.green);
        g.fillArc(170,110,50,50,30,300);
    }
}
```

(9) setBackground():-
This function sets background color of applet screen.

Syntax:- setBackground(Color-name/obj);

(10) Color():-
It is a class to specify color.

Syntax:- color obj = new Color(R, G, B);
R, G, B → 0 to 255

(11) setForeground():-
This function sets foreground colour of applet screen.

Syntax:- setforeground(Colorname/obj);

(12) drawOval();-
This function is used to draw oval shaped such as circle, ellipse, ... etc on applet screen.

Syntax:- graphicsobj.drawOval(x,y,width,height);
Ex:- g.drawOval(10,25,50,60);

(13) fillOval();-
This function is used to draw a filled oval shapes such as circle, ellipse, ... etc on applet screen

Syntax:- graphicsobj.fillOval(x,y,width,height);

Ex:- g.fillOval(10,25,50,60);

(14) drawArc():-
This function is used to draw an arc on applet screen.

Syntax:- g.drawArc(x,y,width,height,startAngle, endangle/Arcangle);

Ex:- g.drawArc(10110,50,50,30,300);

(15) fillArc():-
This function is used to draw a filled arc on applet screen.

Syntax:- g.fillArc(x,y,width,height,startangle, endangle);

Ex:- g.fillArc(170,110,50,50,30,300);

JANUARY 2015

WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	

FEB MAR APR MAY JUN

13 January 2015
Tuesday • WK 03 (013-352)

(16) drawPolygon( ):-
This function is used to draw a polygon on applet screen.

Syntax:- g.drawPolygon(int x[], int y[], int no_points);

(17) fillPolygon( ):-
This function is used to draw a filled polygon on a applet screen.

Syntax:- g.fillPolygon(int x[], int y[], int no_points);

(18) drawPolyline( ):-
This function is similar to polygon, except last point is not connected to first.

Syntax:- g.drawPolyline(int

(19) setFont( ):-
This function is used to set font for text. We have to specify the object of Font class as argument.

January 2015 14
WK 03 (014-351) • Wednesday

Ex:- Font m;
m = new Font("Times new Roman", Font.BOLD, 48);
g.setFont(m);

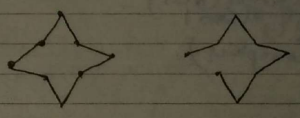
Note:-
The Font class has following format:-

Font(fontname, style, size);
constructor String int int

```
//Example of Graphics class function
import java.applet.*;
import java.awt.*;
public class Demo extends Applet
{
    public void paint(Graphics g)
    {
        int x[]={10,30,40,50,70,50,40,30};
        int y[]={40,30,10,30,40,50,70,50};
        g.drawPolygon(x,y,x.length);
        g.setColor(Color.yellow);
        g.drawPolyline(x,y,x.length);
    }
}
```

JANUARY 2015

WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	



15 January 2015
Thursday • WK 03 (015-350)

```
import java.applet.*;
import java.awt.*;
public class Demo extends Applet
{
    public void paint(Graphics g)
    {
        Font m = new Font("arial", Font.ITALIC, 40);
        g.setFont(m);
        g.drawString("MANMOHAN SINGH", 100, 150);
    }
}
```

Component class methods

- setVisible(boolean) :- shows the component, if parameter is true.
- setSize(w, h)
- setSize(Dimension)
- setLocation(x, y)
- setFont(Font)
- setEnabled(boolean)
- setBounds(Rectangle)
- setBackground(Color)
- setForeground(Color)
- repaint()
- paint(Graphics)
- getX()
- getY()
- getLocation()
- getSize()

January 2015
16 Friday
WK 03 (016-349)

Exploring java.awt package :-

- java.lang.Object
  - Component {abstract}
    - Button → ~~ActionListener~~
    - Checkbox
    - Label
    - Choice
    - List
    - TextComponent {abstract}
      - TextField
      - TextArea
    - Container {abstract}
      - Panel → java.applet.Applet
      - Window
        - Frame
        - Dialog
          - FileDialog
  - MenuComponent {abstract}
    - MenuBar
    - MenuItem
      - CheckboxMenuItem
      - Menu
        - PopupMenu

JANUARY 2015

WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	

FEB MAR APR MAY JU

17 January 2015
Saturday • WK 03 (017-348)

Label :-

It is used to display a message.

Syntax :- Label();
Label(String);
Label(String, align);

Ex :- Label m = new Label();
Label n = new Label("Hello");
Label o = new Label("Hi", Label.RIGHT);

Note :-
We have to add each component to a window/applet window using add() function/method.

Syntax :- add(component object);

Ex :- add(m);

setText(String)
String getText()
setAlignment(int)
int getAlignment()
static int CENTER
" " LEFT
" " RIGHT → field

18 January 2015
WK 03 (018-347) • Sunday

Q. WAP to explain the example of Label component.

```
Prog :- //Example of Label Component
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        Label m = new Label("Hi");
        Label n = new Label("Hello");
        Label o = new Label("Bye");
        add(m);
        add(n);
        add(o);
    }
}
```

Button() :- setLabel() / String getLabel()

To display a push button.

Syntax :- Button();
Button(String);

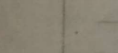
Ex :-
```
Button m = new Button("OK");
add(m);
```

OK

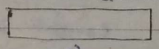
19
Monday • WK 04 (019-346)
January
2015

Q. WAP to explain the example of button component.

```
//Example of button
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        Button b=new Button();
        Button b1=new Button("OK");
        add(b);
        add(b1);
    }
}
```



TextField():-
To display single line textbox.



Syntax:-
TextField();
TextField(int column); (size)
TextField(String);
TextField(String, column);

Constructor

set(Columns(int)
setText(String)
setEditable(boolean)
getSelectedText()
select(start-index, end-index)
fill end-1

20
WK 04 (020-345) • Tuesday
January
2015

Ex:-

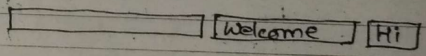
Q. WAP to explain the example of TextField().

Prog:-
```
//Example of TextField
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        TextField t1=new TextField();
        TextField t2=new TextField(30);
        TextField t3=new TextField("Welcome");
        TextField t4=new TextField("Hi",5);
        add(t1);
        add(t2);
        add(t3);
        add(t4);
    }
}
```

JANUARY 2015

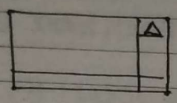
WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	

Output:-



Wednesday • WK 04 (021-344) January 2015

TextArea( ):- To specify multi-line textbox.



→ append(String) → inserts at end
→ insert(string, index) → inserts at specified index
→ replaceRange(string, startindex, endindex)

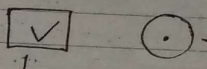
Syntax:-
```
TextArea( );
TextArea(String);
TextArea(int row, int col);
TextArea(String, row, col);
TextArea(String, row, col, scroll-bar);
```
↓
TextArea.SCROLLBARS_BOTH
TextArea.SCROLLBARS_VERTICAL_ONLY
TextArea.SCROLLBARS_HORIZONTAL_ONLY
TextArea.SCROLLBARS_NONE

```
// Example of TextArea
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        TextArea t1 = new TextArea( );
        TextArea t2 = new TextArea("welcome");
        TextArea t3 = new TextArea(5, 20);
```

January 2015 22 WK 04 (022-343) • Thursday

```
        TextArea t4 = new TextArea("Hi", 7, 15);
        add(t1);
        add(t2);
        add(t3);
        add(t4);
    }
```

Checkbox :- To display checkbox as well as radio-button (exclusive checkbox).



Non-exclusive Check-box → Ability for multiple selection.

radio-button (Exclusive checkbox) → for single selection.

CheckBox has two states :- True and False
↓ checked ↓ unchecked

Syntax:-
```
Checkbox( );
Checkbox(String);
Checkbox(String, State);
```
↓ Boolean

23 Friday • WK 04 (024-342) January 2015

boolean getState()
void setState(boolean)
String getLabel()
setLabel(String)

null, if not part of any

(for radio Button) → Checkbox(string, state, CheckboxGroup);

can be in either way

//Example of CheckBox

```
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        Checkbox c1=new checkbox();
        Checkbox c2=new Checkbox("Bca");
        Checkbox c3=new Checkbox("Mca",true);
        Checkbox c4=new Checkbox("B.tech",false);
        add(c1);
        add(c2);
        add(c3);
        add(c4);
    }
}
```

☐ ☐ Bca ☑ Mca ☐ Btech

January 2015 24 WK 04 (024-341) • Saturday

=> CheckboxGroup()
Checkbox getSelectedCheckbox(); return the selected checkbox in group
void setSelectedCheckbox(Checkbox cb)

//Example of CheckBox Group

```
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        CheckboxGroup m=new CheckboxGroup();
        Checkbox c1=new Checkbox("Bca",false,m);
        Checkbox c2=new Checkbox("Mca",true,m);
        Checkbox c3=new Checkbox("Btech",false,m);
        add(c1);
        add(c2);
        add(c3);
    }
}
```

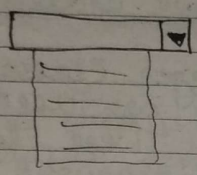
○ Bca ⊙ Mca ○ Btech

JANUARY 2015

WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	

FEB MAR APR MAY

Choice :- To design drop-down list box (combo-box in VB)



- add(string)
- addItem(string) > same
- String getItem(int) ; index no.
- int getItemCount() → returning no. of items
- int getSelectedIndex()
- insert(string, int) → index
- remove(int)
- remove(String)
- select(int)
- select(String)

Ex:-

```
Choice m = new Choice();
m.add("Madhav");
m.add("Aayush");
m.add("Ravi");
add(m);
```

//Example of Choice

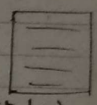
```
import java.applet.*;
import java.awt.*;
public class Jayanti extends Applet
{
    public void init()
    {
        Choice m=new Choice();
        m.add("Diksha");
        m.add("Amrita");
        m.add("Rahul");
        m.add("Jayanti");
        m.add("Ravi");
        add(m);
    }
}
```


List :- To display/design list-box.

Syntax:-

```
List()
List(int rows)        → no. of rows
List(int rows, boolean multiple)
```

↓ for multiple selection of items (true for multi-select, false for single select)



item

functions:-

```
add(String item);          → add at the end of list.
add(String item, int index-no.);
```

-1 to add at end.

//Example of List

```
import java.applet.*;
import java.awt.*;
public class Ekta extends Applet
{
    public void init()
    {
        List m=new List();
        m.add("Diksha");
        m.add("Amrita");
        m.add("Shweta");
        m.add("Jayanti");
        add(m);
    }
}
```

JANUARY

WK	M	T	W	T	F	S	S
01				1	2	3	4
02	5	6	7	8	9	10	11
03	12	13	14	15	16	17	18
04	19	20	21	22	23	24	25
05	26	27	28	29	30	31	

27
Tuesday • WK 05 (027-338)
January 2015

```
//Example of List
import java.applet.*;
import java.awt.*;
public class Ravi extends Applet
{
  public void init()
  {
    List m=new List(3,true);
    m.add("Diksha");
    m.add("Karuna");
    m.add("Akriti");
    m.add("Prem");
    add(m);
  }
}
```

20/2/15
January 2015
28
WK 05 (028-337) • Wednesday

Layout Manager Classes :-

Layout Manager Classes are pre-defined classes which are used to create the user-interface. These classes provides a set of rules through which we can design the interface.

There are following Layout Manager classes :-

1. FlowLayout
2. GridLayout
3. BorderLayout
4. CardLayout
5. GridBagLayout

Note:- setLayout() method is used to set a particular Layout Manager class.
Syntax:- setLayout(obj);
obj → Layout Manager class

1. FlowLayout :-

It is the default Layout for applet and panel. This Layout adds object to a container in rows, from left to right where rows are constructed from top to bottom. When the first row fills the container, the components are wrapped to the next row automatically. With FlowLayout

only the alignment and the horizontal & vertical spacing between components can be changed.

Note:-
The default alignment is centre & default spacing is five pixel, both horizontally & vertically.

Syntax:- FlowLayout()
FlowLayout(int align);
↓↓
FlowLayout.LEFT
FlowLayout.RIGHT
FlowLayout.CENTRE
FlowLayout(align, int horizontal-gap, int vertical-gap);

Q. WAP to explain the example of FlowLayout Manager class.

Prog:-
```
//Example of FlowLayout Manager
//class
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        FlowLayout m=new FlowLayout(FlowLayout.LEFT,15,25);
        setLayout(m);
        Label x=new Label("Hi");
        Label y=new Label("Hello");
        TextField z=new TextField();
        add(x);
        add(y);
        add(z);
    }
}
```

Or

```
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
    public void init()
    {
        setLayout(new FlowLayout(FlowLayout.LEFT,15,25));
        add(new Label("Hi",Label.RIGHT));
        add(new Label("Hello",Label.LEFT));
        add(new TextField(20));
    }
}
```

31

Saturday • WK 05 (031-334)

January 2015

(2) GridLayout :-

It is a pre-defined class which is used to create the user-interface. It divides the container into several rows and columns logically and then the controls are placed on the container row by row.

Syntax:- GridLayout();
GridLayout(row, col);
GridLayout(row, col, hgap, vgap);
(hgap: horizontal, vgap: vertical)

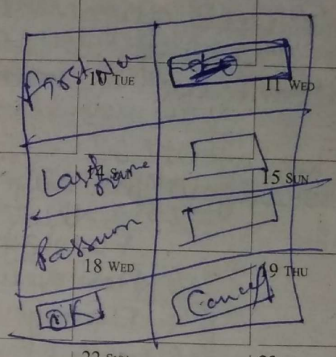
```
//Example of GridLayout
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
  public void init()
  {
    setLayout(new GridLayout(4,2));
    add(new Label("First Name"));
    add(new TextField(20));
    add(new Label("Last Name"));
    add(new TextField(20));
    add(new Label("Password"));
    add(new TextField(20));
```

ACTION PLAN

FEBRUARY 2015

Schedule your appointments, events and important date

01 SUN	02 MON	03 TUE	04 WED
05 THU	06 FRI	07 SAT	08 SUN
09 MON	10 TUE	11 WED	12 THU
13 FRI	14 SAT	15 SUN	16 MON
17 TUE	18 WED	19 THU	20 FRI
21 SAT	22 SUN	23 MON	24 TUE
25 WED	26 THU	27 FRI	28 SAT



Time...

We cannot do everything at once, but we can do something at once.

```
        add(new Button("OK"));
        add(new Button("cancel"));
    }
}
```

(3) **BorderLayout:-**

It is default layout of Frame. It divides the container into five regions namely NORTH, SOUTH, EAST, WEST, CENTRE.

Syntax:- BorderLayout();
BorderLayout(Horizontalgap, Vertical-gap);

Note:-
In BordLayout, add() method is used to add component at appropriate direction.

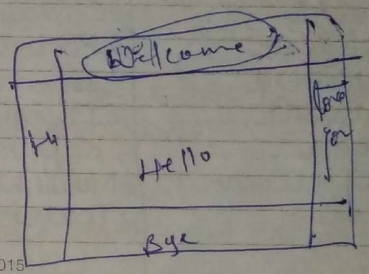
Syntax:- add(component-obj, direction);

//Example of BorderLayout

```
import java.applet.*;
import java.awt.*;
public class Diksha extends Applet
{
```



```
public void init()
{
    Label x = new Label("Hi");
    Label y = new Label("Hello");
    Label z = new Label("Bye");
    Label m = new Label("Welcome");
    Label n = new Label("Love You");
    BorderLayout br = new BorderLayout();
    setLayout(br);
    add(m, BorderLayout.NORTH);
    add(x, BorderLayout.WEST);
    add(z, BorderLayout.SOUTH);
    add(y, BorderLayout.CENTRE);
    add(n, BorderLayout.EAST);
}
}
```



24/2/15

07

Saturday • WK 06 (038-327)

February 2015

EVENT HANDLING:-

⇓ based on

Deligation model

Source (built-in classes) | Listener (built-in interfaces)

- MouseEvent ⇒ MouseListener
- MouseMotionEvent ⇒ MouseMotionListener
- ActionEvent ⇒ ActionListener
- AdjustmentEvent ⇒ AdjustmentListener
- ContainerEvent ⇒ ContainerListener
- FocusEvent ⇒ FocusListener
- InputEvent ⇒ InputListener
- ItemEvent ⇒ ItemListener
- KeyEvent ⇒ KeyListener
- MouseWheelEvent ⇒ MouseWheelListener
- TextEvent ⇒ TextListener

Button → ActionPerformed(ActionEvent e)
checkBox → itemStateChanged(ItemEvent e)

25/2/15

08

February 2015

WK 06 (039-326) • Sunday

Syntax:-

addTypeListener(TypeListener el)

```
//prog to explain the example of
//event handling
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Diksha extends Applet
                    implements KeyListener
{
    public void init()
    {
        addKeyListener(this);
    }
    public void keyPressed(KeyEvent e)
    {
        showStatus("Hi");
    }
    public void keyReleased(KeyEvent e)
    {
        showStatus("Hello");
    }
    public void keyTyped(KeyEvent e)
    {
        showStatus("Bye");
    }
}
```

09 February 2015 10

Monday • WK 07 (040-325) WK 07 (041-324) • Tuesday

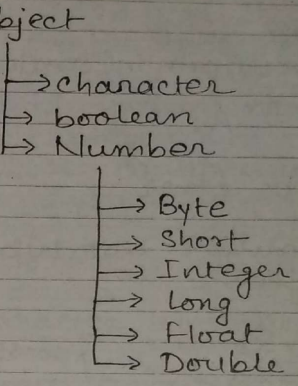
Wrapper Class:-

Wrapper class is the class that converts a built-in datatype to its equivalent object type. Wrapper class facility of java makes java as pure object oriented programming language. With the help of wrapper class, built-in types/datatypes wraps with object type.

There are following wrapper classes of built-in datatypes:-

Datatype	Wrapper class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Wrapper class Hierarchy



Q. WAP to print the sum of two nos. with the help of wrapper class.

```
//prog to sum of two nos. through
//wrapper class
class Xyz
{
    public static void main(String args[])
    {
        Integer x=new Integer(25);
        Integer y=new Integer(500);
        Integer r;
```

```
        r = x+y;
        System.out.println("Sum is "+r);
    }
}
```

Byte Wrapper class

Wraps primitive datatype byte to an object.

- → Byte((byte) value) } constructor
- → Byte(String)
- → byte parseByte(String) converts string to byte. } static method
- → Byte valueOf(String) converts string to Byte.
- → Byte valueOf(byte) converts byte value to Byte
- → String toString(byte) converts byte to string
- → byte byteValue() → Non static method converts a Byte object to byte value

Q. WAP to explain the example of byteValue().

Prog:-
```
//Example of byteValue()
class Demo
{
    public static void main(String args[])
    {
        Byte b = new Byte("50");
        byte x;
        x = b.byteValue();
        System.out.println(x);
    }
}
```

Short Wrapper class:-

Wraps primitive datatype short to an object.

- → Short shortValue() → converts short object to short value
- → Short((short) value) } constructor
- → Short(String)
- → Short parseShort(String) converts string to short } static
- → Short valueOf(String) converts string to short
- → Short valueOf(Short) converts short value to short object
- → String toString(Short) → convert short to st...

Integer Wrapper class

Wraps primitive datatype int to an object.

- Integer(int) } constructor
- Integer(String) }
- int parseInt(string) — converts string to int.
- Integer valueOf(string) — converts string to Integer.
- Integer valueOf(int) — converts int value to Integer
- String toString(int) — converts int to string
- int intValue() → non-static — converts a Integer object to int value


```
// example of intValue()
class Demo
{
    public static void main(String args[])
    {
        Integer m = new Integer(50);
        Integer x;
        x = m.intValue();
        System.out.println(x);
    }
}
```

Character Wrapper class :-

Wraps primitive datatype char to an object.

- Character(char) — constructor
- Character valueOf(char) — To convert a char datatype to object
- String toString(char) — To convert a char datatype to String
- char charValue() — To convert an object to char datatype

15
Sunday • WK 07 (046-319)
February 2015

Q. WAP to explain the example of charValue().

Prog:-
```
//example of charValue()
class Demo
{
    public static void main(String args[])
    {
        Character m = new Character('N');
        char x;
        x = m.charValue();
        System.out.println(x);
    }
}
```

```
//example of ShortValue()
class Demo
{
    public static void main(String args[])
    {
        Short m = new Short("500");
        Short x;
        x = m.ShortValue();
        System.out.println(x);
    }
}
```

February 2015
16
WK 08 (047-318) • Monday

Long Wrapper class

Wraps primitive datatype long to an object.

- constructor
  - → Long((Long) value)
  - → Long(String)
- static
  - → long parseLong(string) converts string to long
  - → Long valueOf(long) converts long value to long
  - → Long valueOf(string) converts string to Long.
  - → String toString(Long) converts long to string
- non-static
  - → Long Longvalue() converts a Long object to Long

FEBRUARY 2015

WK	M	T	W	T	F	S	S
05							1
06	2	3	4	5	6	7	8
07	9	10	11	12	13	14	15
08	16	17	18	19	20	21	22
09	23	24	25	26	27	28	

17 February 2015
Tuesday • WK 08 (048-317)

Q. WAP to explain the example of Long value().

```
class Demo
{
    public static void main(String args[])
    {
        Long m = new Long("10");
        Long x;
        x = m.longValue();
        System.out.println(x);
    }
}
```

Float Wrapper class:-

Wraps primitive datatype float to an object.

- Constructor
  - → Float((float)value)
  - → Float(String)
- Static
  - → float parseFloat(String) converts string to float
  - → Float valueOf(String) converts string to float

February 2015 18
WK 08 (049-316) • Wednesday

- static
  - → Float valueOf(float) converts float value to float
  - → String toString(float) converts float to string
- Non-static
  - → float floatValue() converts a float object to float value

Q. WAP to explain the example of floatValue().

Prog:-

```
class Demo
{
    public static void main(String args[])
    {
        Float m = new float(45.30);
        float x;
        x = m.floatValue();
        System.out.println(x);
    }
}
```

FEBRUARY 2015

WK	M	T	W	T	F	S	S
05							1
06	2	3	4	5	6	7	8
07	9	10	11	12	13	14	15
08	16	17	18	19	20	21	22
09	23	24	25	26	27	28	

19 February
Thursday • WK 08 (050-315) 2015

Double Wrapper class :-

Wraps primitive datatype double to an object.

- constructor
  - → Double(double)
  - → Double(String)
- static
  - → double parseDouble(String)
    converts string to double
  - → Double valueOf(String)
    converts string to Double
  - → Double valueOf(double)
    converts double value to Double
  - → String toString(double)
    converts double to string
- non-static
  - → double doubleValue()
    converts a Double object to double value

February 20
2015 WK 08 (051-314) • Friday

Q. WAP to explain the example of doubleValue().

```
class Demo
{
    public static void main(String args[])
    {
        Double m = new Double("35.4326");
        double x;
        x = m.doubleValue();
        System.out.println(x);
    }
}
```

Boolean Wrapper class :-

Wraps primitive datatype boolean to an object.

- Constructor
  - → Boolean(boolean value)
  - → Boolean(String)
- static
  - → Boolean parseBoolean(String)
    converts string to boolean
  - → Boolean valueOf(String)
    converts string to Boolean

→ Boolean valueOf(String)
converts String to Boolean

static
→ Boolean valueOf(boolean)
converts boolean value to Boolean

→ String toString(boolean)
converts boolean to String

non-static → boolean booleanValue()
converts a Boolean object to boolean value.

Q. WAP to explain the example of booleanValue():

```
class Demo
{
    public static void main(String args[])
    {
        Boolean m = new Boolean(1);
        boolean x;
        x = m.booleanValue();
        System.out.println(x);
    }
}
```

23/3/15

23 February
Monday • WK 09 (054-311) 2015

Creating Frame in Java :-

Frame class is used to create a frame in java.

There are two methods in java through which a frame can be created in our program:-

1. By inheriting / extending Frame class.
2. By creating the object of Frame class.

Q. WAP to explain the example of thread having a button on that frame.

```
//example of thread having a button
//on that frame

import java.awt.*;
class Show extends Frame
{
    Show()
    {
        setTitle("This is frame");//super("This is frame");
        Button b=new Button("Click Here");
        b.setBounds(30,100,80,30);
        add(b);
        setSize(300,400);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String args[])
    {
        Show s=new Show();
    }
}
```

① eclipse → ⊜ → Double click to open
② File → new → java project
③ After typing a new project name → click on Next
④ Now click on finish to finish the screen.
⑤ To add a new page → RC on src folder of project and select new & then choose class.

February 2015 WK 09 (055-310) • Tuesday 24

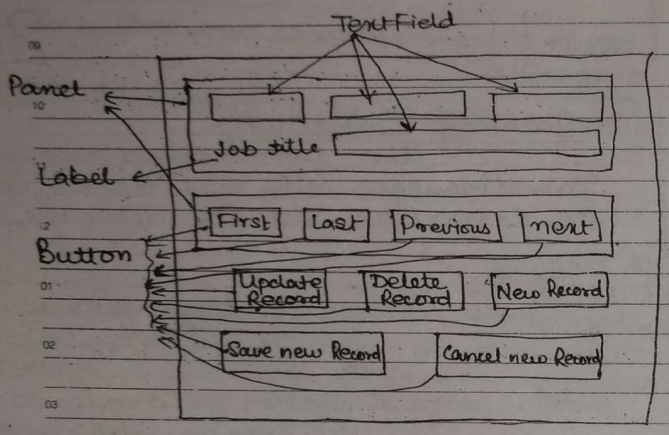
```
//By creating the object of Frame class

import java.awt.*;
class Show
{
    Show()
    {
        Frame m=new Frame("This is demo frame");
        Button b=new Button("Click Here");
        b.setBounds(30,100,80,30);
        m.add(b);
        m.setSize(300,400);
        m.setLayout(null);
        m.setVisible(true);
    }
    public static void main(String args[])
```

25 Mini Project

Wednesday • WK 09 (056-309)

February 2015



Services
↳ Databases
↳ Java DB
↳ Create Database

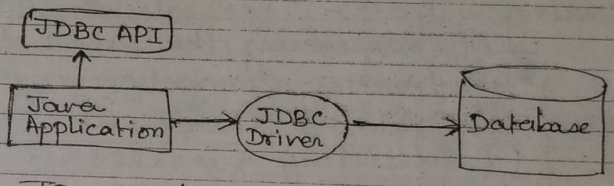
February 2015

26

WK 09 (057-308) • Thursday

Java Database Connectivity (JDBC):-

It is a java API (Application Programming Interface), which is used to connect and execute query to the database. JDBC API uses JDBC drivers to connect to the database.



JDBC driver:-

It is a software component that enables java application to interact with the database.

There are four types of JDBC drivers:-

(1) JDBC - ODBC bridge driver
(2) Native - API driver (Partially java driver)
(3) Network protocol driver
(4) Thin driver (fully java driver)

FEBRUARY 2015

WK	M	T	W	T	F	S	S
05							1
06	2	3	4	5	6	7	8
07	9	10	11	12	13	14	15
08	16	17	18	19	20	21	22
09	23	24	25	26	27	28	

MAR APR MAY JUN

27

Friday • WK 09 (058-307)

February 2015

Steps to connect to the Database in java:-

There are five steps through which any java application can connect with the database in java using JDBC:-

1. Registering the driver class
2. Creating Connection
3. Creating statement
4. Executing queries
5. closing Connection

net-beans

(1) Registering the driver class:-

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver.

Syntax:-

public static void forName(String className) throws ClassNotFoundException

↓ represents driver

Ex:- Class.forName("oracle.jdbc.driver.OracleDriver");

February 2015

28

WK 09 (059-306) • Saturday

(2) Creating Connection:-

The getConnection() method of DriverManager class is used to establish the connection to the database.

Syntax:-

public static Connection getConnection(String url) throws SqlException

or

public static Connection getConnection(String url, String user, String password) throws

represents RDBMS software & its driver

Connection m = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xyz", "Shailesh", "");

↓ Interface

database name

local/your computer

port no.

(3) Creating Statement:-

The createStatement() method of Connection interface is used to create statement. The reference of Statement interface is responsible to execute queries with the database.

public Statement createStatement() throws SqlException

Ex: Statement s = m.createStatement();

(4) Executing Queries:-

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax:-

public ResultSet executeQuery(String sqlcommands) throws SqlException

Ex: ResultSet rr = s.executeQuery("select * from employee");
(employee → table name)

```
while(rr.next())
{
    Sop ln(rr.getInt(1) + " " + rr.getString(2));
}
```
(1 → column no./name; 2 → column no.)

(5) Closing Connection:-

The close() method of Connection interface is used to close the connection.

Syntax:-

public void close() throws SQLException

Ex: m.close();

To add database driver in NetBeans:-

Right click on Library of Project and choose "Add JAR/Folder ...".

Program Files(X86) → Java → JDK1. → db → lib → derbyclient

derbyclient.jar → open

IMPORTANT Source No Design/History 2015

```
import java.sql.*;
public class MyEmployeeFrm extends
                        javax.swing.JFrame
{
    Connection cn;
    Statement st;
    ResultSet rs;
    public void myConnect()
    {
        try
        {
            String m="jdbc:derby://localhost:
                        1527:webacademy";
            String nm="aayush";
            String paas="shankar";
            cn=DriverManager.getConnection
                        (m,nm,paas);
            st=cn.createStatement();//select *
                        from emp
            String mmm="select * from emp";
            st=cn.createStatement();
            rs=st.executeQuery(mmm);
            rs.next();
            int id=rs.getInt(1);
            String a=rs.getString(2);
            String b=rs.getString(3);
            String c=rs.getString(4);
```

ACTION PLAN

2015 MARCH

Schedule your appointments, events and important dates.

01 SUN	02 MON	03 TUE	04 WED
05 THU	06 FRI	07 SAT	08 SUN
09 MON Job	10 TUE	11 WED	12 THU
13 FRI	14 SAT	15 SUN	16 MON
17 TUE	18	19 THU	20
21	22 SUN	23 MON	24 TUE
25 WED	26 THU	27 FRI	28 SAT
29 SUN	30 MON	31 TUE	

FIRST LAST PREVIOUS NEXT

Update Record Delete Record New Record

Save New Record Cancel New Record

JAN FEB MAR APR MAY JUN

Perception...

We see things the way we are, whereas we should see them the way they are.

Inspirations

01
Sunday • WK 09 (060-305)
March 2015

```
import java.Sql.*;
public class Shruti extends javax.swing.JFrame
{
    int cr;
    Connection con;
    Statement st;
    ResultSet rs;
    public Shruti()
    {
        initComponents();
        myyconnect();
    }
    public void myyconnect()
    {
        try
        {
            String mpath="jdbc:derby://localhost:1527/webacademy";
            String musr="aayush";
            String mpass="shankar";
            con=DriverManager.getConnection(mpath,musr,mpass);
            st=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);
            String mcmd="select * from emp";
            rs=st.executeQuery(mcmd);
            rs.next();
            int a=rs.getInt(1);
            String b=rs.getString(2);
            String c=rs.getString(3);
            String d=rs.getString(4);
            String e=Integer.toString(a);
```

March 2015
02
WK 10 (061-304) • Monday

```
            mtxt1.setText(e);
            mtxt2.setText(b);
            mtxt3.setText(c);
            mtxt4.setText(d);
        }
        catch(SQLException e)
        {
            System.out.println(e);
        }
    }
```

APR MAY JUN

MARCH 2015

WK	M	T	W	T	F	S	S
09	30	31					1
10	2	3	4	5	6	7	8
11	9	10	11	12	13	14	15
12	16	17	18	19	20	21	22
13	23	24	25	26	27	28	29

03 March 2015
Tuesday • WK 10 (062-303)

Coding of FIRST Button & LAST

```
public void jButton1ActionPerformed(java.
{
    try
    {
        rs.first(); ⟶ rs.last();
        int a=rs.getInt(1);
        String b = rs.getString(2);
        String c =rs.getString(3);
        String d = rs.getString(4);
        String e= Integer.toString(a);
        mtxt1.setText(e);
        mtxt2.setText(b);
        mtxt3.setText(c);
        mtxt4.setText(d);
    }
    catch(SQLException e)
    {
        System.out.println(e);
    }
}
```

Coding of Previous Button & Next

```
private void jButton3ActionPerformed(— )
{
    try
```

March 2015 04
WK 10 (063-302) • Wednesday

```
        if (rs.previous()) ⟶ (rs.next())
        {
aut.event.ActionEvent evt)
            int a =rs.getInt(1);
            String b = rs.getString(2);
            String c = rs.getString(3);
            String d= rs.getString(4);
            String e=Integer.toString(a);
            mtxt1.setText(e);
            mtxt2.setText(b);
            mtxt3.setText(c);
            mtxt4.setText(d);
        }
        else
        {
            rs.next(); ⟶ rs.previous();
            JOptionPane.showMessageDialog
                (this,"End of File");
        }
    }
```

MARCH 2015

WK	M	T	W	T	F	S	S
09	30	31					1
10	2	3	4	5	6	7	8
11	9	10	11	12	13	14	15
12	16	17	18	19	20	21	22
13	23	24	25	26	27	28	29

APR MAY JUN

05 March 2015
Thursday • WK 10 (064-301)

Coding of [UPDATE] Button

```
private void jButton5ActionPerformed(—)
{
  try
  {
    String a=jTextField1.getText();   // mtxt1
    String b=jTextField2.getText();   // mtxt2
    String c = mtxt3.getText();
    String d = mtxt4.getText();
    int e= Integer.parseInt(a);

updation in Resultset {
    rs.updateInt(1,e);
    rs.updateString(2, b);
    rs.updateString(3, c);
    rs.updateString(4, d);
update table row} rs.updateRow();
    JOptionPane.showMessageDialog(this,
                "Record updated");
  }
  catch(SQLException e)
  {
    System.out.println(e);
  }
}
```

06 March 2015
WK 10 (065-300) • Friday

Coding of [New Record] Button

```
Private void jButton7ActionPerformed(-)
{
  try
  {
    cr=rs.getRow();
    mtxt1.setText("");
    mtxt2.setText("");
    mtxt3.setText("");
    mtxt4.setText("");
    jButton1.setEnabled(false);
    jButton2.   "
    jButton3.   "
    jButton4.   "
    jButton5.   "
    jButton6.   "
    jButton7.   "
  }
  catch (SQLException e)
  {
    Sop ln(e);
  }
}
```

APR MAY JUN

MARCH 2015

WK	M	T	W	T	F	S	S
09	30	31					1
10	2	3	4	5	6	7	8
11	9	10	11	12	13	14	15
12	16	17	18	19	20	21	22
13	23	24	25	26	27	28	29

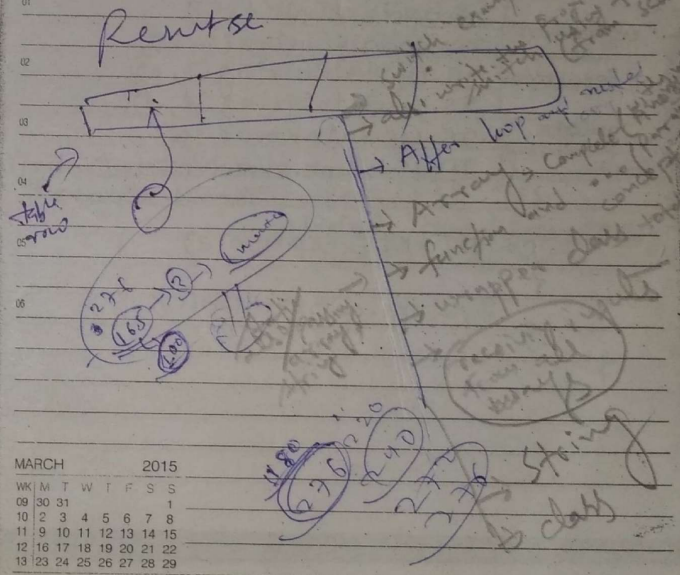
07 March 2015
Saturday • WK 10 (066-299)

Coding of [CANCEL NEW RECORD] Button

```
try
{
    rs.absolute(cr);   // send to particular record no. ; current record
    mtxt1.setText(Integer.toString(rs.getInt(1));
    mtxt2.setText(rs.getString(2));
    mtxt3.setText(rs.getString(3));
    mtxt4.setText(rs.getString(4));
    jButton1.setEnabled(true);
    jButton2.   "
    jButton3.   "
    jButton4.   "
    jButton5.   "
    jButton6.   "
    jButton7.   "
}
catch(SQLException e)
{
    Sopln(e);
}
```

March 2015 08
WK 10 (067-298) • Sunday

After loop
function ~~length~~ break, Continue
~~String~~ and passing ~~string~~ built-in functions
wrapped day
Array ~~initialize~~ and types as well as
passing array to function



MARCH 2015

WK	M	T	W	T	F	S	S
09	30	31					1
10	2	3	4	5	6	7	8
11	9	10	11	12	13	14	15
12	16	17	18	19	20	21	22
13	23	24	25	26	27	28	29