Java(Initial)

J2se – System/networking Project.

J2EE – Web Oriented

J2ME – Mobile Edition/Micro Edition.

1. Platform independent:

It run on any OS.

1. Multithreading :

Running different Lines of code in a single program simultaneously using Thread class.

1. Client/Server technology:

Socket programming is used to Client and Server Networking.

i)ServerSocket – act as a server(Class)

ii)Socket – act as client.

1. GUI:

Applets and Swing for Developing GUI.

1. OOPS

Late 1980 – Oak

Ealry 1992 –Java(James,Patrick,Chris).

|  |  |
| --- | --- |
| c | Java |
| Structural Oriented | OOPS |
| #include<stdio.h> Header File | No Header files we use import Packages import java.io.\*;  Import java.lang.Thread;  Import java.net.\*;  Import java.applet.\*;  Import java.swing.\*; |
| 3. 32 Keywords | 49 Keywords  No – sizeof,extern,register,signed,unsigned. |
| 4.Primitive and User defined datatypes | Only Primitive Data Types. |
| #define A 40 – macros….(Global Decleration) | No global Decleration. |
| Pointers are there | No pointers but we can access the address by creating Object. |
|  |  |
|  |  |

Object: Instance of the class It hold the reference

Class: Blue Print /Template.

Encapseulation: Wrappimg of data into Single(data hiding).

Public Methods and Private Variables.

Abstraction: Hiding the information from class.

Inheritance: Accessing the properties of in class into another(Code Reusability)

Single,mulitip;e,multilevel,hybrid,hierachial

Poly morphism – One Class takes many Forms.

1.Static/Compile Polymorphism – Method/Constructor/Operator OverLoading.

2. Dynamic/Runtime-- Method Overriding.

Constructor:

Whenever we create object for a class then memory allocated to object…That memory allocation is called Constructor.

Constructor invoke when object is created.

It is used to initialize the value of object.

By default all class have default Constructor.

When we create own Constructor if the constructor and class name is same then default constructor is destroyed.

Constructor can have access Specifier.

Constructor should not have a return type.

Constructor

Class A

{

A()

{

}

Void A()

{

}

}

Destructor – Deallocation of Memory. (~)

|  |  |
| --- | --- |
| C++ | Java |
| 1.Object:  Instance of the class It hold the reference | Instance of the class It hold the reference |
| Class: Blue Print /Template. |  |
|  |  |
|  |  |
| Single,mulitip;e,multilevel,hybrid,hierachial Inheritance | We don’t have multiple Inheritance instead we have interface |
| 1.Static/Compile Polymorphism – Method/Constructor/Operator OverLoading.  2. Dynamic/Runtime-- Method Overriding. | In java there is no Operator OverLoading. |
| Constructor is same for both | In Java we don’t have destructor.  Instead We have 1.Automatic garbage Collection.  2.System.gc/RunTime.gc () Where it automatically calls protected void finalize()  {  //Contains the resources to be deallocated and invoked only once in lifecycle of prg  Lifecycle of prg  } |
| String is a datatype | String is class/Literal |

jdk,jre,jvm,jit

jdk:

jre+Extra tools to develop the Java Program.

Javac is a compiler while java is interpreter.

Jvm is used to read and execute the byte code.

Jit is present in the JVM is Just in time compiler..It converts byte code to native machine code at the run time.

JRE is used to provide the supporting libraries files to java program.

Fundamentals of Java:

Creation of package:

package com.packagename.className;

package com.pack //Package Declaration

//import java packages.

Import java.io.\*;

Import java.lang.\*;

Class Sample

{

Public static void main(String[] args)

{

System.out.println(“Hello world”);

}

}\

Coding Standards

1.Class

Starting of Each Word Should be Capital.

2.Method:

From Second word onwords Starting letter onwords capital.

3.Variables:

Everything is Small

4.Constants:

Everything is Capitals.

5.Identifiers:

Name given for Clas,Variables and Methods.

Always Starting With Alphabets and

TWo Special Characters \_ and $ can

Contains numbers.

Keywords:

49 Keywords.

Goto,const are keywords of java but If we use get Compilation error….Instead of goto use Continue and for Const Use Final.

Three Reserved Words..True false for Boolean.

Null for Object.

Data Types:

Java contain only Primitive data type.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Byte | Min | max |
| byte | 1 | -2^7 | 2^7 |
| short | 2 | -2^15 | 2^15-1 |
| int | 4 | -2^31 | 2^31-1 |
| long | 8 | -2^63 | 2^63-1 |
| float | 4 |  |  |
| double | 8 |  |  |
| char | 2 byte unsigned integer | 0 | 65,535 |
| Boolean | 1 only give true or false |  |  |

Literal:

How to store the Value:

1. Int Literal:
2. Decimal Literal int a=33;int b=276437;
3. Octal Literal int a=01;02,03,04,05,06,07,010,011… Octal Literal preceded with 0.
4. Hexa Decimal Interal…preceded wit 0 and x 0x13;
5. Float Literal:

Float value should end with f/F..like float=a=3.14f;

By default all the decimal values are taken as double by Java.

1. Double Literal:

double a=3.14;

double a=3.14d;

double a=3.14;

1. Short Literal:

Short a=10;

1. Long Literal:

Long a=10;

Long a=10l;

Long a=10L;

1. Boolean Literal:

boolean b1=true;

boolean b2=false;

1. char a=’c’;

char a=”b”;

char a=’ab’;

char a=5;

char a =-23;

char a=’\u0001’;//Unicode representation

1. String:

Object Declaration with null reference.

String s1=null;

Thread t=null;

Variables:

Identifier used to store the values.

Two types:

Instance variable and Class Variable

Intance Variable:

Any variable declared inside the class outside the method.

No need to initialize It takes Default values Depending on data types int,short byte,long=0;

float,double=0.0;

boolean = false;

Object=null;

char= /u0000;

Local Variable:

Any variable declared inside the method compulsory It should be Initialized otherwise compilation error Occurs.

Access Specifier /Access Modifier:

Public,Private,Protected and Default are the Access Specifiers in java….The Default Access Specifier is default.

Public : Access in any where.

Private: Only access within a class.

Protected: can Access any class wothi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Visibility | Default | Public | Private | Protected |
| Same Class same Package | Y | Y | Y | Y |
| Different Class and Same Package(Non-Subclass) | y | y | N | N |
| Diff class different Package(subclass) | No | Yes | No | Yes |

Arithmatic Operators:

+,\*,-,/;

Modulus Operators:

%;

Relational Operators:

Equality and Inequality Operators:

== and !=

Assignment Operator:

=;

Compound Assignment:

+=,\*=,-=,/=;

Increment and Decrement Operator:

Bit wise Operator.

& Bitwise And

| Bitwise OR.

^-Bit wise XOR.

~- Ones Complement.

>> Left Shift.

<< Right Shift.

8>>2 n\*2^S (8==n,2==s)

8<<2 n/2^S

Logical Operator:

& Logical AND.

| Logical OR.

! Logical NOT.

Short Circuit Logical Operator:

Used to Check the Condition

&&

||

New Operator:

Three ways of creating an Object.

Type1:

A a = new A();

Type 2:

A a;/Object Declaration It contains null reference.

a = new A();

3) new A();

Dot Operator:

To access the Methods and varaiable;

Instance of Operators:

Ternary Operator:

Z = a>b?a:b;

Type Conversion:

Convert one Data type to another data type.

1. Implicit Conversion-Lower to Higher data Type.
2. Explicit Conversion-Higher to Lower.

Execution Control Statements in java:

1. Conditional Statements:
2. If Else Statements:

If(condition)

{

}

Else

{

}

1. Switch

Switch(exp) ---char,int,byte,short,enum,String(jdk1.7)

{

Case args: (args Should be final)

//stmt

Break;

….

…..

…..

Default:

//stmt

break;

}//don’t use continue in switch.

1. Looping Statements:
2. For Loop
3. While Loop
4. Do-While loop
5. Flow Breaking Statements:
6. Break;

It stops the entire Iteration.

1. Labeled break
2. Un Labeled break
3. Continue;

Stops the Current Iteration

1. Labeled Continue
2. Un Labeled Continue
3. Return

Transfer the Control back to the Calling Program.

Arrays:

Int a[];

Int a[] = new int[4];

* It is correct

Int a[4] = {1,2,3,4};

* It is wrong we have to use new operator

Int a[];

a[0]=1; a[1]=2; a[2]=3;

* It is correct

int[] a = a new int[4];

int a[] = new int

Anonymous Array:

Int a[] = new int[]{1,2,3,4};

Int a[] = {1,2,3,4};

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

sop(a[i]);

for(int a1:a)//For Each STMT from jdk1.5 only printing

{

Sop(a1);

}

For Each:

1. Variable should declared only inside for loop of same data type
2. 1D array stored in variable,2D stored in 1D and so on…

Int a[][] = { { a,2},{3,4}};//R

Int a[2][2]= {{1,2},{3,4}};/W

Int a[][]=new int[5][5];/R

Int[][] a= new int[5][5];/R

Int[] a[]=new int [5][5];/R

Int a[][] = new[5][5];

a[0][0] = 1; a[0][1] = 2; a[1][0] = 3;

int a[][] = new int[5][]; //arrays of array

a[0] = new int[2];

a[1] = new int[1];

a[0][0] = 1; a[0][1] = 2; a[1][0] = 3;//only 12 bytes

int a[][]= {{1,2},{3,4}};

for(int a1[]:a)

for(int a2:a1)

sop(a2);

}

}

STRINGS :

It is a class/Literal in java.

1. String is an immutable class;

Where we can’t increase or decrease the size

Strings Constructors:

String()

String(String s);

String(byte[] b,int start,int end);

String(char[] c,int start,int end);

String(byte[] b);

String(char[] c);

Char c[] = {‘j’,’a’,’v’,’a’};

String s = new String(c);

Sop(s); //o/p java)

Methods in String class:

toString method is used to Override

charAt returns single character

void getChars() returns Group of Character

byte[] getBytes() Convert String to the byte array

char[] toCharArray() converts String to Char Array.

Boolean startsWith(String)

Boolean endsWith(String)

int length() used to get the length of the String .

boolean equals()

Checks the equality of Contents by Considering the case.

Boolean equalsIgnoreCase(String s)

== () Object Reference

String substring(int start)

String substring(int start, int end)

Int compareTo(String s)

Comparing and sorting function.

Int compareToIgnoreCase(String s).

Int indexOf(char c)

Return the first occurrence of the given String.

Int lastIndexOf(char c)

Return the Last Occurrence of the given String.

String toUpperCase()

toLowerCase()

String trim()

String concat(String s)

String replace(char ori,char replace)

String[] split(String delimiter)

Static String format(String format,String value)

Used to change the Format.

Boolean regionMatches(Boolean ignorecase,int start,String value,int howmany);

Where it matches the part of the String

Wrapper Class:

The class that supports the primitive datatype to perform operation on that datatype.

All wrapper class are also immutable.

Each primitive datatype has separate Wrapper Class.

int – Integer

byte – Byte

char – Character

float—Float

double – Double

boolean – Boolean

short – Short

long – Long

Object Class is the Super class of all class.

Integer Byte Float Short Double is under class Number…Number is under Object Class.

Where as Boolean and Char is under Object.

Integer:

To perform the Operation in int dataTypes.

Constructors:

Integer(int a)

Integer i = new Integer(42);

sop(i);

Integer(String s)

Integer i2 = new Integer(“44”);

sop(i2);

Integer i3 = new Integer(“abc”); //Throws number Format Exception.

Methods:

public static int parseInt(String s)

It coverts String to data types

public static int parseInt(String s,int radix)

int intValue()

It converts Wrapper class to DataType

Static Integer valueOf(String s)

It convers String to Wrapper class.

Byte Wrapper Class

Constructor:

Byte which take byte as an argument

Byte which take any String as an Argument.

Byte b1 = new Byte((byte)14);

Byte b2 = new Byte(“14”);

Short s1 = new Short((short)20);

Short s2 = new Short(“20”);

Long l = new Long(10);

Long l = new Long(“105”);

Boolean

Constructor:

Boolean which take boolean as an argument

Boolean which take any String as an Argument.

Other than true in String argument leads to False.

In String Argument Case insensitive.

Float Wrapper Class:

Constructors:

1. Float(double d)

Float f1 = new Float(3.14);

1. Float(float f)

Float f2 = new Float(3.14f);

1. Float(String s)

Float f3 = new Float(“3.14”);

Methods:

static int compare(float f1,float f2)

float a=3.14f,b-3.14f;

int compareTo(Float f)

static boolean isNaN(float f);

Double Wrapper Class:

Constructor:

Double which take double as an argument.

Double which take String as an argument.

Character Wrapper Class:

Constructor:

We have only one constructor Character(char)

Method:

Static Boolean isLetter(char ch)

Static Boolean isDigit(char ch);

Static Boolean isLetterOrDigit(char ch);

Static Boolean isUpperCase(char ch);

Static Boolean isLowerCase(char ch);

Static Boolean isWhiteSpac(char ch);

Auto-Boxing ,Un-Boxing:

Available only from jdk 1.5.

Auto-Boxing:

Automatic Conversion of datatypes to Wrapper class

Un-boxing:

Automatic conversion of Wrapper class to datatypes.

|  |  |
| --- | --- |
| With autoboxing | Without autoboxing |
| Integer I;  int j;  i=5;  j=10;  j=I;(un boxing) | Integer I;  Int j;  i= new Integer(5);  j=10;  j=i.intValue() |

Scannner:

Present in Util.\* package

It is used to get input from the user based on datatypes.

Scanner s =new Scanner(System.in);

String s1 = s.nextLine();

String s2 = s.next();

Int i=s.nextInt();

**import** javax.swing.plaf.synth.SynthSpinnerUI;

**class** A

{

String s;

A(String s1)

{

s=s1;

}

**public** String toString()

{

**return** s;

}

}

**public** **class** Sample1 {

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

// **TODO** Auto-generated method stub

A a1=**new** A("Hello");

System.***out***.println(a1);

String s2="welcome";

System.***out***.println(s2.charAt(2));

String groupCharacters = "This is the of getChars";

**int** start = 10;

**int** end = 14;

**char** buf[] = **new** **char**[end-start];

groupCharacters.getChars(start, end,buf,0);

System.***out***.println(buf);

String s3 = "Manikandan.J";

**byte** b[] = s3.getBytes();

**for**(**byte** b1: b)

{

System.***out***.println(b1);

}

**char** c[] = s3.toCharArray();

**for**(**char** c1: c)

{

System.***out***.println(c1);

}

System.***out***.println("Manikandan".startsWith("Mani"));

System.***out***.println("Manikandan".endsWith("dan"));

System.***out***.println(s3.length());

String s4 = **new** String("KAMAL HASSAN");

String s5 = **new** String("kamal Hassan");

System.***out***.println(s4.equals(s5));

System.***out***.println(s4.equalsIgnoreCase(s5));

System.***out***.println(s4==s5);

String s8 = s4;

String s6 = "hello";

String s7 = "hello";

System.***out***.println(s6.equals(s7));

System.***out***.println(s6==s7);

String s9 = "hello World";

System.***out***.println(s9.substring(6));

System.***out***.println(s9.substring(2,7));

String s10="hello";

String s11 = "hello";

String s12 = "hemlo";

String s13 = "flag";

System.***out***.println(s10.compareTo(s11));//0

System.***out***.println(s10.compareTo(s12));//-1

System.***out***.println(s10.compareTo(s13));//2

String s14 = "Now is the time for the all good men to come to their country and pay their due taxes";

System.***out***.println(s14.indexOf('t'));

System.***out***.println(s14.lastIndexOf('t'));

System.***out***.println(s14.indexOf("the"));

System.***out***.println(s14.lastIndexOf("the"));

System.***out***.println(s14.indexOf('t',10));

System.***out***.println(s14.lastIndexOf("the",60));

String s15 = **new** String(" hello world ");

System.***out***.println(s15.length());//13

System.***out***.println(s15.trim().length());//11

System.***out***.println(s15.length());//13

String s16 = **new** String("hello");

System.***out***.println(s16.concat("world"));

System.***out***.println(s16);

String s17 = "hello";

System.***out***.println(s17.replace('e','i'));

String s18 = "one-two-three";

String s19[] = s18.split("-");

**for**(String s20: s19)

{

System.***out***.println(s20);

}

String s21 = "one.two.three";

String s22[] = s21.split("\\.");

**for**(String s20: s22)

{

System.***out***.println(s20);

}

String s27 = "NO concession,no concillation, mo give and take money";

String te = "concession|concillation|(give and take)";

String s28[] = s27.split(te);

**for**(String s29: s28)

{

System.***out***.println(s29);

}

System.***out***.println(String.*format*("|%5d|",4));

System.***out***.println(String.*format*("|%-5d|",4));

System.***out***.println(String.*format*("|%05d|",4));

System.***out***.println(String.*format*("|%5s|","hello world"));

System.***out***.println(String.*format*("$%15s$","hello world"));

System.***out***.println(String.*format*("|%5.2f|",123.456));

String s30 = "ABC WINDOWs test";

System.***out***.println(s30.regionMatches(**true**,4,"windows",0,7));

String s="42";

**int** i = Integer.*parseInt*(s);

System.***out***.println(i);

**int** j= Integer.*parseInt*(s,5);

System.***out***.println(j);

Integer i1 = **new** Integer(33);

System.***out***.println("Object:"+i1);

System.***out***.println(i1.intValue());//33

Integer i2 = Integer.*valueOf*(s);

System.***out***.println(i2.intValue());

**float** af=3.14f;

**float** bf= 3.14f;

System.***out***.println(Float.*compare*(af, bf));

Float f1 = **new** Float(3.14);

Float f2 = **new** Float(3.14);

System.***out***.println(f1.compareTo(f2));

Float f12 = **new** Float(Math.*sqrt*(-4));

System.***out***.println(f12.isNaN());

}

}

Hello

l

e of

77

97

110

105

107

97

110

100

97

110

46

74

M

a

n

i

k

a

n

d

a

n

.

J

true

true

12

false

true

false

true

true

World

llo W

0

-1

2

7

80

7

70

11

48

13

11

13

helloworld

hello

hillo

one

two

three

one

two

three

NO

,no

, mo

money

| 4|

|4 |

|00004|

|hello world|

$ hello world$

|123.46|

true

42

22

Object:33

33

42

0

0

true

Object:

Instance of a class.

It holds the memory reference.

‘This’ keyword:

This refers to current class instance variable used when Parameter name and Instance Variables names are same.

“This” Constructor:

It is used to invoke different constructor of the same class

It is always present in the first line.

Polymorphism:

One class which take multiple forms:

1. Static or compile time polymorphism.

Method Overloading and Constructor Over Loading

1. Method Overloading:

Same Method name but different number order or datatype of the argument present in the same class.

It doesn’t care about the return type.

1. Constructor Overloading:

Var args:

Passing Variable no of arguments to a method and available from jdk 1.5 version.

In combination of normal arguments and variable argument …The Variable argument always last argument.

The method will have argument…

Access Specifier:

It can be applied for everything.

Non Access Specifier:

It has own its restriction

1. Static:

It is non access specifier..

When a classes declared to be static then no need to create object. Only inner class can be static outer class should not be static.

When a method is declare to be static and if it is present in the same class it is called by method name, different class means….called by class name and method name.

If a variable is declared as static it will be initialized only once and invoked as variable name in the same class,

Classname.Variable present in different class.

Static method can access only static content otherwise we have to create an object and access.

Static block will be executed before a main method.

Static import

Available from jdk 1.5 where we call static method and static variable directly without using class name.

Inheritance:

Accessing the properties of one class to another class and used for code reusability….Java uses extend keyword for achieving Inheritance.

Class A {

Base/parent/super class

}

Class B extends A

{

Derived/child/sub class

}

Super Keyword:

Used to access base class constructor, base class method and base class variable. Only in the case of base of constructor super should be present in first line.

How constructors are invoked in inheritance???

In inheritance the constructor invoked as Top down approach.

First it goes to related class constructor and check we use this or super.

Otherwise it will tries to invoke default constructor of the base class.

Is a relationship:

Wherever inheritance comes, that type of relationship is called as “ is a relationship”

Has a Relationship:

Class wheel

{

}

Class car

{

Wheel w = new Wheel(); //Car has a wheel

}

Dynamic Polymorphism:

Method Overriding:

Same method name same return type of a method.

Same number and datatype of an argument……Present in different class and the class should be inherited.

In order to have effevtive overriding.

WE have a Dynamic method ..may be create object of base class….

At compilation the compiler thinks object is created for base cls

Class..only at the time of ecectuion..The object contains the refernvr of derived clas..In this way achieve run time polymorphism.

Final Key word:

It is a non access specifier

Is the class is declared as final

Then it can be inherited…

If the variable declared as a final it can’t be changed

Accessors and Mutators method:

Accessors : Getters Method

Mutators : Setters Method.

Object Class:

It is the super class for all the classes.

1. Constructors:

Object() //default constructor.

1. Methods:

String toString()

boolean equals(Object o)

int hashCode()

returns the internal address of the values stored in the object.

If two object equal according to equal method hashcode method produces same integer.

Protected void finalise()

final void wait()

final void wait(long millisecond)

final void notify()

final void notifyAll()

Abstract Keyword:

It is a non access specifier.

When a class is declared to be abstract ..It can’t be instantiated.

When a method is declared to abstract it does not contains definition..Just It ends with semi colon.

Varaibles can’t be abstract.

A class should be declared as abstract.

If it contains an abstract method…but not Necessarily .all abstract class should have abstract methods.

Abstract class can also contains some normal methods.

The abstract class can also be inherited, at the time we have to compulsory give the definition of the abstract method…in the inherited class or define the class itself to be abstract.

Abstract class contains a default constructors.

Interface:

1. Instead of multiple inheritance we use interface.
2. Interfaces are syntactically similar to classes…Which contains method declaration and variable declaration and initialization.

Syntax: accesSpecifier interface interfaceName(methodDecleration)

By default all the interfaces are abstract.

So we can’t create object for interface….

By default all the interface methods are public and abstract.

By default all Interface variables are public,static and final..So Interface variables are access using interfaceName.VariableName

Interface are used using implements keyword.

If a class implements an interface we have to compulsorily provide the definition of interface method as public acess specifier or make the class to be abstract

The implemented class also can contains normal method.

The interfaces can also be herited;

In java we have aa class extends another class..

Only class implements many interface

One Interface extends many interface

An Interface without any methods and variables nothing will be inside the interface is callwd market interface.

Ex

Serialixable,

Exception handling in java:

Object 🡪 Throwable🡪 Exception .

Exception{ClassNotFound Exception,SQL Exception,IO Exception and Runtime Exception}.

Run Time Exception{Error,Arthimatic Exception,Number Format Exception,Null Pointer Exception}

Checked Exception:

Sub Classes of Exception,It will insists the programmer to surround the code using try catch or throws..Otherwise the compiler will not compile the program.

Unchecked Exception:

Subclasses of runtime Exception:

The Compiler will not insist the programmer to surround the code using try catch or throw but at the time of execution it throws the exception.

Arithmatic Exception:

Divide by Zero

Array Index out of Bounce Exception:

String index out of Bounce Exception:

Array Store Exception:

When we store invalid elements differ to the type of array is called as ASE.

Negative Array Size exception:

Class Cast Exception:

Class A

{

}

class B extends A

{

}

class C

{

}

A=b;

B=a;

Null Pointer Exception

A a = new A();

a.add();

a = null;

a.add();

Number Format Exception:

IF we tries to store the characters in int the n number format exception evokes.

Illegal Monitor State Exception

Illgeal Thread State Exception.

Keywords to handle Exception:

1. Try:

Program to be monitored for Exception is put inside try block.

ii)Catch block:

Use to caught the exception

1. Finally It is an Optional statement is used to close the resources,If it is used Whether the exception Occurs or not it will be executed after the try block;
2. Mainly used in file programming ,Data base programming and Socket Programming.

Multi Catch Statement:

Single try can have multiple catches…

Throw Keyword:

Used to manually throw an Exception.

Syntax : throw new ArithmeticException()

Finally:

Throws:

Used to throw the exception and execute as usual…It can be used only in methods….

User Defined Exception / Custom Exception:

Java.util.\* package:

It is called as Utility package or collection frame work.

Used to store collection of Objects.

Collection Interface,

Collections Class.

Collection interface:

It is a core interface to store a collection of objects…

Collection class:

It provides with static algorithm to support Util Package.

Collection Interface:

By default all the interface are abstract.

Collection

List: Ordered Duplicates Allowed.

Set: Unordered no duplicates

List:

1. ArrayList
2. Linked List
3. Vector.

Set:

1. Hash Set.
2. Linked Hash Set.
3. Sorted Set.

A)Tree set

Array List Class:

It’s a dynamic array is used to increase/decrease the size at run time.

Used for Faster Selection and Slower insertion and Deletion.

Default Capacity is 10;

Constructors:

1. Default Constructor:
2. The Constructor which takes int capacity.
3. The Constructor which takes Collection C.

Method:

1. Void add(Object obj)

to add Single object.

1. Void add(int index,Object obj)
2. Void addAll(Collection C):
3. Void addAll(int index,Collection C)
4. Object get(int index)
5. Void remove(int index)
6. Object set(int index,Object obj)
7. Int indexOf(Object obj)
8. Int lastIndexOf(Object obj)
9. Boolean contains(object obj)
10. Int size()
11. Boolean isEmpty()

Linked List:

Similar to array list..This is also a Dynamic array…Linked List is extended List interface and Double Que.

Faster insertion and Deletion.

LinkedList implements Deque double ended queue where as insertion and Deletion at both ends.

Vector:

Similar to array List…It is similar to Array List because it is also a dynamic array…But Vector is Synchronized/Thread Safe.

It is the Legacy class.

The Default capacity is 10;

1. Constructors:
2. Default Constructors:
3. Vector (int Capacitity)
4. Vector (int Capacity,int Increment)
5. Vector(Collection c)

Has Set:

Set<String> s =new HashSet<>();

s.add(“A”); m.put(key,value);

s.add(“B”); m.put(“A”

s.add(“A”)

Whenever we Add elements to hash set internally it creates map which contains the inserted value as a key and dummy object as the value.

Constructors:

1. Default Constructors:
2. HashSet(Int Capacity).
3. HashSet(int Capcity,float fillratio).
4. HashSet(int Collection C).

Fillratio ranges from 0.0 to 1.0….

Default 16 capcity.

Multiple by 0.75

If the capacity is less than the number of elements is multiplied by fill ratio…inorder to increase the capacity……

Linked Hash Set:

Prints the element in order which we are inserted.

Tree Set Class:

Used to sort the elements in alphabetical or ascending Order.

ArrayList is best in list

And Set in HashSet.

Map Interface:

Used to store a collection of object as a unique key value pair

Map is un-ordered

Map.Entery interface:

Used to access the key and values separately.

void put(Object key,Object value)

void putAll(Map m)

Object get(Object key)

void remove(Object key)

boolean containsKey(Object key)

boolean containsValue(Object value)

set entrySet() 🡪 converts both key and values to Set interface

set keyset() 🡪 converts only key to set interface.

Map extends soredmap implements Tree map

Map implements Abstract map extends Hashmap extends LinkedHashMap.

Hash Map:

Contains unique key value pair and prints in random order.

Constructors:

1. Default Constructors:
2. HashMap(Int Capacity).
3. HashMap(int Capcity,float fillratio).
4. HashSet(int Map m).

LinkedHashMap

Contains Key Value pair and print in a order in which we have inserted…

Tree Map:

Used to sort the elements based on the keys.

HashTable:

It is a legacy class similar to hashMap.

It is similar to hash map but it is synchoronized and thread safe

Properties Class:

It is the sub class of HashTable contains unique key value pair..Both should be in the form of String.

Date class in java:

Used to print date and time.

Constructors:

Default constructors :

Print Current date.

Date(long millisecond)

Prints date and time from jan 1st 1970

Methods:

Int compare(date d)

Boolean before(Date d)

Boolean after(Date d)

Void setTime(long millisecond)

Long getTime

Calender class:

IT is a abstract class used to extract the usefull information from date and time component.

Constructors:

It has only default constructors.

Public static calender getInstance(); - this is used to create object.

Public int get(int field)

Public int add(int field ,int value)

Public int set(int field,int values)

Public boolean before(Calender c)

Public booleanAfter();

Public int get actuall maximum(int field)

Public int get actuall minimum()

Public getMaximum(int field)

finalDateGetTime:

generta

finalVoidSet to(dateTOCalender.)

Gregorian Calendar

Concrete of implementation of calender class used to extract information

Constructors

Default Constructors:

It will print current date and time

GregorianCalender(int year,int mont,int year);

GregorianCalender(int year,int mont,int year,day);

Methods:

Boolean isLeapYear(int Year)

Public Static final int Year

,,,,,, int month

Month = 0-1

Public static final int(minutr.

Era:

Am/pm

,,, ,,,,, ,,,Monday to unday

Day\_half\_month

Day unferScore week

Day UnderScore\_of\_undersate the

Simple Date Format Class:

Java.text.\*;

Present in java.text.\* package is use for formatting and parsing..

Formating: Date to String.

Parsing: String to Date.

Constructors:

Default Constructors:

Constructors which take String pattern..

Patterns Available :

G:

Represents the ERA.

y: Year in Number.

Y: Represents the Week of Year.

M: Month in Year

MMMM: January

MMM:Jan

MM: 01.

W:Week in Month.

D : Day in year

d = Day in month;

E : Day name in week.

a = am/pm

H = Hour in day 0 – 23.

k = Hour in day 1 – 24.

K = Hour in am/pm 0 – 11.

h = Hour in am/pm 1 – 12

m = minutes.

s = seconds.

S = milliseconds.

Methods:

Void setLeaneant(Boolean on)

Date parse (String s)

stringFormat(Date d)

Java.io.\* ; Package.

File: access info about the existing file

Stream: Perform Operation file.

Stream is of two types :

ByteStream(read & write in bytes)

1. Abstract Input Stream(read in bytes)

FileInputStream

1. Abstract Output Stream(Write in bytes)

FileOutputStream

CharacterStream (read & write in char)

Filereader

File Writer

Reader Class:

Used to read in the form of bytes.

Used to read in the form of character.

intRead = Used to Read a single character.

Return -1 at EOF

intRead(c[])

Used to read the array of c.

And returns the no of char..

intRead(c[],int offset,int noOfBytes)

File Reader :

It extends Reader.

Used to read the data from files in the form of characters.

If file doesn’t exist..It throws FileNotFoundException

Constructors:

FileReader(String FilePath)

FileReader(String Directory Path,File Name)

FileReader(File F)

Writer Class:

Abstract class is used to write in the form of character.

void write(int b)

void write(c[])

void write(c[],int offset,int numChars)

void close()

void flush()

File Writer extends Writer

Used to write in a file in the form of Characters.

If file doesn’t exits it creates new File If exits it over ride the content.

File Writer(String FilePath,Boolean append)

Present in java.lang.\* it is mutable class present in variable no of characters.

It is synchronized or thread safe and we can’t override equals method in StringBuffer.

Constructors:

Default Constructors:

StringBuffer(String s)

StringBuffer(int capacity) default Capacity = 16.

Int length()

Void setCharAt(int loc,char c)

Void setLength(int len)

Char charAt(int loc)

Void append(String s)

Void append(int a)

Void append(Object o)

Reverse()

Delete()

deleteCharAt()

void replace(int start,int end,String s)

StringBuilder:

Similar to String Buffer but it is not synchronized and not a thread safe…Bu It gives better performance than String Buffer.

Thread

Used to run multiple lines of code in a single program simultaneously using Thread class.

Syntax:

public class Thread extends Object implements Runnable

{

}

Constructors:

Default Constructors:

Thread(String ThreadName)

Thread(String threadName,Runnable r)

Thread(ThreadGroup tg,String threadName)

Thread(ThreadGroup tg,Runnable r)

Thread(ThreadGRoup tg,Runnable r,String threadName)

Constants:

1)Thread priority.MIN = 1

2)ThreadPriortiy.Max =10;

NORM = 5;

Methods:

Public void Start()

Public Static Void sleep(Long millisecond)

There are two types of thread:

1. Main Thread

Only one main thread

Wherever PSVM present tby default thread will be running called main thread. It will be executed first and blocked to invoke the child thread.

1. Child thread

Any no of thread but only evoked after block

We create n no of Child Thread.

Life Cycle of Child Thread:

Five Stages of Child thread

Begin

Runnable Running

Blocked

Sleep() Suspend() wait()

Resume() notify() | stop()🡪 Dead

Child Thread:

Two ways :

By implementing runnable interface

Interface contains method declaration .

It acts Like a child thread,to create the thread any one of seven Constructors.

Runnable interface contains one method public void run which contains the operation or processing of child thread.

WE can’t create or pas s the argument directly to interface instead use this keyword.

We can’t create object for interface

When we print thread object o/p will be

Thread[threadname(main by default),Thread priority,(5 by default),Threadgroupname(main byb default)]

1. By extending Thread class we can invoke child thread.

Implements runnable is the better method compare with extends thread.

Synchronization:

When multiple thread acts as a single resource the thread has to utilize the resource completely or fullfledged is called synchronization.

It can be achieved by using Synchronize keyword.

Non access specifier contains own restriction…

Synchronize is also Non access Specifier…It only applied to methods or Synchronized block of object.

Synchronized(object)

{

}

Annotations:

Available from jdk 1.5

Annotations are like meta data means you are free to add your code and also applied them to variables parameters,methods ,constructors and class..

Annotations are used instead of xml file.

Annotations contains two things

1. Annotation type:

Creation of Annotation:

public @interface AnnotationName

{

}

Method declaration should not throw any Exceptions

Method declaration should not contain any parameter.

Method declaration should have a return type either String, datatype or Object.

1. Annotations:

Applying the created annotation in the java program either the class level or method level.

Two types of Annotations:

1. Simple Annotations:

Present in java.lang.\* package and it can be applied only on the java Programs

Types:

1. @Override

Ensures the annotated method is used to override the method in the super class.

1. @Deprecated

This type of annotations ensures the compiler warns you when you are use the deprecated elements of the program.

1. @Suppress warning

This type of Annotation ensures that the compiler shield the warning message in the annotated element.

1. Meta Annotations:

Annotations about annotations in used at the time of creating new annotations present in java.lang.annotaitons.\*; pacakage.

1. @Target:

Specifies where the annotations applied in the java in the program like @Target(elementType.METHOD)

@Target(elementType.Constructor)

@Target(elementType.Variable)

@Target(elementType.Type)

Applicable for any elements of the class.

1. @Retention:

Specifies where and how long the annotation at this type are to retained..Like @Retention(RetentionPolicy.SOURCE)

@ Retention(RetentionPolicy.CLASS)

@Retention(RetentionPolicy.RUNTIME)

1. @Documented:

javadoc is used to document methods,variables and constructors of a class in the html file

@Documented used to Document the annotations also using javadoc tool.

Junit testing:

It is a unit testing open Source framework for java Programming language used to test individual classes.

Configure the junit jar file inside the project.

org.junit.AssertClass:

It provides set of assert methods usefull for writing tests.

1. void assertEquals(boolean Expected,boolean actual)
2. void assertFalse(boolean condition)
3. void assertNotNull(Object obj)
4. void assertNull(Object obj)
5. void assertTrue(boolean condition)

junit Annotation:

1. @Test

This is the test method to run

1. @Before:

Run before @Test

1. @After:

Run after @Test

1. @BeforeClass:

Run Ones before any of the TestMethod

1. @AfterClass:

Run Ones after any of the TestMethod