**Day1**

**Java Vs C++**

Development wise differences

1. Java is platform independent language but c++ is dependent upon operating system.

At compilation time Java Source code(.java) converts into byte code(.class) .The interpreter translates this byte code at run time into native code and gives output.

2. Java uses both a compiler and interpreter, while C++ only uses a compiler

Syntactical differences

1. There is no final semi-colon at the end of the class definition.

2. Functions are called as methods.

3. main method is a member of class

& has a fixed form

public static void main(String[] args) -- argument is an array of String. This array contains the command-line arguments.

4. main method must be inside some class (there can be more than one main function -- there can even be one in every class)

5. Like the C++ << operator,

To write to standard output, you can use either of the following:

System.out.println( ... )

System.out.print( ... )

The former prints the given expression followed by a newline, while the latter just prints the given expression.

These functions can be used to print values of any type. eg :

System.out.print("hello"); // print a String

System.out.print(16); // print an integer

System.out.print(5.5 \* .2); // print a floating-point number

The + operator can be useful when printing. It is overloaded to work on Strings as follows:

If either operand is a String, it

converts the other operand to a String (if necessary)

creates a new String by concatenating both operands .

Features wise differences.

1. C++ supports pointers whereas Java does not support pointer arithmetic. It supports Restricted pointers.

Java references (Restricted pointers) can't be arithmatically modified.

2. C++ supports operator overloading , multiple inheritance but java does not.

3. C++ is nearer to hardware than Java.

4. Everything (except fundamental or primitive types) is an object in Java (Single root hierarchy as everything gets derived from java.lang.Object).

Java is similar to C++ but it doesn't have the complicated aspects of C++, such as pointers, templates, unions, operator overloading, structures, etc. Java also does not support conditional compilation (#ifdef/#ifndef type).

Thread support is built into Java but not in C++. C++11, the most recent iteration of the C++ programming language, does have Thread support though.

Internet support is built into Java, but not in C++. On the other hand, C++ has support for socket programming which can be used.

Java does not support header files and library files. Java uses import to include different classes and methods.

Java does not support default arguments.

There is no scope resolution operator :: in Java. It has . using which we can qualify classes with the namespace they came from.

There is no goto statement in Java.

Because of the lack of destructors in Java, exception and auto garbage collector handling is different than C++.

Java has method overloading, but no operator overloading unlike C++.

The String class does use the + and += operators to concatenate strings and String expressions use automatic type conversion,

Java is pass-by-value.

Java does not support unsigned integers.

--------------

Why java doesn't support c++ copy constructor?

Java does. They're just not called implicitly like they are in C++ .

Firstly, a copy constructor is nothing more than:

public class Blah

{

private int foo;

public Blah() { } // public no-args constructor

public Blah(Blah b) { foo = b.foo; } // copy constructor

}

Now C++ will implicitly call the copy constructor with a statement like this:

Blah b2 = b1;

Cloning/copying in that instance simply makes no sense in Java because all b1 and b2 are references and not value objects like they are in C++. In C++ that statement makes a copy of the object's state. In Java it simply copies the reference. The object's state is not copied so implicitly calling the copy constructor makes no sense.

-----------------

All stand-alone C++ programs require a function named main and can have numerous other functions. Java does not have stand alone functions, all functions (called methods) are members of a class. All classes in Java ultimately inherit from the Object class, while it is possible to create inheritance trees that are completely unrelated to one another in C++. In this sense , Java is a pure Object oriented language, while C++ is a mixture of Object oriented and structure language.

The interface keyword in Java is used to create the equivalence of an abstract base class containing only method declarations and constants. No variable data members or method definitions are allowed(true till Java 8) . C++ does not support interface concept. Java does not support multiple inheritance. To some extent, the interface feature provides the desirable features of multiple inheritance to a Java program without some of the underlying problems.(death of a diamond)

Java is running on a Virtual Machine, which can recollect unused memory to the operating system, so Java does not destructor. Unlike C++, Java cannot access pointers to do memory operation directly. This leads to a whole host of subtle and extremely important differences between Java and C++.

Furthermore, the C++ compiler does not check whether all local variables are initialized before they are read. It is quite easy to forget initializing a variable in C++. The value of the variable is then the random bit pattern that happened to be in the memory location that the local variable occupies.

Java does not have global functions and global data. Static in Java is just like global in C++, can be accessed through class name directly, and shared by all instances of the class. For C++, static data members must be defined out side of class definition, because they don't belong to any specific instance of the class.

Generally Java is more robust than C++ because:

Object handles (references) are automatically initialized to null.

Handles are checked before accessing, and exceptions are thrown in the event of problems.

You cannot access an array out of bounds.

Memory leaks are prevented by automatic garbage collection.

While C++ programmer clearly has more flexibility to create high efficient program, also more chance to encounter error.

**About JVM**

JVM has various sub components internally.

1. Class loader sub system: JVM's class loader sub system performs 3 tasks

a. It loads .class file into memory.

b. It verifies byte code instructions.

c. It allots memory required for the program.

2. Run time data area: This is the memory resource used by JVM and it is divided into 5 parts

a. Method area: Method area stores class code and method code.

b. Heap: Objects are created on heap.

c. Java stacks: Java stacks are the places where the Java methods are executed. A Java stack contains frames. On each frame, a separate method is executed.

d. Program counter registers: The program counter registers store memory address of the instruction to be executed by the microprocessor.

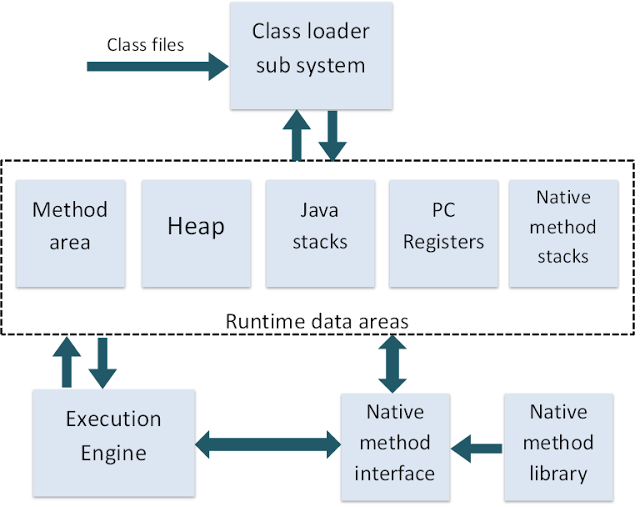
e. Native method stacks: The native method stacks are places where native methods (for example, C language programs) are executed. Native method is a function, which is written in another language other than Java.

3. Native method interface: Native method interface is a program that connects native methods libraries (C header files) with JVM for executing native methods.

4. Native method library: holds the native libraries information.

5. Execution engine: Execution engine contains interpreter and JIT compiler, which covers byte code into machine code. JVM uses optimization technique to decide which part to be interpreted and which part to be used with JIT compiler. The HotSpot represent the block of code executed by JIT compiler.

**JVM Architecture**

****

**what\_is\_jdk\_jre\_jvm**

**What is JRE?**

Together, the Java virtual machine and Java API form a "platform" for which all Java programs are compiled. In addition to being called the Java runtime system, the combination of the Java virtual machine and Java API is called the Java Platform (or, starting with version 1.2, the Java 2 Platform). Java programs can run on many different kinds of computers because the Java Platform can itself be implemented in software.

**JDK (Java Development Kit)**

Java Developer Kit contains tools needed to develop the Java programs, and JRE to run the programs. The tools include compiler (javac.exe), Java application launcher (java.exe), Appletviewer, etc…

Compiler converts java code into byte code. Java application launcher opens a JRE, loads the class, and invokes its main method.

You need JDK, if at all you want to write your own programs, and to compile them. For running java programs, JRE is sufficient.

JRE is targeted for execution of Java files

i.e. JRE = JVM + Java Packages Classes(like util, math, lang, awt,swing etc)+runtime libraries.

JDK is mainly targeted for java development. I.e. You can create a Java file (with the help of Java packages), compile a Java file and run a java file

**JRE (Java Runtime Environment)**

Java Runtime Environment contains JVM, class libraries, and other supporting files. It does not contain any development tools such as compiler, debugger, etc. Actually JVM runs the program, and it uses the class libraries, and other supporting files provided in JRE. If you want to run any java program, you need to have JRE installed in the system

The Java Virtual Machine provides a platform-independent way of executing code; programmers can concentrate on writing software, without having to be concerned with how or where it will run.

If u just want to run applets (ex: Online games or puzzles), JRE needs to be installed on the machine.

**JVM (Java Virtual Machine)**

As we are all aware when we compile a Java file, output is not an 'exe' but it's a '.class' file. '.class' file consists of Java byte codes which are understandable by JVM. Java Virtual Machine interprets the byte code into the machine code depending upon the underlying operating system and hardware combination. It is responsible for all the things like garbage collection, array bounds checking, etc… JVM is platform dependent.

The JVM is called "virtual" because it provides a machine interface that does not depend on the underlying operating system and machine hardware architecture. This independence from hardware and operating system is a cornerstone of the write-once run-anywhere value ofJava programs.

There are different JVM implementations are there. These may differ in things like performance, reliability, speed, etc. These implementations will differ in those areas where Java specification doesn’t mention how to implement the features, like how the garbage collection process works is JVM dependent, Java spec doesn’t define any specific way to do this.

**variable\_naming\_rules**

What are the rules for naming variables in java?

Answer:

All variable names must begin with a letter of the alphabet, or an underscore ( \_ ), or a dollar sign ($). Can't begin with a digit. The rest of the characters may be any of those previously mentioned plus the digits 0-9.

The convention is to always use a (lower case) letter of the alphabet. The dollar sign and the underscore are discouraged.

**data\_type\_ranges**

1. byte: The byte data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive). --- -2^7 ---- 2^7-1

2. short: The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive).

-2^15 ---- 2^15-1

3.int: The int data type is a 32-bit signed two's complement integer. It has a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647 (inclusive).

4. long: The long data type is a 64-bit signed two's complement integer. It has a minimum value of -9,223,372,036,854,775,808 and a maximum value of 9,223,372,036,854,775,807 (inclusive).

5.float: The float data type is a single-precision 32-bit IEEE 754 floating point.

Covers a range from 1.40129846432481707e-45 to 3.40282346638528860e+38 (positive or negative).

BE careful -- in assigning integer to float & vice versa.

6. double : 8 bytes IEEE 754. Covers a range from 4.94065645841246544e-324d to 1.79769313486231570e+308d (positive or negative).

7. boolean : Typically 1-bit. May take on the values true and false only.true and false are defined constants of the language. Booleans may not be cast into any other type of variable nor may any other variable be cast into a boolean.

8. char -- unsigned char. --- UTF 16

range 0----65535

**automatic\_conversions**

As per Java language specs ------

Widening primitive conversions do not lose information about the overall magnitude of a numeric value. Indeed, conversions widening from an integral type to another integral type do not lose any information at all; the numeric value is preserved exactly. Conversions widening from float to double in strictfp expressions also preserve the numeric value exactly; however, such conversions that are not strictfp may lose information about the overall magnitude of the converted value.

Conversion of an int or a long value to float, or of a long value to double, may result in loss of precision-that is, the result may lose some of the least significant bits of the value. In this case, the resulting floating-point value will be a correctly rounded version of the integer value, using IEEE 754 round-to-nearest mode

In other words even though you may lose information, you know that the value will still be in the overall range of the target type.

Widening Primitive Conversion

The following 19 specific conversions on primitive types are called the widening primitive conversions:

byte to short, int, long, float, or double

short to int, long, float, or double

char to int, long, float, or double

int to long, float, or double

long to float or double

float to double

Widening primitive conversions do not lose information about the overall magnitude of a numeric value.

Conversion of an int or a long value to float, or of a long value to double, may result in loss of precision-that is, the result may lose some of the least significant bits of the value. In this case, the resulting floating-point value will be a correctly rounded version of the integer value

To put it another way, the JLS(Java language specification) distinguishes between a loss of magnitude and a loss of precision.

int to byte for example is a (potential) loss of magnitude because you can't store 500 in a byte.

long to float is a potential loss of precision but not magnitude because the value range for floats is larger than that for longs.

So the rule is:

Loss of magnitude: explicit cast required;

Loss of precision: no cast required.

**Read me day1**

WHY Java ?

1. Simple & robust.

2. Platform or architecture independent

3. Secure

4. Automatic memory management.

------------

Pre-requisites

1. JDK 1.8(Java SE 8) must be installed using java installer.

2. 1st entry of path must be

<JDK1.8>\bin

3. How to set path ?

My Computer --- R Click -- Properties --- Advanced --- Environment variables --- System Variables --- Edit Path --- Add 1st entry as <jdk1.8>\bin

DO NOT delete existing path.

Confirm it.

How ?

Open cmd prompt

Type set path

It should show you --path info --- where 1st entry is <jdk1.8>\bin .

Type from cmd prompt

java -version

4. Create empty workspace. Create folders day wise. Create src & bin to store java sources & .class files separately.

Rules on Identifiers

Identifiers must start either with a letter, a currency character ($), or a connecting

character such as the underscore ( \_ ), cannot start with a number!

Can't use a Java keyword as an identifier.

Are Case sensitive .

Norms

class , interfaces , enum names- must start with upper case & then follow camel case

data members/methods(funs) -- must start with lower case & then follow camel case

constants -- all uppercase.

Objective --- create a java appln to display welcome msg on the console.

Legal class level access specifiers - default(scope=current pkg only), public (scope=accessible form any where)

compiler usage --- javac -d ..\bin First.java--> place the .class files to bin folder

Objective : accept 2 nums as cmd line args , add them & disp the result.

API -- java docs

java.lang => pkg name --default

Integer -- class

public static int parseInt(String s) throws NumberFormatException

Basic rules

0. Files with no public classes can have a name that does not match any of the classes in the file

1. There can be only one public class per source code file.

2. If there is a public class in a file, the name of the file must match the name

of the public class. For example, a class declared as public class Example { }

must be in a source code file named Example.java.

3. If the class is part of a package, the package statement must be the first line

in the source code file, before any import statements that may be present.

4. If there are import statements, they must go between the package statement

(if there is one) and the class declaration. If there isn't a package statement,

then the import statement(s) must be the first line(s) in the source code file.

If there are no package or import statements, the class declaration must be

the first line in the source code file.

5. import and package statements apply to all classes within a source code file.

In other words, there's no way to declare multiple classes in a file and have

them in different packages, or use different imports.

6. A file can have more than one non public class.

Automatic conversions(widening ) ---Automatic promotions

byte--->short--->int---> long--->float--->double

char ---> int

Rules ---

src & dest - must be compatible, typically dest data type must have higher magnitude than src data type.

Any arithmetic operation involving bytes --- result type=int

Any arithmetic op involving short --- result type=int

int & long ---> long

long & float ---> float

byte,short......& float & double----> double

Narrowing conversion --- forced conversion(type-casting)

eg ---

double ---> int

double ---> float

Steps for attaching scanner -- data read from console

1. import java.util.\*; or import java.util.Scanner;

2. create instance of Scanner class

constr -- Scanner (InputStream in)

System.in --- stdin

usage -- Scanner sc=new Scanner(System.in);

3.To check data type

boolean hasNextInt(),hasNextByte(),hasNextLong()......

4. To read data

int nextInt() throws InputMismatchException

double nextDouble() throws InputMismatchException

String next(),String nextLine(),boolean nextBoolean()....

-----------------------------------------------

Object oriented principles

Class Programming

Encapsulation -- consists of Data hiding + Data Abstraction

Information hiding -- achieved by private data members & supplying public accessors.

Abstraction -- achieved by supplying an interface to the Client (customer) .Highlighting only WHAT is to be done & not highlighting HOW it's internally implemented.

Adv ---security

ease of maintenance

ease of usage

ensures easy modification

Objective --- Create Java appln for follow. -represent 3D Box----tightly encapsulated class.

Data members -width,height,depth : double

constr- to init Box params.

Business logic(behaviour) --- displayBoxDetails(void ret) ,calcVolume(ret vol)

Create Java application -- which allows user to supply 3 dims from user using scanner. --- create Box object & display dims & display vol.

Pointers vs java references

pointer arithmatic is not allowed in java.

reference --- holds internal representation of address --

method local vars Vs instance data members

method local vars --- allocated on stack, def. -- uninitialized.

inst. data members --- allocated on heap, inited to def vals(eg - boolean -false,int -0,double 0.0, ref-null)

Javac doesn't allow accessing ANY (prim type or ref type) un-initialized data member.

Automatic Gargabe Collection --- to avoid mem. leaks/holes

JRE creates 2 thrds --- main thrd(to exec main() sequentially) -- fg thrd

G.C --- daemon thrd ---bg thrd --- gets activated periodically --- releases the memory occupied by un-referenced objects allocated on the heap(the obj whose no. of ref=0)

To release/close non- Java resources.(eg - closing of Db conn, closing file handles is NOT done auto. by GC)

Garbage= un -reachable object.

How to request for garbage collection ?

API of System class

System.gc()

Array handling

Arrays are created as objects in java. (array of prim types or array of refs)

syntax --

int[] arr;

arr=new int[10];

Enhanced for loop(for-each loop)

eg : double[] nums={1.4,4.56,5.67};

for (double d : nums) //d=nums[0],d=nums[1] .....d=nums[nums.length-1]

sop(d);

Objective -- Accept no of data samples(double) from User

Create array & display --confirm default values.

Accept data from User(scanner) & store it in suitable array.

Display array data from for loop.

for-each syntax.

Objective --- Accept from user --- no of boxes to be made.

Accept box dims from user & display box dims using for-each

--------------------------

Regarding Packages

What is a package ?

Collection of functionally similar classes & interfaces.

Creating user defined packages

Need ?

1. To group functionally similar classes together.

2. Avoids name space collision

3. Finer control over access specifiers.

About Packages

1. Creation : package stmt has to be placed as the 1st stmt in Java source.

eg : package p1; => the classes will be part of package p1.

2.Package names are mapped to folder names.

eg : package p1; class A{....}

A.class must exist in folder p1.

3. For simplicity --- create folder p1 -- under <src> & compile from <src>

From <src>

javac -d ..\bin p1\A.java

-> javac will auto. create the sub-folder <p1> under the <bin> folder & place A.class within <p1>

NOTE : Its not mandatory to create java sources(.java) under package named folder. BUT its mandatory to store packged compiled classes(.class) under package named folders

Earlier half is just maintained as convenience(eg --- javac can then detect auto. dependencies & compile classes ).

4. To run the pkged classes from any folder : u must set Java specific env var. : classpath

set classpath=g:\dac1\day2\bin;

With the classpath set, JVM's classloader 1st searches in the curnt folder, if not found then will continue to search in the folders specified in the classpath & try to load the pkged classes.

classpath= Java only env .var

Used by JRE's classloader : to locate & load the classes.

Classloader will try to locate the classes from current folder, if not found --- will refer to classpath entries : to resolve & load Java classes.

What should be value of classpath ---Must be set to top of pakged hierarchy(i.e .class eg : bin)

set classpath=G:\dac\dac1\day2\bin;(cmd line invocation)

**Day4**

Regarding inheritance

In OOP, we often organize classes in hierarchy to avoid duplication and reduce redundancy. The classes in the lower hierarchy inherit all the variables (attributes) and methods (dynamic behaviors) from the higher hierarchies.

A class in the lower hierarchy is called a subclass (or derived, child, extended class). A class in the upper hierarchy is called a superclass (or base, parent class).

By pulling out all the common variables and methods into the superclasses, and leave the specialized variables and methods in the subclasses, redundancy can be greatly reduced or eliminated as these common variables and methods do not need to be repeated in all the subclasses. Re usability is maximum.

A subclass inherits all the member variables and methods from its superclasses (the immediate parent and all its ancestors). It can use the inherited methods and variables as they are. It may also override an inherited method by providing its own version, or hide an inherited variable by defining a variable of the same name.

eg :

in package -- com.app.core

Person -- firstName,lastName

Student --firstName,lastName,grad year

Faculty -- firstName,lastName,yrs of experience , sme

OR

Emp-Mgr-SalesMgr scenario.

OR

Fruit -- Apple -- FujiApple

A subclass inherits all the variables and methods from its superclasses, including its immediate parent as well as all the ancestors.

It is important to note that a subclass is not a "subset" of a superclass. In contrast, subclass is a "superset" of a superclass. It is because a subclass inherits all the variables and methods of the superclass; in addition, it extends the superclass by providing more variables and methods.

Inheritance --- generalization ----> specialization.

IS A Relationship.

Why -- code re usability.

super class ---base class

sub class --derived class

keyword --extends

sub class IS A super class and something added(additonal state + additional behaviour) and something modified(behaviour ---method overriding)

Types of inheritance

1. single inheritance ---

class A{...} class B extends A{...}

2. multi level inhertance

class A{...} class B extends A{...} class C extends B{...}

3. multiple inhertiance --- NOT supported

class A extends B,C{...} -- compiler err

Why --For simplicity.

(Diamond problem)

We have two classes B and C inheriting from A. Assume that B and C are overriding an inherited method and they provide their own implementation. Now D inherits from both B and C doing multiple inheritance. D should inherit that overridden method. BUT which overridden method will be used? Will it be from B or C? Here we have an ambiguity.

Constructor invocations in inheritance hierarchy -- single & multi level.

eg -- Based on class A -- super class & B its sub class.

Further extend it by class C as a sub-class of B.

super keyword usage

1. To access super class's visible members

2. To invoke immediate super class's matching constructor --- accessible only from sub class constructor.(super(...))

eg -- Person,Student,Faculty

Emp --- id , name , basic

Manager -- id , name , basic , bonus

,SalesManager

Shape, Circle,Rectangle

LoanAccount,HomeLoanAccount,VehicleLoanAccount,

Student,GradStudent,PostGradStudent

eg :

Person -- firstName,lastName

Student --firstName,lastName,grad year

Faculty -- firstName,lastName,yrs of experience , sme

Add the functionality to show individual details.

Regarding this & super

1. Only a constr can use this() or super()

2. Has to be 1st statementt in the constr

3. Any constructor can never have both ie. this() & super()

4. super & this (w/o brackets) are used to access (visible) members of super class or the same class.

Polymorphism ---one functionality --multiple (changing) forms

1. static -- compile time --early binding ---resolved by javac.

Achieved via method overloading

rules -- can be in same class or in sub classes.

same method name

signature -- different (no/type/both)

ret type --- ignored by compiler.

eg --- void test(int i,int j){...}

void test(int i) {..}

void test(double i){..}

void test(int i,double j,boolean flag){..}

int test(int a,int b){...} ---> compiler err.

RULE -- when javac doesn't find exact match --tries to resolve it by the closest arg type(just wider than the specified arg)

solve --- EasyOver.java

(More interesting examples after boxing & var-args)

2. Dynamic polymorphism --- late binding --- dynamic method dispatch ---resolved by JRE.

Dynamic method dispatch -- which form of method to send for execution ---This decision can't be taken by javac --- BUT taken by JRE

Achieved via -- method overriding

Method Overriding --- Means of achieving run-time polymorphism

NO "virtual" keyword in java.

All java methods can be overridden : if they are not marked as private,static,final

Super-class form of method - --- overridden method

sub-class form --- overriding form of the method

Rules : to be followed by overriding method in a sub-class

1. same method name, same signature, ret type must be same or its sub-type(co-variance)

eg of co-variance

class A {

A getInstance()

{

return new A();

}

}

class B extends A

{

B getInstance()

{

return new B();

}

}

2. scope---must be same or wider.

3. Will be discussed in exeception handling.

Can not add in its throws clause any new or broader checked exceptions.

BUT can add any new unchecked excs.

Can add any subset or sub-class of checked excs.

class A

{

void show() throws IOExc

{...}

}

class B extends A

{

void show() throws Exc

{...}

}

Can't add super class of the checked excs.

example of run time polymorphism -- Car & its sub classes.

From JDK 1.5 onwards : Annoations are available --- metadata meant for Compiler or JRE.(Java tools)

Java Annotation is a tag that represents the metadata i.e. attached with class, interface, methods or fields to indicate some additional information which can be used by java compiler and JVM.

Annotations in java are used to provide additional information, so it is an alternative option for XML.

eg @Override,@Deprecated,@SuppressWarnings,@FunctionalInterface

@Override --

Annotation for javac

While overridng the method --- if u want to inform the compiler that : following is the overriding form of the method use :

@Override

method declaration

Run time polymorphism or Dynamic method dispatch in detail

Super -class ref. can directly refer to sub-class object(direct=w/o type casting) as its the example of up-casting(similar to widening auto. conversion) .

When such a super class ref is used to invoke the overriding method: which form of the method to send for execution : this decision is taken by JRE & not by compiler. In such case --- overriding form of the method(sub-class version) will be dispatched for exec.

Super -class ref. can directly refer to sub-class inst BUT it can only access the members declared in super-class -- directly.

eg : A ref=new B(); ref.show() ---> this will invoke the sub-class: overriding form of the show () method

----------------------------------------------

Applying inheritance & polymorphism

java.lang.Object --- Universal super class of all java classes including arrays.

Object class method

public String toString() ---Rets string representation of object.

Returns --- Fully qualified class Name @ hash code

hash code --internal memory representation.(hash code is mainly used in hashing based data structures -- will be done in Collection framework)

Why override toString?

To replace hash code version by actual details of any object.

eg -- Use it in sub classes. (override toString to display Account or Point2D or Emp details)

Object class method

public boolean equals(Object o)

Returns true --- If 'this' (invoker ref) & o ---refers to the same object(i.e reference equality) i.e this==o , otherwise returns false.

Need of overriding equals method ?

To replace reference equality by content equality , based upon prim key criteria.

eg : In Car scenario

(Primary key -- int registration no)

Objective ---Applying inheritance & polymorphism to organization scenario

Emp based org structure --- Emp , Mgr , Worker

Emp state--- id(primary key), name, deptId , basic

system generated ID (auto increment)

Behaviour --- get emp details -- via toString

Mgr state ---id,name,basic,dept,perfBonus

Behaviour ----get mgr details, compute net salary (formula: basic+perfBonus)

& get performance bonus.

Worker state ---id,name,basic,dept, hoursWorked,hourlyRate

Behaviour--- get worker details, compute net salary (formula: = basic+(hrs\*rate)

get hrlyRate of the worker

Design classes based on above info.

Create a tester to create org structure --- m1,w1,m2,w2

Use single for-each loop Display emp info & net sal.

From same for-each loop display mgr's bonus or worker's rate

instanceof -- keyword in java --used for testing run time type information

eg ---

Emp e =new Mgr(...);

if (e instanceof Mgr) ---rets true iff e --->Mgr obj

if (e instanceof Worker) ---rets true iff e --->Worker obj

if (e instanceof Emp)

As an extension , take above code , attach scanner , accept emp details from user & test it.

Anything "not so object oriented" about Emp , Mgr & worker hierarchy?

abstract : keyword in Java

abstract methods ---methods only with declaration & no definition

eg : public abstract double calNetsalry();

Any time a class has one or multilple abstract methods ---- class must be declared as abstract class.

eg. public abstract class Emp {....}

Abstract classes can't be instantiated BUT can create the ref. of abstract class type to refer to concrete sub-class instances.

Emp e1=new Emp(...);//illegal

Emp e1=new Mgr(....);//legal

Abstract classes CAN HAVE concrete(non-abstract) methods.

Abstract classes MUST provide constructor/s to init its own private data members.

Can a class be decalred as abstract & final ? NO

Can an abstract class be crerated with 100% concrete functionality?

Yes

eg --- Event adapter classes

Use "abstract" keyword in Emp , Mgr ,Worker hierarchy & test it

final -- keyword in java

Usages

1 final data member(primitive types) - constant.

eg -- public final int data=123;

2. final methods ---can't be overridden.

usage eg public final void show{.....}

eg -- Object class -- wait , notify ,notifyAll

3. final class --- can't be sub-classed(or extended) -- i.e stopping inheritance hierarchy.

eg -- String ,StringBuffer,StringBuilder

4. final reference -- references can't be re-assigned.

eg -- final Emp e=new Mgr(.......);

e=new Worker(.....);//compiler err

--------------------

Special note on protected

Protected members acts as default scope within the same package.

BUT outside pkg -- a sub-class can access it through inheritance(i.e just inherits it directly) & CANT be accessed by creating super class instance.

**Day 5**

**About interfaces**

Interface in Java

An interface in java is a blueprint of a class. It has public static final data members and abstract methods only.

The interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface not method body(true till JDK 1.7) . It is used to achieve full abstraction and multiple inheritance in Java.

Java Interface also represents IS-A relationship.

It cannot be instantiated just like abstract class.

Why use Java interface?

There are mainly three reasons to use interface.

1. It is used to achieve full abstraction.

2. By interface, we can support the functionality of multiple inheritance.

3. It can be used to achieve loose coupling.

(Interfaces allow complete separation between WHAT(specification or a contract) is to be done Vs HOW (implementation details) it's to be done

eg : In JDBC (java db connectivity)

java.sql.Connection i/f ----Sun

Imple. classes -- DB vendors --

Oracle --Oracle DB engine --Imple cls for Comnection i/f

Red Hat -- Mysql -- Imple cls for Comnection i/f

The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members.

Relationship between classes and interfaces

A class extends another class, an interface extends another interface but a class implements an interface.

Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.

i/f syntax

public / no mod interface Name extends i1,i2...,i3

{

//d.m --- public static final

//method -- public abstract

}

Implementation class syntax

public/no mod class Name extends SuperClsName implements i1,i2,i3.....

{

//Mandatory for concrete class --MUST implement all methods from all i/fs

}

Multiple inheritance in java

interface Printable{

void print();

}

interface Showable{

void show();

}

class A implements Printable,Showable{

public void print(){System.out.println("Hello");}

public void show(){System.out.println("Welcome");}

public static void main(String args[]){

A obj = new A();

obj.print();

obj.show();

}

}

Question

Multiple inheritance is not supported through class in java but it is possible by interface, why?

Multiple inheritance is not supported in case of class, sincee it can create an ambiguity. But it is supported in case of interface because there is no ambiguity as implementation is provided by the implementation class.

For example:

interface Printable{

void print();

}

interface Showable{

void print();

}

class TestTnterface1 implements Printable,Showable{

public void print(){System.out.println("Hello");}

public static void main(String args[]){

TestTnterface1 obj = new TestTnterface1();

obj.print();

}

}

As you can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestTnterface1, so there is no ambiguity.

Interface inheritance

A class implements interface but one interface extends another interface .

interface Printable{

void print();

}

interface Showable extends Printable{

void show();

}

class Testinterface2 implements Showable{

public void print(){System.out.println("Hello");}

public void show(){System.out.println("Welcome");}

public static void main(String args[]){

Testinterface2 obj = new Testinterface2();

obj.print();

obj.show();

}

}

Q) What is marker or tagged interface?

An interface that have no member is known as marker or tagged interface. For example: Serializable, Cloneable, Remote etc. They are used to provide some essential information to the JVM(Run time marker) so that JVM may perform some useful operation.

//How Serializable interface is written?

public interface Serializable{

}

Nested Interface in Java

Note: An interface can have another interface i.e. known as nested interface.

eg :

interface printable{

void print();

interface MessagePrintable{

void msg();

}

}

**Difference between abstract class and interface**

Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.

But there are many differences between abstract class and interface.

Abstract class Vs Interface

1) Abstract class can have abstract and non-abstract methods. Interface can have only abstract methods.

2) Abstract class doesn't support multiple inheritance. Interface supports multiple inheritance.

3) Abstract class can have final, non-final, static and non-static variables. Interface has only public, static and final variables.

4) Abstract class can have static methods, main method and constructor. Interface can't have static methods, main method or constructor.

5) Abstract class can provide the implementation of interface. Interface can't provide the implementation of abstract class.

6) The abstract keyword is used to declare abstract class. The interface keyword is used to declare interface.

7) Example:

public abstract class Shape{

public abstract void draw();

} Example:

public interface Drawable{

void draw();

}

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

**Abstract Class vs. Interface**

Java provides and supports the creation of abstract classes and interfaces. Both implementations share some common features, but they differ in the following features:

1. All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.

2. A class may implement a number of Interfaces, but can extend only one abstract class.

3. In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.

Abstract classes can implement interfaces without even providing the implementation of interface methods.

4. Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.

5. Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.

6. An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated BUT can contain a constructor

**Read me co-variance**

Regarding co variance

Overrding form of method ---

has same method name, same signature, ret type must be same or its sub-type(co-variance)

eg of co-variance

class A {

A getInstance()

{

return new A();

}

}

class B extends A

{

B getInstance()

{

return new B();

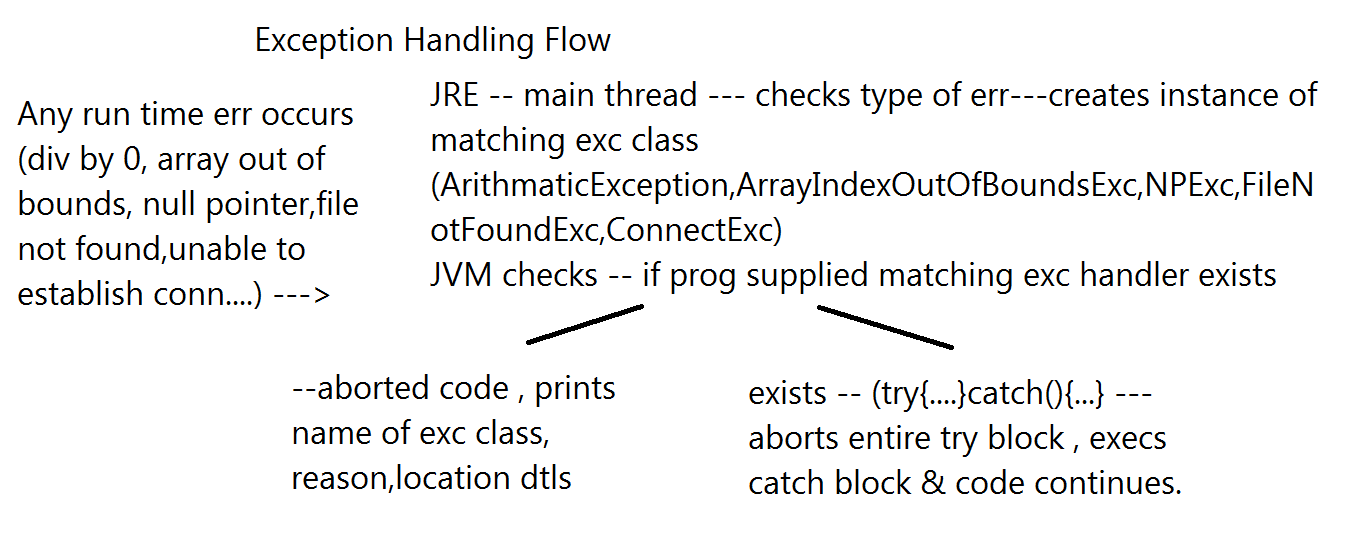
}

}

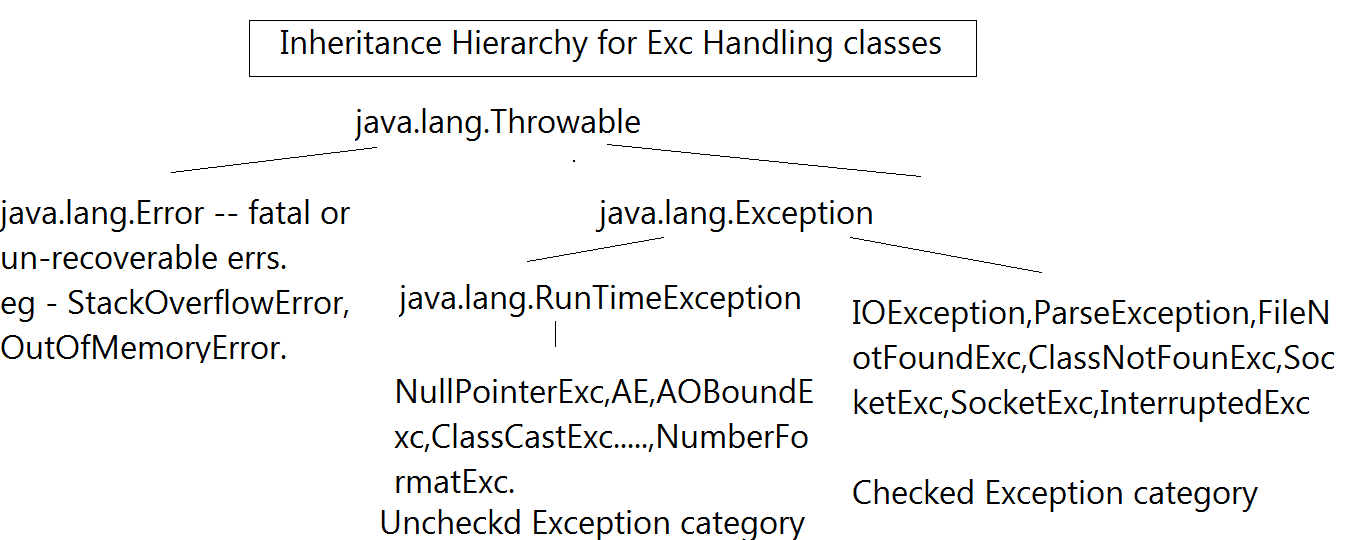
Solve eg in covariance demo.

**Exception Handling**

**exc-handling-flow**

****

**exc-inheritance-hierarchy**

****

**Read me Exception Handling**

Exception Handling

Regarding Exception Handling in java.....

Any run time err occurs(eg file not found,accessing out of array size,accessing func from null ref, divide by 0)

---JRE(main thrd) --- creates matching exc class instance(java.io.FileNotFoundException,java.lang.ArrayOutOfBoundsExc,NullPointerExc,ArithmeticExc)

--- JRE checks -- if prog has proivided exc handling code ?

--- NO -- JRE aborts java code(by supplying def handler) & prints details --F.Q exc class name,reason behind failure & location details(err stack trace

--- YES (try---catch) JRE execs exc handling block & continues with the rest of the code.

syntax(key words) --- try,catch,finally,throw,throws

Inheritance hierarchy of exc classes

unchecked vs checked excs.

Creating custom excs

JDK 1.7 syntax --- try-with-resources(in I/O or device prog)

Checked & Unchked exc are detected or occur only in run-time

JRE DOES NOT distinguish between them

Compiler(javac) differentiates bet them

Javac forces handling of the checked exc. upon the prog.(Handling by supplying matching try-catch block or including it in the throws clause.)

Legal syntax

1. try {...} catch (exc1 e){}

2. try {...} catch (exc1 e){} catch (exc2 e) {} ....

3. try {...} catch (exc1 e){} catch (exc2 e) {}catch(Exception e){catch-all}

3.5 3. try {...} catch (exc1 e){} catch (exc2 | exc3 e) {}catch(Exception e){catch-all}

4. throws syntax ---

method declaration throws comma separated list of exc classes.

eg : Integer class API

public static int parseInt(String s) throws NFE

FileReader API

public FileReader(String fileName) throws FNFE

throws --- meant for javac

Meaning -- Method MAY raise specified exc.

Current method is NOT handling it , BUT its caller should handle.

Mandatory--- only in case of un handled chked excs.

4.5 Exception class API

0. Exception(String mesg)

1. public String toString() -- rets Name of exc class & reason.

2. public String getMessage() -- rets error mesg of exception

3. public void printStackTrace() --- Displays name of exc class, reason, location dtls.

5. finally --- keyword in exc handling

finally -- block -- finally block ALWAYS survives(except System.exit(0))

i.e in the presence or absence of excs.

5.1 try{...} catch (Exception e){....} finally {....}

5.2 try{...} catch (NullPointerException e){....} finally {....}

5.3 try {...} finally {....}

Creating Custom Exc(User defined exception or application exc)

1. Create a pkged public class which extends Throwable(not reco but legal)/Exception(recommended)/Error(not reco but legal)/RuntimeExc(not reco but legal)

2.CustExc(String msg) : overload the constr : to invoke the super-class constr.

of the form

Exception (String msg)

OR

CustExc(String msg,Throwable rootCause)

public Exception(String message,Throwable cause)

Objective :

Check the speed of vehicle on a freeway

Accept the speed using Scanner : can be speed too low(exc) or too high(exc) or in range

keyword -- throw --for throwing user define exc.

syntax :

throw Throwable instance;

throw new NullPointerExc();

throw new InterruptedExc();

throw new Throwable("abc");

throw new Account(...);//javac err

throw new AccountOverdrawnExc("funds too low...");

**Day7**

**Enums**

What is enum in java ?

Enumerations (in general) are generally a set of related constants.

They have been in other programming languages like C++ from beginning.

Supported in Java since JDK 1.5 release.

Enumeration in java is supported by keyword enum. enums are a special type of class that always extends java.lang.Enum.

A simple usage will look like this:

public enum DIRECTION {

EAST,

WEST,

NORTH,

SOUTH //optionally can end with ";"

}

Here EAST, WEST, NORTH and SOUTH are implicitely of type

public final static Direction EAST;

Super class of all enums

public abstract class Enum<E extends Enum<E>>

extends Object

implements Comparable<E>, Serializable

ie. they are comparable and serializable implicitly.

All enum types in java are singleton by default.

So, you can compare enum types using ‘==’ operator also.

Since enums extends java.lang.Enum, so they can not extend any other class because java does not support multiple inheritance . But, enums can implement any number of interfaces.

enum can be declared within a class or separately.

eg of enum within a class

When declared inside a class, enums are always static by default

eg public class TestOuter

{

enum Direction

{

EAST,

WEST,

NORTH,

SOUTH

}

}

To access a direction -- use TestOuter.Direction.NORTH.

Constructors of enum

By default, enums do not require you to give constructor definitions & javac implicitely calls super class constructor

Enum(String name,int ordinal)

Methods of Enum

Enum[] values() --rets array of enum type of refs.--pointing to singleton objs

Enum valueOf(String name) -- string to enum type converter

values & valueOf methods generated by compiler --so not part of javadocs.

If u pass a different name (eg -- ABC) to valueOf ---throws IllegalArgumentException

String name() --rets name of constant in string form

int ordinal() --rets index of the const as it appears in enum.--starts with 0

You can supply your own constructor/s to initialize the state(data member of enum types.

enum Direction {

// Enum types

EAST(0), WEST(180), NORTH(90), SOUTH(270);

// Constructor

private Direction(final int angle) {

this.angle = angle;

}

// Internal state

private int angle;

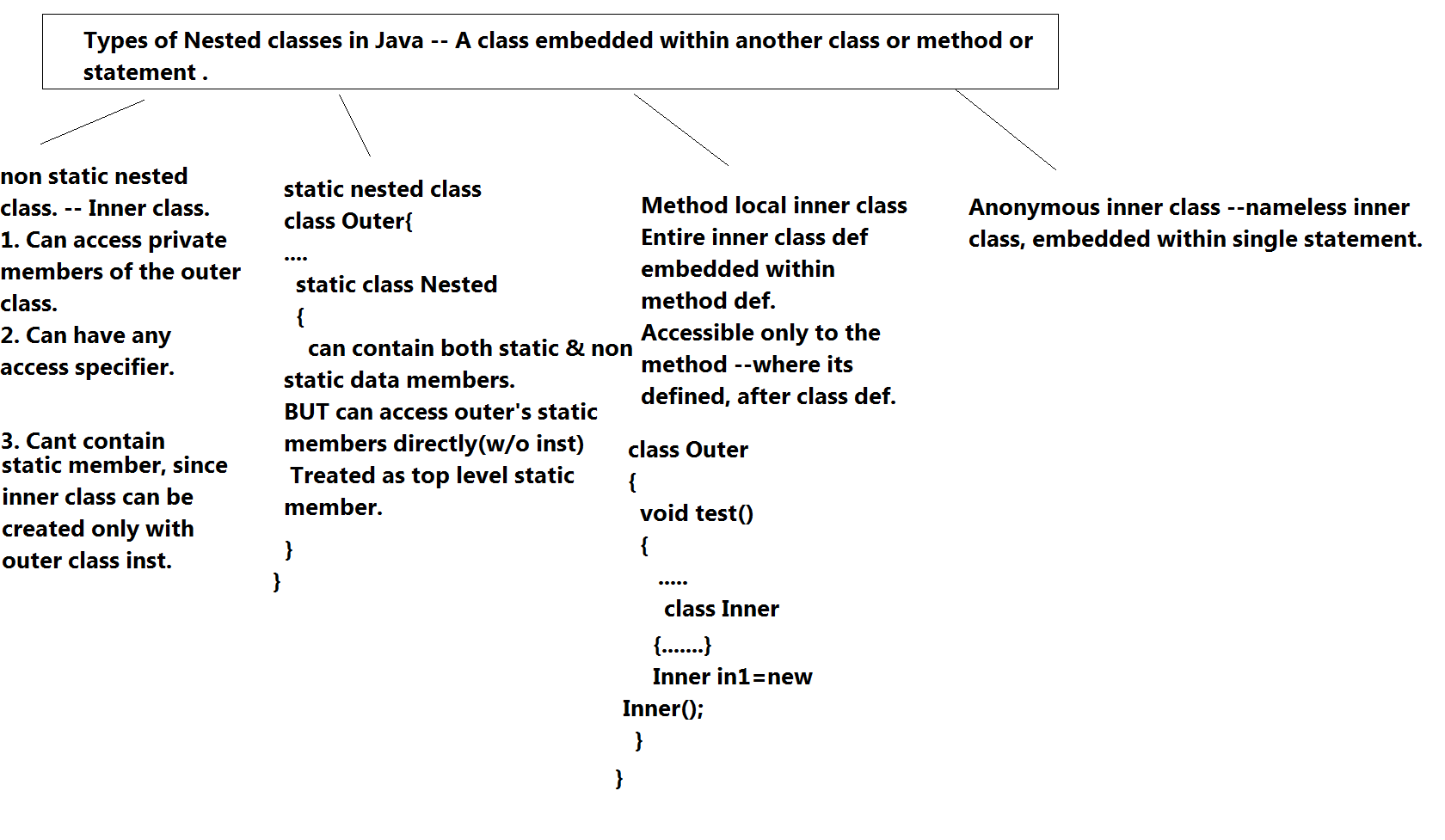
public int getAngle() {

return angle;

}

}

**Nested classes**

****

**Nested classes Read me**

1. The inner class(non-static nested) has access to all of the outer class's members, including those marked private , directly(without inst.)

2. To instantiate an inner class, you must have a reference to an instance of the outer class.

syntax :

Instantiating a non-static nested class requires using both the outer inst and nested class names as follows:

BigOuter.Nested n = new BigOuter().new Nested();

3. Such Inner classes can't have static members.(From JDK 1.8 onwards can have --static final members)

About method-local inner classes

1.A method-local inner class is defined within a method of the enclosing class.

2.For the inner class to be used, you must instantiate it, and that instantiation must happen within the same method, but after the class definition code.

3. A method-local inner class cannot use variables declared within the method

(including parameters) unless those variables are marked final or effectively final.

static nested classes

1.A static nested class is not an inner class, it's a top-level nested class.

2. You don't need an instance of the outer class to instantiate a static nested class.

4.It cannot access non-static members of the outer class directly BUT can access static members of the outer class.

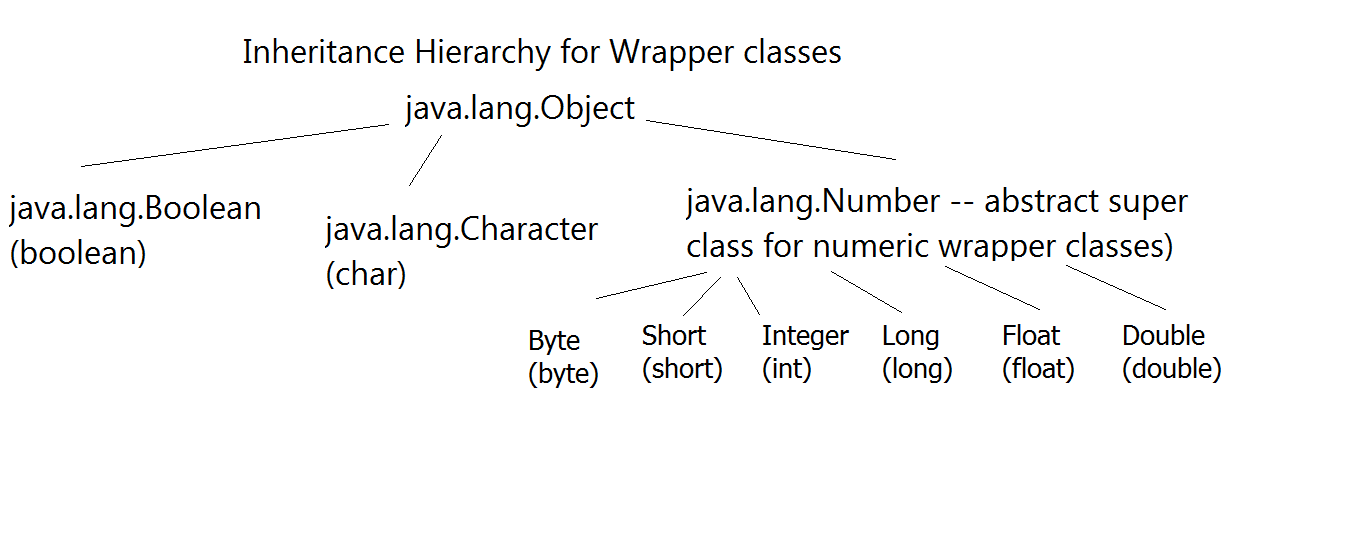
5. It can contain both static & non-static members.

6. JVM will not load any class's static init block -- until u actually refer to something from that class.

(Lazy loading) This is true for static nested classes too.

**Wrappers**

**Wrapper-class-hierarchy**



**Read me wrapper**

Regarding wrapper classes

1. What's need of wrapper classes?

---1. to be able to add prim types to growable collection(growable data structure eg -- LinkedList)

--- 2. wrapper classes contain useful api(eg --- parseInt,parseFloat....)

2. What are wrappers? --- Class equivalent for primitive types

-- Inheritance hierarchy

java.lang.Object --- Character (char)

java.lang.Object --- Boolean

Object -- Number -- Byte,Short,Integer,Long,Float,Double

3. Constrs & methods --- for boxing & unboxing

boxing= conversion from prim type to the wrapper type(class type)

un-boxing = conversion from wrapper type to the prim type

eg

Integer(int data) --- boxing

Integer i1=new Integer(100);

//un-boxing

int data=i1.intValue();

Integer i1=100;//no err from JDK 1.5

sop(i1);

int data=1234;

i1++;//Integer--->int(auto unboxing), inc ,auto box

Object o=123.45;//auto-boxing(double--->Double)--up casted to Object

Number n1=true;//auto-box----X(up casted) to Number

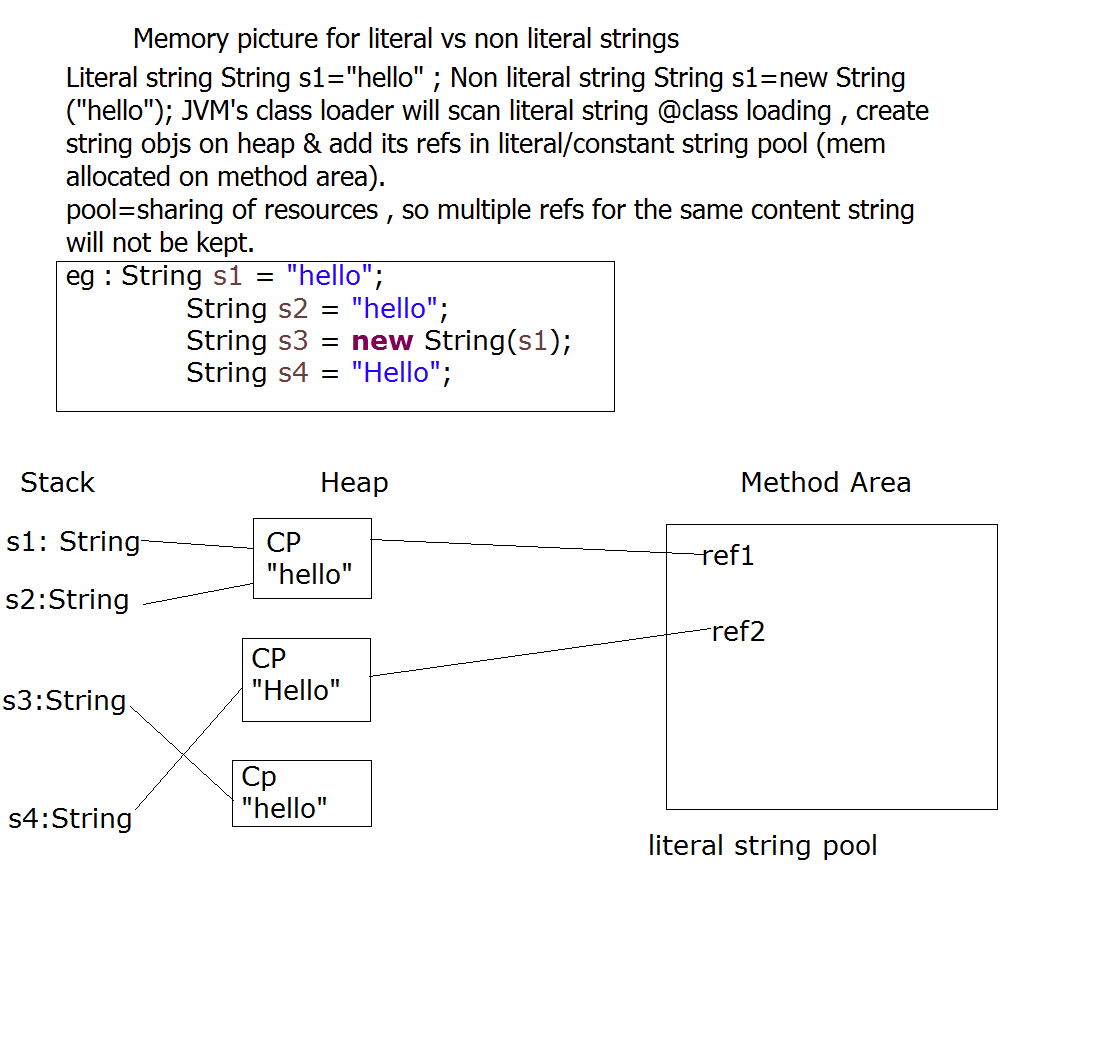
Object o2=false;//auto box -- up casting

Double d1=1234;//auto boxing (int --->Integer) ---X--Double

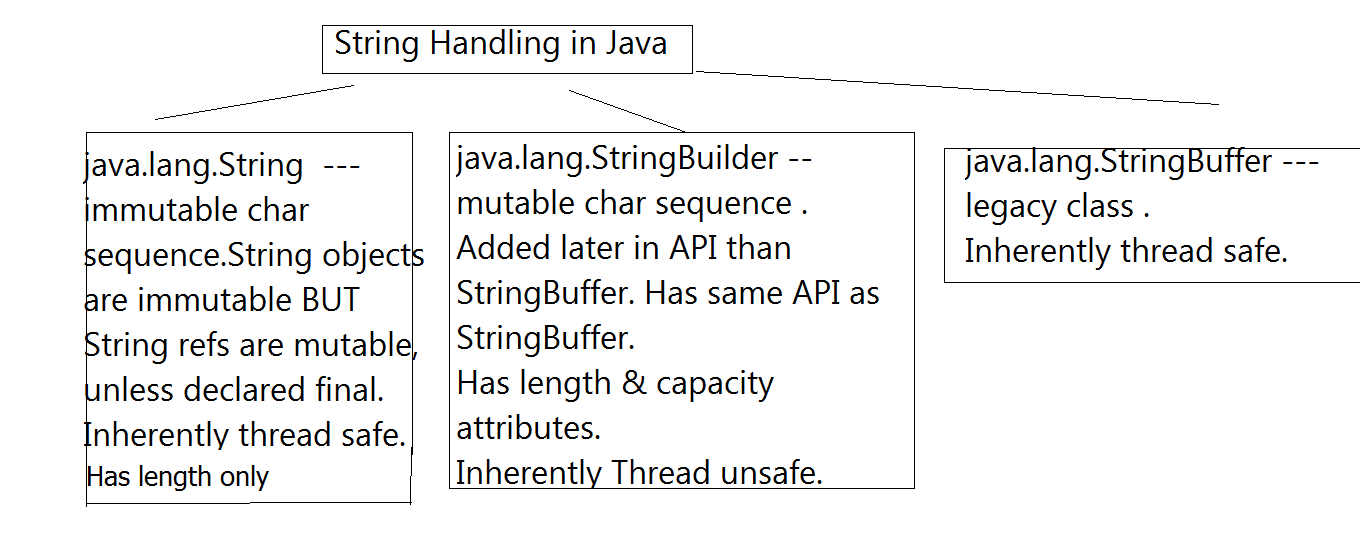
4. JDK 1.5 onwards --- boxing &unboxing performed automatically by java compiler,when required. --- auto-boxing , auto-unboxing,

5. examples

**literal string pool**



**String Overview**



**Read me String var-args**

String class API

Important String class constructors

1.String(byte[] bytes) --- byte[] ----> String converter

2.String(char[] chars) --- char[] ---> String converter

3.String (byte[] bytes,int offset,int len) ---byte[] ----> String converter from the specified offset , specified len no of bytes will be converted.

eg . String s=new String(bytes,3,4); --- String will contain bytes[3]----bytes[6]

4. String(char[] ch,int offset,int len)

5. String(String s)

String class methods --- to go through

charAt,compareTo,contains,copyValueOf,format,valueOf,getBytes,toCharArray,toLowerCase,indexOf,lastIndexOf,split,replace,startsWith,endsWith,length,intern

1.

boolean equals(Object o) ---- ret true iff 2 strings are having same contents (case sensitive)

About equals()

super class def. --- java.lang.Object

public boolean equals(Object o)

Rets true iff both refs(this & o) are equal i.e referring to the same object.

Sub-class developers MUST override equals for content-wise(depending on Object's state) comparison.

2. concat,charAt,indexOf,lastIndexOf,toUpperCase,toLowerCase,format,split

printf & Formatter class

Refer to java.util.Formatter class for formatting conversion details.

Imp ---

Formatting details

%c -- character

%b -- boolean

%h -- hex value of hashcode of obj ref.

%s -- string

%d -- int

%f,%g -- float/double

%x -- hex value

%n -- line separator

%tD -- Date

%tT -- Time

%tc -- Time stamp(date & Time)

%td-%1$tb-%1$tY -- can be applied to GC or Date.

Date/Time Handling in Java

API

1. java.util.Date--- represents system date.

Constructor

1.Date() --- creates Date class instance representing system date.

2.Date(long msec) --- creates Date class instance representing date for msec elapsed after epoch(=1st Jan 1970)

2. java.util.GregorianCalendar

month range --- 0-11

GregorianCalendar(int yr,int mon,int date);

GregorianCalendar(int yr,int mon,int date,int hr,int min,int sec);

3. Date/Time formatting via printf

%tc -- for complete timestamp(date & time)

%tD -- for date

%tT -- time

Arguments --- Date, GregorianCalendar

static import syntax ---

eg -- import static java.util.Calendar.\*;

or import static java.lang.System.\*;

in such src - u can access directly static members of Calendar class or from 2nd statement u can directly use out.println("testing static imports!");

var-args

variable args syntax.--- Must be last arg in the method args.

Can use primitive type or ref types.

Legal ---

void doStuff(int... x) {

} // expects from 0 to many ints

Usage : ref.doStuff();

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

ref.doStuff(ints);

ref.doStuff(20,34,56);

System.out.printf("%n");

// as parameters

void doStuff2(char c, int... x) { } // expects first a char,

// then 0 to many ints

void doStuff3(Animal... animal) { } // 0 to many Animals

invocations ---

ref.doStuff3();

ref.doStuff3( animals);

ref.doStuff3( a1,a2,a3);

Illegal:

void doStuff4(int x...) { } // bad syntax

void doStuff5(int... x, char... y) { } // too many var-args

void doStuff6(String... s, byte b) { } // var-arg must be last

**Regarding Generics**

Generic syntax ---

Available from Java SE 5 onwards.

Represents Parameterized Types.

Can Create Generic classes, interfaces, methods and constructors.

Operates on Parameterized data Types

In Pre-generics world , similar achieved via Object class reference.

Advantages

Adds Type Safety to the code @ compile time

Can add type safe code where type-mismatch errors(i.e ClassCastExceptions) are caught at compile time.

No need of explicit type casting, as all casts are automatic and implicit.

A generic class means that the class declaration includes a type parameter.

eg --- class MyGeneric<T>

T ---type --- ref type

**Day 8**

**Collection Framework**

**ArrayList\_Vs\_LinkedList**

Difference between ArrayList and LinkedList in Java

ArrayList and LinkedList both implements List interface and their methods and results are almost identical. However there are few differences between them which make one better over another depending on the requirement.

ArrayList Vs LinkedList

1) Search: ArrayList search operation is pretty fast compared to the LinkedList search operation. get(int index) in ArrayList gives the performance of O(1) while LinkedList performance is O(n).

Reason: ArrayList maintains index based system for its elements as it uses array data structure implicitly which makes it faster for searching an element in the list. On the other side LinkedList implements doubly linked list which requires the traversal through all the elements for searching an element.

2) Deletion: LinkedList remove operation gives O(1) performance while ArrayList gives variable performance: O(n) in worst case (while removing last element) and O(1) in best case (While removing 1st element).

Conclusion: LinkedList element deletion is faster compared to ArrayList.

Reason: LinkedList’s each element maintains two pointers (addresses) which points to the both neighbor elements in the list. Hence removal only requires change in the pointer location in the two neighbor nodes (elements) of the node which is going to be removed. While In ArrayList all the elements need to be shifted to fill out the space created by removed element.

3) Inserts Performance: LinkedList add method gives O(1) performance while ArrayList gives O(n) in worst case. Reason is same as explained for remove.

4) Memory Overhead: ArrayList maintains indexes and element data while LinkedList maintains element data and two pointers for neighbor nodes hence the memory consumption is high in LinkedList comparatively.

There are few similarities between these classes which are as follows:

Both ArrayList and LinkedList are implementation of List interface.

They both maintain the elements insertion order which means while displaying ArrayList and LinkedList elements the result set would be having the same order in which the elements got inserted into the List.

Both these classes are non-synchronized and can be made synchronized explicitly by using Collections.synchronizedList method.

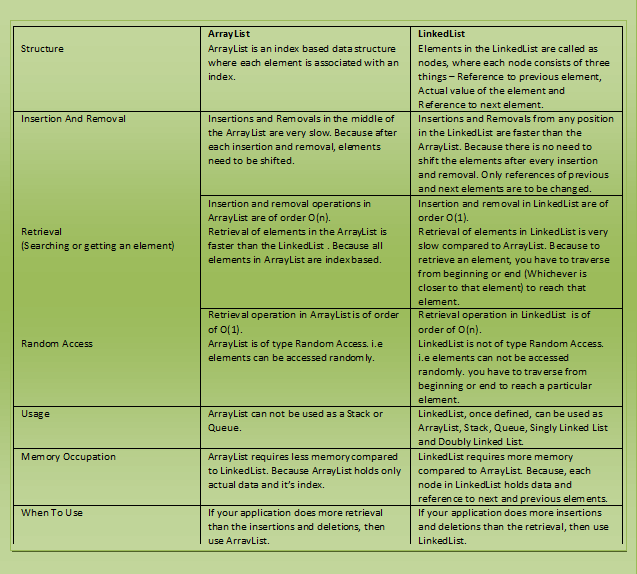
The iterator and listIterator returned by these classes are fail-fast (if list is structurally modified at any time after the iterator is created, in any way except through the iterator’s own remove or add methods, the iterator will throw a ConcurrentModificationException).

When to use LinkedList and when to use ArrayList?

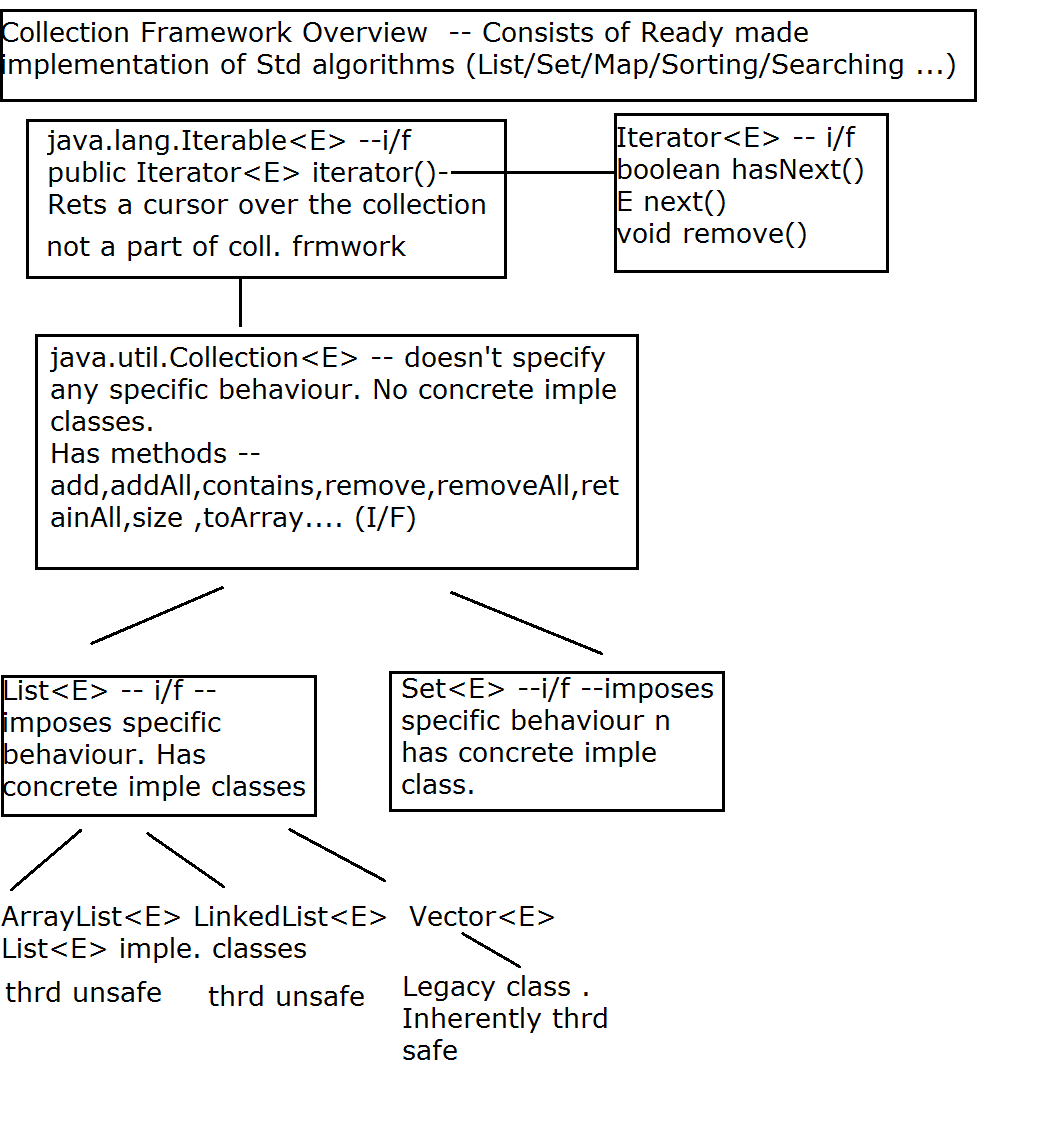
1) As explained above the insert and remove operations give good performance (O(1)) in LinkedList compared to ArrayList(O(n)). Hence if there is a requirement of frequent addition and deletion in application then LinkedList is a best choice.

2) Search (get(index) method) operations are fast in Arraylist (O(1)) but not in LinkedList (O(n)) so If there are less add and remove operations and more search operations requirement, ArrayList would be your best bet.

**ArrayListVsLinkedList**



**collection frmwork overview**

****

**Collection readme**

List<E> features

1. List represents ordered collection --- order is significant

2. Allows null references

3. Allows duplicates

4. Supports index based operation

java.util.ArrayList<E> -- E -- type of ref.

1. ArrayList<E> -- constructor

API

ArrayList() -- default constructor. -- creates EMPTY array list object , with init capacity=10,size=0;

eg ---ArrayList<Integer> l1=new ArrayList<Integer>();

1.5 1. ArrayList<E> -- constructor

API

ArrayList(int capacity) -- -- creates EMPTY array list object , with init capacity=capacity,size=0;

eg ---ArrayList<Integer> l1=new ArrayList<>(100);

2. add methods

boolean add(E e) --- append

void add(int index,E e) --- insert

void addAll(Collection<E> e) -- bulk append operation

eg : l1 --- AL<Emp>

l1.addAll(.....);

AL,LL,Vector --- legal

HS,TS,LHS --legal

HM,LHM,TM --illegal --javac error

2.5 Retrieve elem from list

E get(int index)

index ranges from ---0 ---(size-1)

java.lang.IndexOutOfBoundsException

3. display list contents using --- toString

4. Attaching Iterator

Collection<E> interface method -- implemented by ArrayList

Iterator<E> iterator()

---places iterator BEFORE 1st element ref.

Iterator<E> i/f methods

boolean hasNext() -- rets true if there exists next element, false otherwise.

E next() --- returns the element next to iterator position

void remove() -- removes last returned element from iterator.

Limitation --- type forward only & can start from 1st elem only.

Regarding exceptions with Iterator/List

1. java.util.NoSuchElementException -- thrown whenever trying to access the elem beyond the size of list.

2. java.lang.IllegalStateException --- thrown whenever trying to remove elem before calling next().

3. java.util.ConcurrentModificationException-- thrown typically --- when trying to use same iterator/list iterator --after structrually modifying list(eg add/remove methods of list)

Exception while accessing element by index.

4. java.lang.IndexOutOfBounds -- thrown typically -- while trying to access elem beyond size(0---size-1)

6. Attaching for-each = attaching implicit iterator.

Attaching ListIterator ---scrollable iterator or to beign iteration from a specific element -- List ONLY or list specific iterator.

ListIterator<E> listItearator() --places LI before 1st element

ListIterator<E> listItearator(int index) --places LI before specified index.

4. search for a particular element in list

boolean contains(Object o)

5. searching for 1st occurrence

use -- indexOf

int indexOf(Object o)

rets index of 1st occurrence of specified elem. Rets -1 if elem not found.

searching for last occurrence

use -- lastIndexOf

int lastIndexOf(Object o)

rets index of last occurrence of specified elem. Rets -1 if elem not found.

5.5

E set(int index,E e)

Replaces old elem at spepcified index by new elem.

Returns old elem

6. remove methods

E remove(int index) ---removes elem at specified index & returns removed elem.

boolean remove(Object o) --- removes element specified by argument , rets true -- if elem is removed or false if elem cant be removed.

Objectives in Integer list

0. Create ArrayList of integers & populate it.

1. check if element exists in the list.

2. disp index of 1st occurance of the elem

3. double values in the list --if elem val > 20

4. remove elem at the specified index

5. remove by elem. -- rets true /false.

NOTE :

For searching or removing in List Implementation classes --- All search methods (contains,indexOf,lastIndexOf,remove(Object o)) -- based upon equals method(of type of List eg --Account/Customer/Emp....)

For correct working

1. Identify prim key & create overloaded constr using PK.

2. Using PK , override equals for content equality.

Objective --- Create simple List(ArrayList) of Account & test complete API

1.1

Create Empty Arraylist of Accounts

1.2 Accept a/c info from user till user types "stop" & populate AL.

1.2.1 -- Display AL content using for-each

1.3 Accept account id & display summary or error mesg

1.4 Accept src id , dest id & funds transfer.

1.5 Accept acct id & remove a/c --

1.6 Apply interest on all saving a/cs

1.7 Sort accounts as per asc a/c ids.

1.8 Sort accounts as per desc a/c ids.

1.9 Sort a/cs as per creation date -- w/o touching UDT

2.0 Sort a/cs as per bal

Sorting --- For sorting elements as per Natural(implicit i.e criteria defined within UDT class definition) ordering or Custom(explicit i.e criteria defined outside UDT , in a separate class or anonymus iner class)

Steps for Natural ordering

Natural Ordering is specified in generic i/f

java.lang.Comparable<T>

T -- UDT , class type of the object to be compared.

eg -- Emp,Account , Customer

I/f method

int compareTo(T o)

Steps

1. UDT must implement Comparable<T>

eg : public class Account implements Comparable<Account>

2. Must override method

public int compareTo(T o)

{

use sorting criteria to ret

< 0 if this < o,

=0 if this = o

> 0 if this > o

}

3.Use java.util.Collections class API

Method

public static void sort(List<T> l1)

l1 -- List of type T.

sort method internally invokes compareTo method(prog supplied) of UDT & using advanced sorting algorithm , sort the list elems.

Limitation of natural Ordering

Can supply only 1 criteria at given time & that too is embedded within UDT class definition

Typically use -- Natural ordering in consistence with equals method.

Alternative is Custom Ordering(external ordering)

I/f used is --- java.util.Comparator<T>

T -- type of object to be compared.

Steps

1. Create a separate class (eg. AccountBalComparator) which implements Comparator<T>

eg

public class AccountBalComparator implements Comparator<Account>

2.Implement(override) i/f method -- to supply comparison criteria.

int compare(T o1,T o2)

Must return

< 0 if o1<o2

=0 if o1=o2

> 0 if o1 > o2

3. Invoke Collections class method for actual sorting.

public static void sort(List<T> l1,Comparator<T> c)

parameters

l1 --- List to be sorted(since List is i/f --- any of its implementation class inst. can be passed)

c - instance of the class which has implemented compare method.(or implemented Comparator)

Internally sort method invokes compare method from the supplied Comparator class instance.

More on generic syntax

Constructor of ArrayList(Collection<? extends E> c)

? -- wild card in generic syntax (denotes any unknown type)

extends -- keyword in generics, to specify upper bound

? extends E -- E or sub type

Complete meaning --- Can create new populated ArrayList of type E , from ANY Collection(ArrayList,LinkedList,Vector,HashSet,LinkedhashSet,TreeSet) of type E or its sub type.

ArrayList<Emp> l1=new ArrayList<>();

l1.add(new Emp(1,"aa",1000);

l1.add(new Emp(2,"ab",2000);

ArrayList<Emp> l2=new ArrayList<>(l1);

sop(l2.size());

----------

HashSet<Emp> hs=new HashSet<>();

hs.add(new Emp(1,"aa",1000);

hs.add(new Emp(2,"ab",2000);

l2=new ArrayList<>(hs);

----

Vector<Mgr> v1=new Vector<>();

v1.add(new Mgr(....));

v1.add(new Mgr(....));

l2=new ArrayList<Mgr>(v1);

Map API

HashMap<K,V> --

1. un-sorted(not sorted as per Natural ordering or custom ordering based criteria) & un-ordered(doesn't remember order of insertion) map implementation class.

2. No duplicate keys.

3. Guarantees constant time performance --- via 2 attributes --initial cpacity & load factor.

4. Allows null key reference(once).

5. Inherently thrd unsafe.

HashMap constrs

1. HashMap<K,V>() --- creates empty map , init capa = 16 & load factor .75

2. HashMap<K,V>(int capa) --- creates empty map , init capa specified & load factor .75

3.HashMap<K,V>(int capa,float loadFactor) --- creates empty map , init capa & load factor specified

4. HashMap constrcutor for creating populated map

HashMap(Map <? extends K,? extends V> m)

? -- wild card in generics, represents unknown type

extends -- represents upper bound

? extends K --- K or its sub type

? extends V -- V or its sub type.

Complete meaning -- Creates populated HM<K,V> from ANY map(ie. any Map imple class)

of type K or its sub type & V or its sub type.

eg : Suppose Emp <---- Mgr

HM<Integer,Emp> hm=new HM<>();

hm.put(1,e1);

hm.put(2,m1);

HM<Integer,Emp> hm2=new HM<>(hm);

sop(hm2);

LHM<Integer,Emp> lhm=new LHM<>(hm);//legal

HM<Integer,Mgr> hm3=new HM<Integer,Emp>(hm);//javac error

TM<Integer,Mgr> hm4=new TM<>();

hm4.put........

HM<Integer,Emp> hm5=new HM<>(hm4);

HM(Map<? extends K,? extends V>map)

put,get,size,isEmpty,containsKey,containValue,remove

Objective : Create AccountMap

Identify key & value type

create empty unsorted map(HashMap<K,V>) & populate the same

Disp all entries of HM ---can use only toString

1.get acct summary --- i/p --id o/p --- err / dtls

2.Withdraw --- specify Account id & Amt ---- o/p : update acct dtls if acct exists o.w err msg or exc

3.funds transfer ---

i/p sid,dest id, amt

4.remove --- account

i/p id

5.Apply interest on on saving type of a/cs.

or

display all accts created after date.

Attach for-each to map & observe.

Sort the map as per : asc order of accts Ids.

Sort the map as per : desc order of accts Ids

Sort the accts as per : balance

If map sorting involves key based sorting criteria --- can be sorted by converting into TreeMap

Constructors of TreeMap

1. TreeMap() -- Creates empty map , based upon natural ordering of keys

2. TreeMap(Map<? extends K,? extends V> map)

Creates populated map , based upon natural ordering of keys

3. TreeMap(Comparator<? super K> c)

Regarding generic syntax & its usage in TreeMap constructor.

<? super K>

? --- wild card --- any unknown type

super --- gives lower bound

K --- key type

? super K --- Any type which is either K or its super type.

TreeMap(Comparator<? super K> c) --- creates new empty TreeMap, which will sort its element as per custom ordering(i.e will invoke compare(...) of Key type )

<? extends K>

? -- any type or wild card

extends -- specifies upper bound

K -- key type

? extends K --- Any type as Key type or its sub type.

same meaning for <? extends V>

TreeMap(Map<? extends K,? extends V> m)

disp acct ids of all accounts ---impossible directly....(will be done by Collection view of map @ the end)

Apply interest to all saving type a/cs

difficult directly ---so get a collection view of the map & sort the same.

Limitations on Maps

1. Maps can be sorted as per key's criteria alone.

2. can't attach iterators/for-each/for

3 Maps can be searched as per key's criteria alone.

To fix --- get a collection view of a map (i.e convert map to collection)

API of Map i/f

1. To get set of keys asso. with a Map

Set<K> keySet();

2. To get collection of values from a map

Collection<V> values();

3. To get set of Entries(key & val pair) ---

entrySet

Set<Map.Entry> entrySet()

Methods of Map.Entry

K getKey()

V getValue()

7. conversion from collection to array

Object[] toArray() -- non generic version --- rets array of objects

T[] toArray(T[] type)

T = type of collection .

Rets array of actual type.

8. sorting lists --- Natural ordering creiteria

Using java.util.Collections --- collection utility class.

static void sort(List<E> l1) ---sorts specified list as per natural sorting criteria.

**Day 11**

**Read me HashMap**

How HashMap internally works in Java

Hash Map is one of the most used collection. It doesn't extend from Collection i/f.

BUT collection view of a map can be obtained using keySet,values or entrySet()

Internal Implementation

HashMap works on the principal of hashing.

Map.Entry interface -static nested interface.

This interface represents a map entry (key-value pair).

HashMap in Java stores both key and value object ref , in bucket, as an object of Entry class which implements this nested interface Map.Entry.

hashCode() -HashMap provides put(key, value) for storing and get(key) method for retrieving Values from HashMap.

When put() method is used to store (Key, Value) pair, HashMap implementation calls hashcode on Key object to calculate a hash that is used to find a bucket where Entry object will be stored.

When get() method is used to retrieve value, again key object is used to calculate a hash which is used then to find a bucket where that particular key is stored.

equals() - equals() method is used to compare objects for equality. In case of HashMap key object is used for comparison, also using equals() method Map knows how to handle hashing collision (hashing collision means more than one key having the same hash value, thus assigned to the same bucket. In that case objects are stored in a linked list (growable --singly linked)

Bucket term used here is actually an index of array, that array is called table in HashMap implementation. Thus table[0] is referred as bucket0, table[1] as bucket1 and so on.

HashMap uses equals() method to see if the key is equal to any of the already inserted keys (Recall that there may be more than one entry in the same bucket). Note that, with in a bucket key-value pair entries (Entry objects) are stored in a linked-list . In case hash is same, but equals() returns false (which essentially means more than one key having the same hash or hash collision) Entry objects are stored, with in the same bucket, in a linked-list.

In a nutshell there are three scenarios in case of put() -

Using hashCode() method, hash value will be calculated. Using that hash it will be ascertained, in which bucket particular entry will be stored.

equals() method is used to find if such a key already exists in that bucket, if no then a new node is created with the map entry and stored within the same bucket. A linked-list is used to store those nodes.

If equals() method returns true, which means that the key already exists in the bucket. In that case, the new value will overwrite the old value for the matched key.

How get() methods works internally

As we already know how Entry objects are stored in a bucket and what happens in the case of Hash Collision it is easy to understand what happens when key object is passed in the get method of the HashMap to retrieve a value.

Using the key again hash value will be calculated to determine the bucket where that Entry object is stored, in case there are more than one Entry object with in the same bucket stored as a linked-list equals() method will be used to find out the correct key. As soon as the matching key is found get() method will return the value object stored in the Entry object.

In case of null Key

As we know that HashMap also allows null, though there can only be one null key in HashMap. While storing the Entry object HashMap implementation checks if the key is null, in case key is null, it always map to bucket 0 as hash is not calculated for null keys.

HashMap changes in Java 8

Though HashMap implementation provides constant time performance O(1) for get() and put() method but that is in the ideal case when the Hash function distributes the objects evenly among the buckets.

But the performance may worsen in the case hashCode() used is not proper and there are lots of hash collisions. As we know now that in case of hash collision entry objects are stored as a node in a linked-list and equals() method is used to compare keys. That comparison to find the correct key with in a linked-list is a linear operation so in a worst case scenario the complexity becomes O(n).

To address this issue in Java 8 hash elements use balanced trees instead of linked lists after a certain threshold is reached. Which means HashMap starts with storing Entry objects in linked list but after the number of items in a hash becomes larger than a certain threshold, the hash will change from using a linked list to a balanced tree, this will improve the worst case performance from O(n) to O(log n).

**Read me Hashing**

Set features

1. Guarantees no duplicate elements(ref)

2. Constant time performance(add,remove,size) ensured via hashing algorithm.(init capa , load factor)

3. Doesn't add any new behaviour.(from Collection i/f) BUT modifies meaning of some of the methods.

4. Allows single null ref.

5. Does not guarantee order of iteration over time.(HashSet)

Implementation classes --- HashSet ---un ordered & unsorted set.

TreeSet -- Sorted set as per either Natural Ordering or Custom ordering.

LinkedHashSet ---remembers order of insertion.

HashSet<E>

1. HashSet() --- Creates new empty HS of def init capa=16 & L.F =.75

2. HashSet(int capa)

3. HashSet(int capa,float loadFactor)

4. HashSet(Collection<? extends E> c) --- creates populated HS<E> from ANY Collection(AL,LL,Vector,HS,LHS,TS) of type E or sub type.

Confirm working of TreeSet constructors.

1. TreeSet<E>() --- new empty TS -- While populating --- TS invokes E's compareTo -- Natural ordering.

2. TreeSet<E>(Collection<? extends E> c) -- new populated set --TS invokes E's compareTo -- Natural ordering.

3. TreeSet(Comparator<? super E> comp) -- new empty TS . While populating ---invokes compare(..) of supplied Comparator class(comp)

Objective

Create empty Emp set(HS) & populate it.

Regarding Hashing based Data structures....(eg : HashSet,HashTable,HashMap,LiknkedHasSet...)

Steps for Creating HashSet of User defined Type OR HashMap

1. Business Object class(for HashSet) or Key class in Map world must override : hashCode & equals method both

public int hashCode() --- Object class rets int : which represents internal addr where obj is sitting on the heap(typically -- specific to JVM internals)

public boolean equals(Object ref) -- Object class rets true : iff 2 refs are referring to the same copy.

2. Rule to observe while overriding these methods

If 2 refs are equal via equals method then their hashCode values must be same.

eg : If ref1.equals(ref2) ---> true then ref1.hashCode() = ref2.hashCode()

Converse may not be mandatory.(i.e if ref1.equals(ref2) = false then its not mandatory that ref1.hashCode() != ref2.hashCode() : but recommended for better working of hashing based D.S)

Thumb rule -- Use same members (private data members) for overriding equals & hashCode methods

How does hashing based data structure ensure constant time performance?

If no of entries > capacity \* load factor --- re-hashing takes place ---

New data structure is created --(hashtable) -- with approx double the original capacity --- HM takes all earlier entries from orig map & places them in newly created D.S -- using hashCode & equals. -- ensures lesser hash collisions.

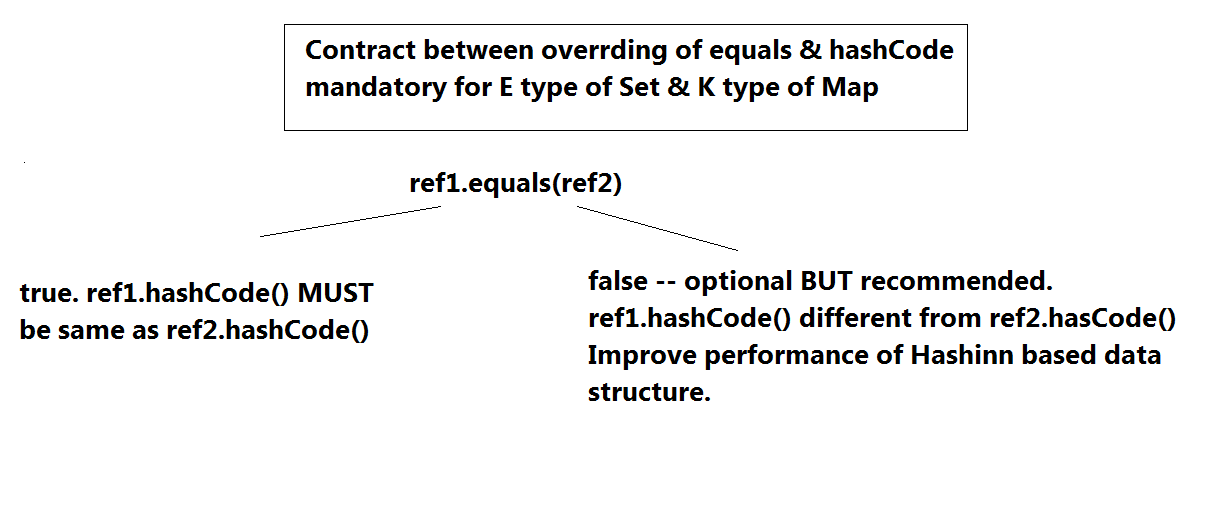
Why there is a guarantee that a duplicate entry / ref can't exist in yet another bucket ?

Answer is thanks to the contract between overriding of hashCode & equals methods

If two keys are the same (equals() returns true when you compare them), their hashCode() method must return the same number. If keys violate this, then keys that are equal might be stored in different buckets, and the hashmap would not be able to find key-value pairs (because it's going to look in the same bucket).

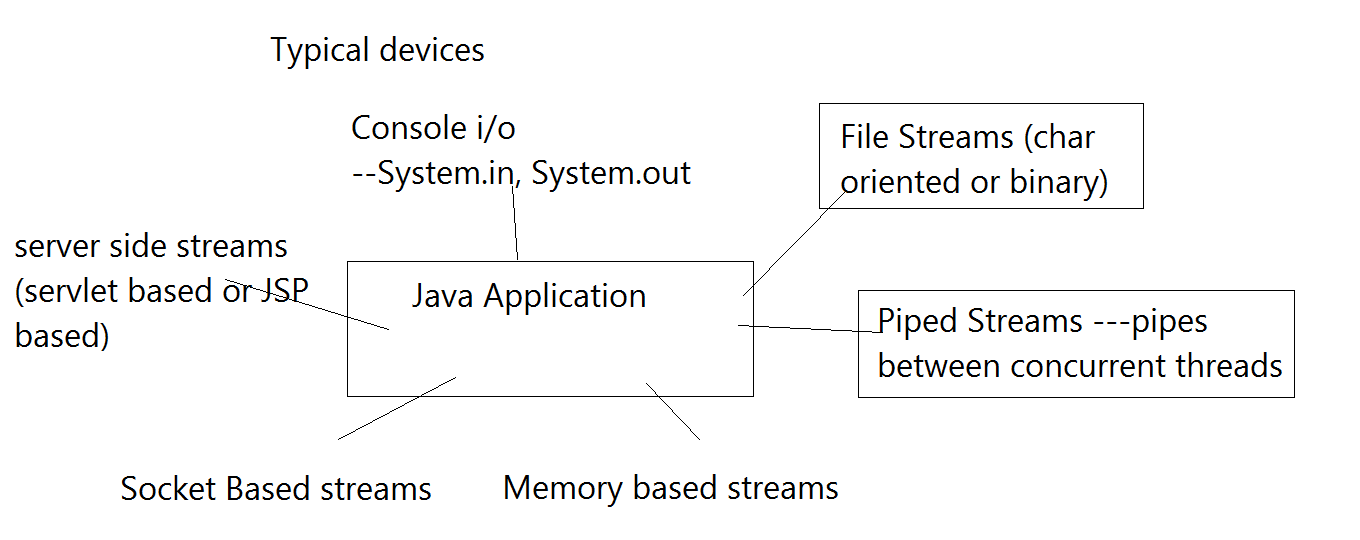
If two keys are different(i.e equals method rets false) , then it doesn't matter if their hash codes are the same or not. They will be stored in the same bucket if their hash codes are the same, and in this case, the hashmap will use equals() to tell them apart.

**Contract**

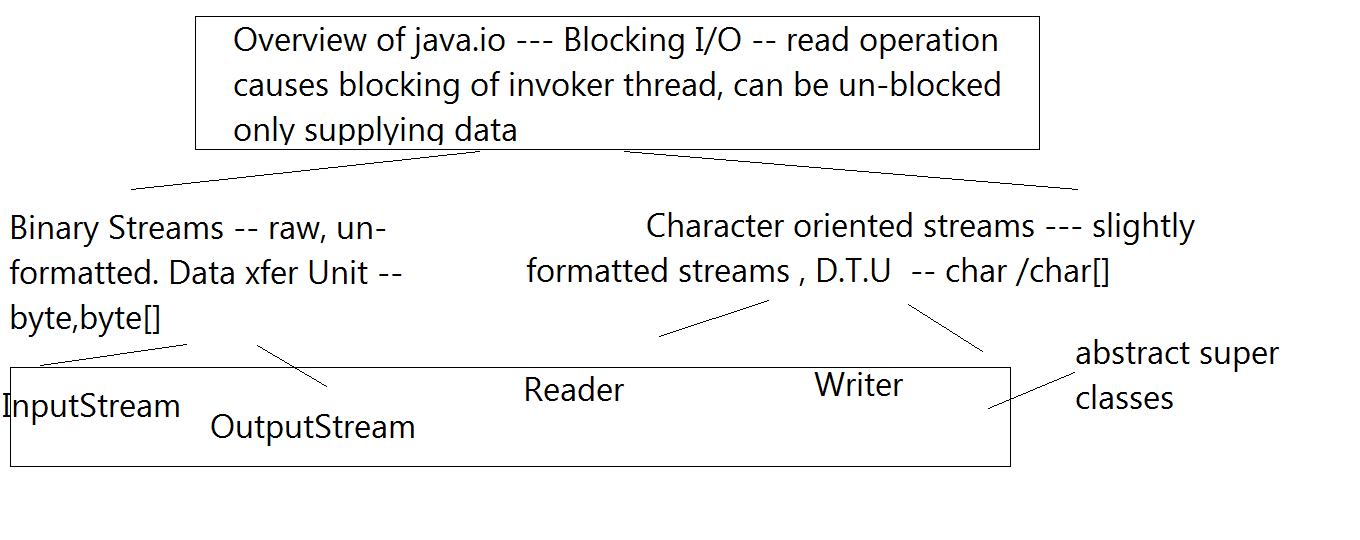
****

**IO handling**

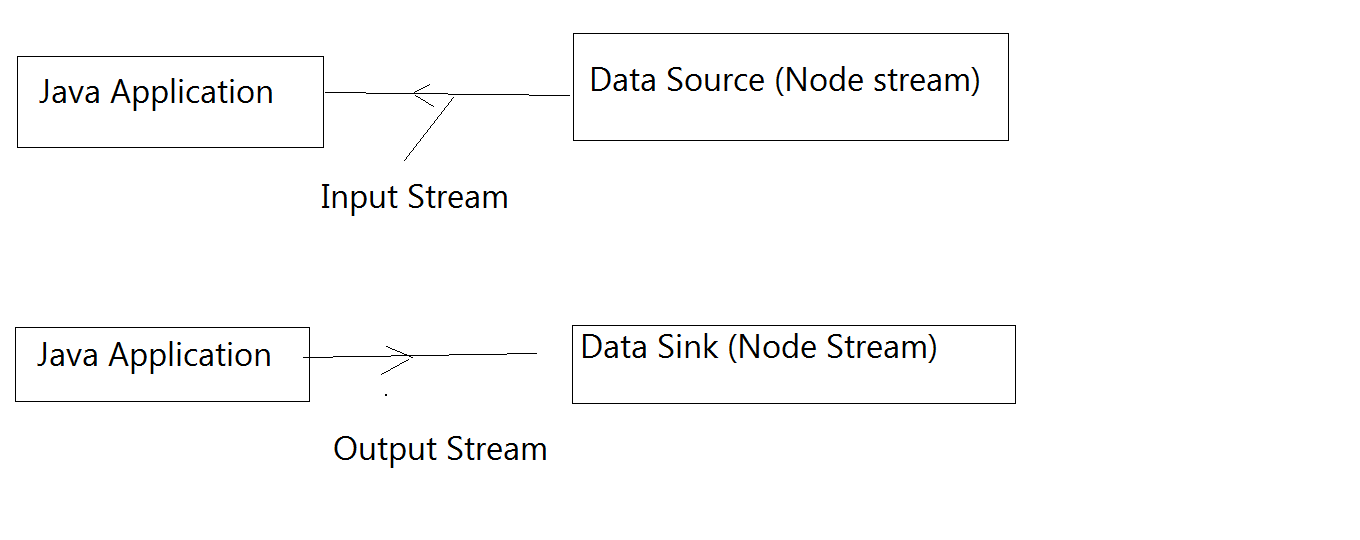
**Devices-overview**

****

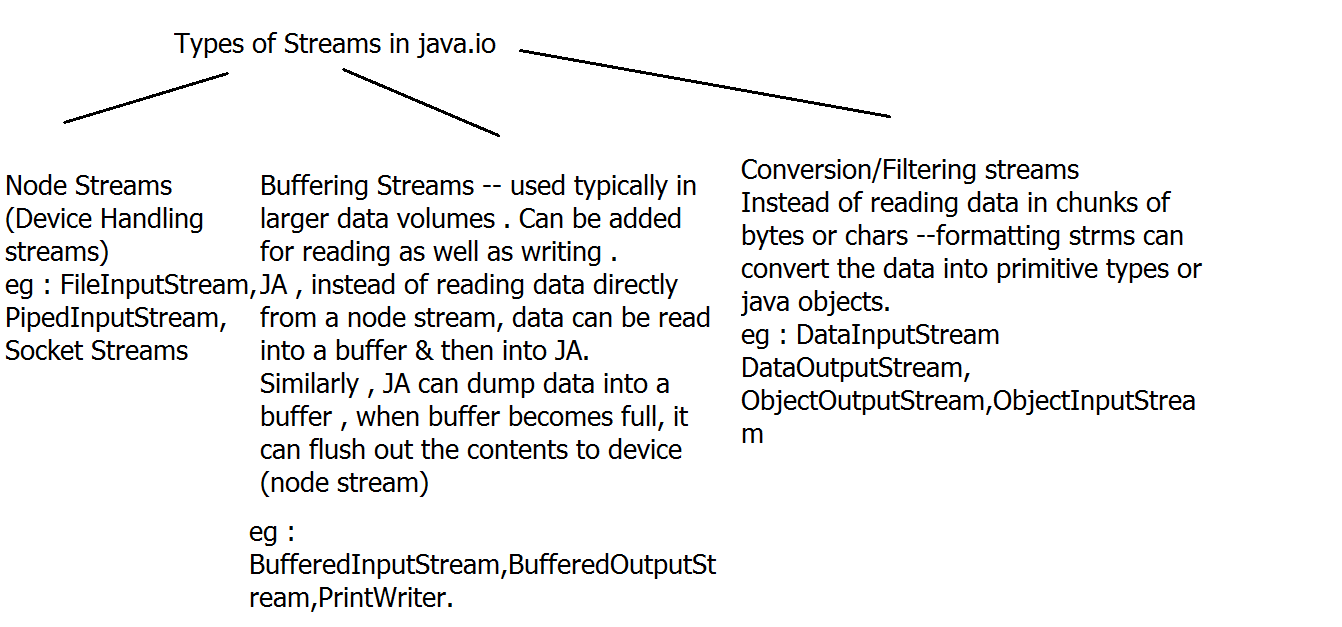
**io-api-overview**

****

**Streams-basics**

****

**Types of streams**

****

**Read me IO**

I/O handling

Desc of FileInputStream --- java.io.FileInputStream

bin i/p stream connected to file device(bin/char) -- to read data.

Desc of FileOutputStream --- java.io.FileOutputStream

bin o/p stream connected to file device(bin/char) -- to write data.

Desc of FileReader--- java.io.FileReader

char i/p stream connected to file device(char) -- to read data.

Desc of FileWriter--- java.io.FileWriter

char o/p stream connected to file device(char) -- to write data.

Objective --- Read data from text file in buffered manner.

1. java.io.FileReader(String fileName) throws FileNotFoundException

--- Stream class to represent unbuffered char data reading from a text file.

Has methods -- to read data using char/char[]

eg -- public int read() throws IOException

public int read(char[] data) throws IOException

Usage eg-- char[] data=new char[100];

int no= fin.read(data);

public int read(char[] data,int offset,int noOfChars) throws IOException

Usage eg-- char[] data=new char[100];

int no= fin.read(data,10,15);

eg -- 12 chars available

no=12;data[10]----data[21]

1.5 FileReader(File f) throws FileNotFoundException

java.io.File -- class represents path to file or a folder.

2. Improved version -- Buffered data read .

For char oriented streams--- java.io.BufferedReader(Reader r)

API of BR ---

String readLine() --- reads data from a buffer in line by line manner-- & rets null at end of Stream condition.

Objective -- Replace JDK 1.6 try-catch-finally BY JDK 1.7 try-with-resources syntax.

Meaning --- From Java SE 7 onwards --- Introduced java.lang.AutoCloseable -- i/f

It represents --- resources that must be closed -- when no longer required.

i/f method

public void close() throws Exception-- closing resources.

java.io --- classes -- have implemented this i/f -- to auto close resource when no longer required.

syntax of try-with-resources

try (//open one or multiple AutoCloseable resources)

{ ......

} catch(Exception e)

{

}

Objective ---To confirm device independence of Java I/O --- replace File device by Console

i.e --- Read data from console i/p --- in buffered manner till 'stop' & echo back it on the console.

required stream classes --- BR(ISR(System.in))

Alternative is --- use Scanner class.

Adv. of Scanner over above chain ----- contains ready-made parsing methods(eg --- nextInt,nextDouble.....)

But Scanner is not Buffered Stream

Can combine both approaches.(new Scanner(br.readLine())

Objective --- Combine scanner & buffered reader api --- to avail buffering + parsing api. ---

BufferedReader provides buffering BUT no simple parsing API. -- supplies br.readLine only

Scanner -- Can be attached to file directly

Constr -- Scanner(File f)

BUT no buffering .

How to use both?

Create BR br=new BR(new FR(...));

while ((s=br.readLine())!=null)

{

//scanner can be attached to string ---Scanner(String s)

Scanner sc=new Scanner(s);

// parse data using Scanner API --next,nextInt,nextBoolean

}

Overloaded constructor of FileReader(File f)

java.io.File ---- class represents path to file / folder

Regarding java.io.File -----

Does not follow stream class hierarchy, extends Object directly.

File class --- represents abstract path which can refer to file or folder.

Usage --- 1. To access/check file/folder attributes(exists,file or folder,read/w/exec permisssions,path,parent folder,create new empty file,create tmp files & delete them auto upon termination,mkdir,mkdirs,rename,move,size,last modified ,if folder---list entries from folder,filter entries)

Constructor ---

File (String path) ---

eg --- File f1=new File("abc.dat");

if (f1.exists() && f1.isFile() && f1.canRead())

...attach FileInputStream or FileReader

File (String path) ---

File class API --- boolean exists(),boolean isFile() , boolean canRead()

Objective --- Text File copy operation --- in buffered manner.

For writing data to text file using Buffered streams

java.io.PrintWriter --- char oriented buffered o/p stream --- which can wrap any device.(Binary o/p stream or Char o/p stream)

Constructors---

PrintWriter(Writer w) --- no auto flushing,no conversion, only buffering

PrintWriter(Writer w, boolean flushOnNewLine)--- automatically flush buffer contents on to the writer stream --upon new line

PrintWriter(OutputStream w) --- can wrap binary o/p stream -- buffering +conversion(char-->binary),no auto-flush option

PrintWriter(OutputStream w , boolean flushOnNewLine) ---

API Methods----print/println/printf same as in PrintStream class(same type as System.out)

Stream class which represents --- Char o/p stream connected to Text file. --- java.io.FileWriter

Constructor

FileWriter(String fileName) throws IOException -- new file will be created & data will be written in char format.

FileWriter(String fileName,boolean append) --- if append is true , data will be appended to existing text file.

-----------------

Collection & I/O

Objective ---

Items Inventory

Item -- code(String-Prim key),desc,category,quantity,price,shipmentDate

constr,toString.

Create suitable collection of Items(HashMap) --- sort map as per desc item code ,& store sorted item dtls in 1 text file .

NOTE : individual item rec MUST be written on separate line.

Sort items as shipment Date & store sorted dtls in another file . Before exiting ensure closing of data strms .

(buffered manner)

Objective -- Restore collection of items created in above requirement ---in form of HashMap . -- buffering is optional.

Objective --- using Binary file streams.

Classes --- FileInputStream -- unbuffered bin i/p stream connected to bin file device.

FileOutputStream --unbuffered bin o/p stream connected to bin file device.

But these classes --- dont provide buffering & have only read() write() methods in units of bytes/byte[]

API of InputStream class

1. int read() throws IOException

Will try to read 1 byte from the stream.

Data un-available method blocks.

Returns byte--->int to caller.

eg -- int data=System.in.read();

2. int read(byte[] bytes) throws IOException

Will try to read data from underlying stream.

Data un-available -- method blocks.

Rets actual no of bytes read.

eg :

byte[] bytes=new byte[100];

int no=System.in.read(bytes);

no data --method blocks.

10 bytes available -no =10;bytes[0]-----bytes[9]

110 bytes available -- no=100;bytes[0]....bytes[99]

3. int read(byte[] bytes,int offset,int maxNoOfBytes) throws IOException

Will try to read data from underlying stream.

Data un-available -- method blocks.

Rets actual no of bytes read.

eg :

byte[] bytes=new byte[100];

int no=System.in.read(bytes,10,15);

no data --BLOCKS

5 bytes available --no=5;bytes[10].....bytes[14]

110 bytes available -- no=15;bytes[10]..bytes[24]

4. int available() throws IOException

Returns no of available bytes in the stream

no data ---DOESN't block -- rets 0.

Important API of OutputStream

1. public void write(int byte) throws IOException

2. public void write(byte[] bytes) throws IOException

3. public void write(byte[] bytes,int offset,int maxNo) throws IOException

bytes[offset].....bytes[offset+maxNo-1] -- written out to stream

4. void flush() throws IOException

5. void close() throws IOException

Using BIS(BufferedInputStream) -- enables buffering BUT doesn't provide any advanced API(ie. read(), read(byte[]), read(byte[] b,int off,int len) . Same is true with BOS.(BufferedOutputStream)

Objective ---

Create Customer/Account based collection. Sort if reqd.

Store Sorted collection to bin file in buffered manner --

& re-store the same.

Use advanced streams in such cases ---

Mixed Data streams

java.io.DataOutputStream ---implements DataOutput i/f

Constructor -- DataOutputStream (OutputStream out)

API ---

public void writeInt(int i) throws IOExc

public void writeChar(char i) throws IOExc

public void writeFloat,writeDouble.....

For Strings

public void writeUTF(String s) throws IOExc ---uses Modified UTF 8 convention

or

public void writeChars(String s) throws IOExc --- uses UTF16 convention

eg : Items Inventory

Item -- code(String-Prim key),desc,category,quantity,price,shipmentDate

constr,toString.

Objective ---

Customer data is already stored in bin file.

Read customer data from Bin file --- in buffered manner & upload the same in HM .display customer details.

Stream class --- java.io.DataInputStream -- implements DataInput

Constructor

DataInputStream(InputStream in)

API Methods

public int readInt() throws IOException

public double readDouble() throws IOException

public char readChar() throws IOException

public String readUTF() throws IOException(must be used with writeUTF)

public String readChars() throws IOException(must be used with writeChars)

Most Advanced streams ---

Binary streams which can read/write data from/to binary stream in units of Object/Collection of Object refs (i.e Data Transfer Unit = Object/Collection of Objects)

Stream Class for writing Objects to bin. stream

java.io.ObjectOutputStream implements DataOutput,ObjectOutput

Description --- ObjectOutputStream class performs serialization.

serialization= extracting state of object & converting it in binary form.

state of object = non-static & non-transient data members

Constructor

ObjectOutputStream(OutputStream out)

out--- dest Binary o/p stream --- where serialized data stream has to be sent.

API methods ---

public void writeInt(int i) throws IOExc

public void writeChar(char i) throws IOExc

public void writeFloat,writeDouble.....

For Strings

public void writeUTF(String s) throws IOExc ---uses Modified UTF 8 convention

public void writeObject(Object o) throws IOException,NotSerializableException

De-serialization---- conversion or re-construction of Java objs from bin stream.

java.io.ObjectInputStream --- performs de-serialization.--- implements DataInput,ObjectInput

Constructor --- ObjectInputStream(InputStream in)

API methods ---

readInt,readShort,readUTF,readChars..... +

public Object readObject() throws IOException

Objective --- attach OIS to the bin file using FIS & display customer data.

Objective :

Confirming concepts of serialization & de-serialization

Emp -- int id, String name,double salary,Address adr;

Address -- String state,city,street.

Objective -- Understanding Set & its implementation classes

HashSet -- based upon hashing algorithm

More involved scenario.

(store customer info & Items to be purchased)

Data members - int no,Customer info, AL<Item>, Date creationDate

**serialization\_readme**

Need -- In the absence of Object streams, if u want to persist(save in permanent manner) state of objects or application data in binary manner --- prog has to convert all data to binary & then only it can be written to streams.

Object streams supply ready made functionality for the same.

Serialization/De-serialization

Ability to write or read a Java object to/from a binary stream

Supported since JDK 1.1

Saving an object to persistent storage(current example -- bin file later can be replaced by DB or sockets) is called persistence

Java provides a java.io.Serializable interface for checking serializability of java classes.(object)

Meaning --- At the time of serialization(writeObject) or de-serialization(readObject) --- JVM checks if the concerned object is Serializable(i.e has it implemented Serializable i/f) --if yes then only proceeds , otherwise throws Exception ---java.io.NotSerializableException

Serializable i/f has no methods and is a marker(tag) interface. Its role is to provide a run time marker for serialization.

Details

What actually gets serialized?

When an object is serialized, mainly state of the object(=non-static & non-transient data members) are preserved.

----------------------------------------------------

If a data member is an object(ref) , data members of the object are also serialized if that object’s class is serializable

eg : If Item class HAS - A reference of ShippingAddress

The tree of object’s data, including these sub-objects constitutes an object graph

eg HashMap<Integer,Account>

out.writeObject(hm);

HM -- Integer,Account --- id,type,balance,creationDate, customerDetails(adr,email,code....)

If a serializable object contains reference to non-serializable element, the entire serialization fails

If the object graph contains a non-serializable object reference, the object can still be serialized if the non-serializable reference is marked “transient”

Details --- transient is a keyword in java.

Can be applied to data member.(primitive or ref)

transient implies ---skip from serialization.(meant for JVM)

Usage -- To persist --partial state of the serializable object

---------------------------------------

If super-class is serializable, then sub-class is automatically serializable.

If super-class is NOT serializable --- sub-class developer has to explicitely write the state of super-class.

What happens during deserialization?

When an object is deserialized, the JVM tries to bring the object back to life by making a new object on the heap that has the same state the serialized object had at the time it was serialized. 1. (Class.forName("com.app.core.Account")--class loading purpose,

2. Class.newInstance(),

3. setting state of the object from bin stream)

The transient variables, which come back have either null (for object references) or as default

primitive values.

Constructor of serializable class does not get called during de-serialization.

What are pre-requisites for de-serialization?

.class file for Class Obj to be de-serialized + Bin data stream containing state.

What is serialversion UID?

Each time an object is serialized, the object (including every object in its graph) is 'stamped' with a version ID number for the object's class. The ID is called the serialVersionUlD, and it's computed based on information about the class structure. As an object is being deserialized, if the class has changed since the object was serialized, the class could have a different serialVersionUID, and deserialization will fail.(java.lang.InvalidClassException). But you can control this - by adding your own UID.

-------------------------------

Serialization format overview

1. Magic no.

2. Serialization format version no.

3. Class desc -- class name,serial version uid,desc of data members(class signature)

4. State of the object.(non static & non transient data members)

-----------

Limitations

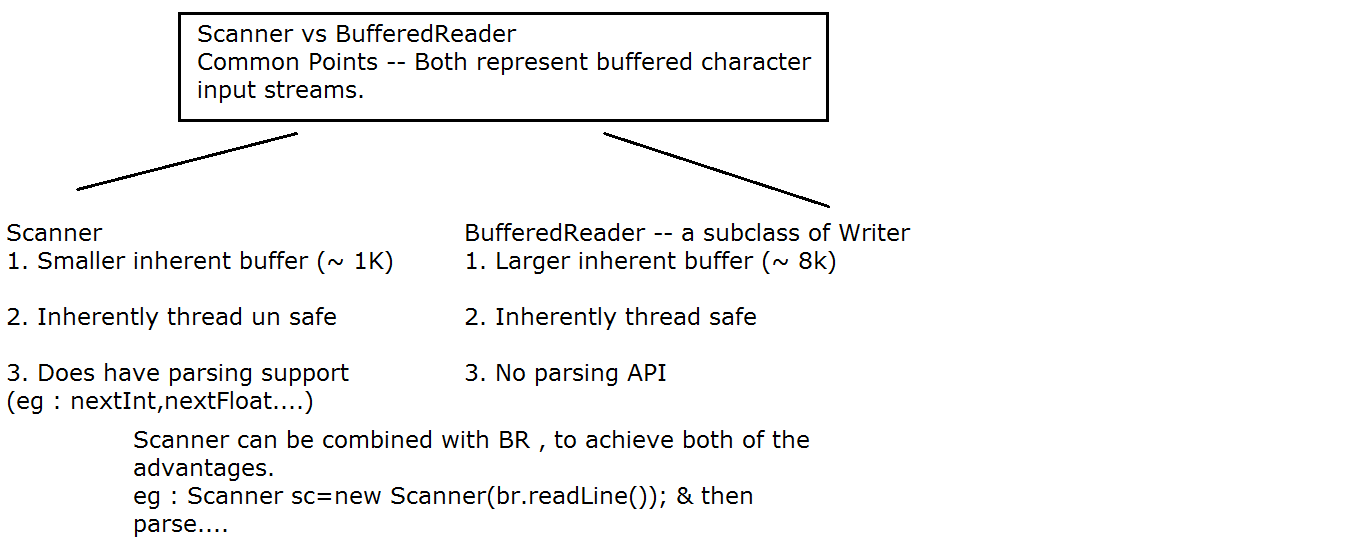
1. Java technology only

2. Difficult to maintain in case of changing class format

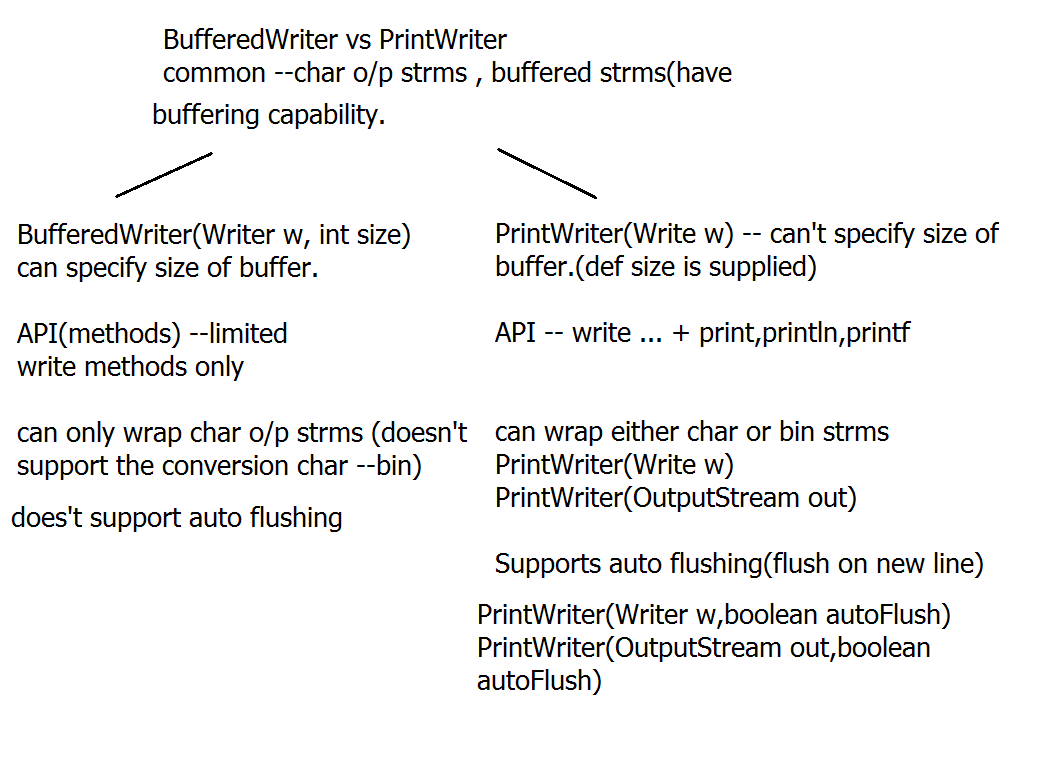
3. May lead to security leaks.

**Day 14**

**Scanner vs BufferedReader**

****

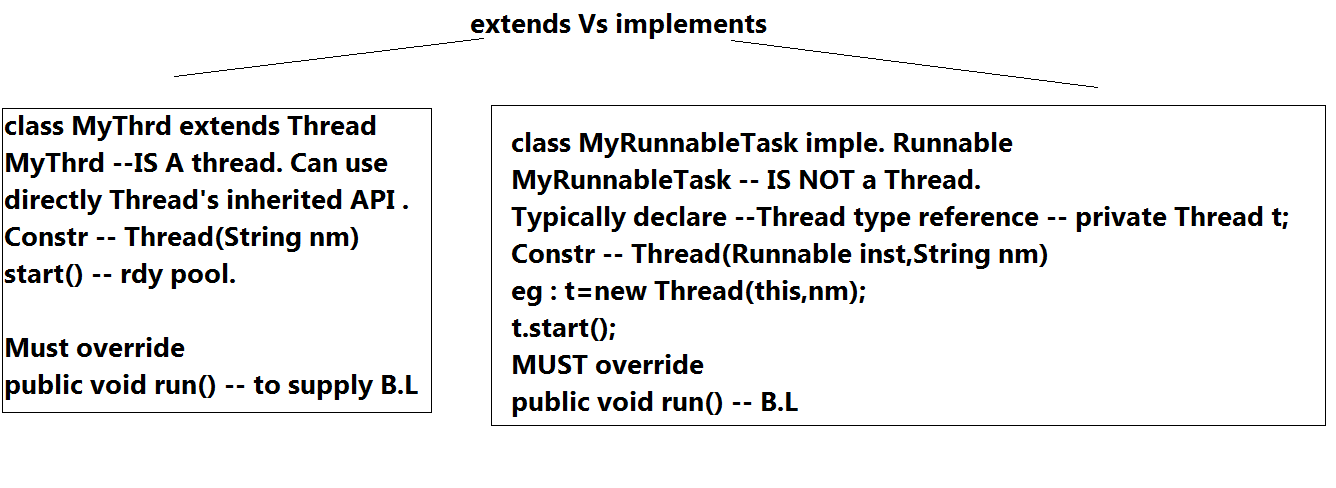
**BufferedWriter vs PrintWriter**

****

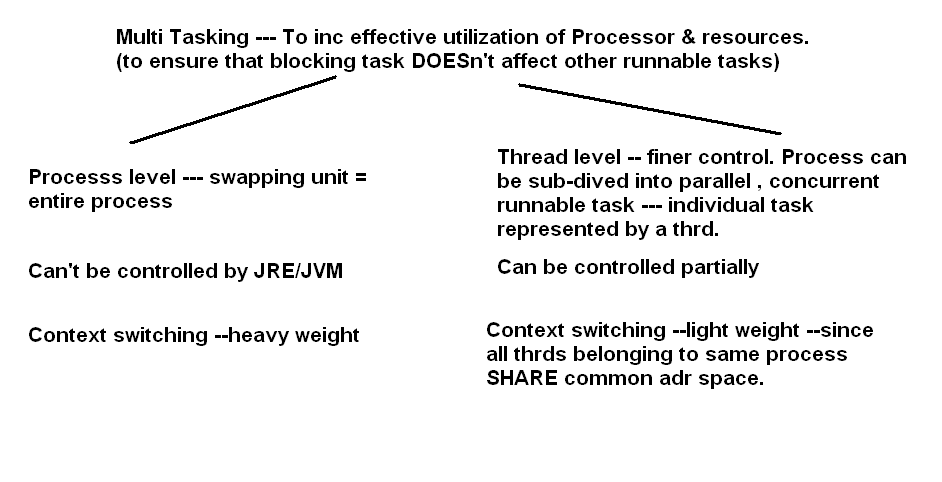
**Day 14**

**Thread help**

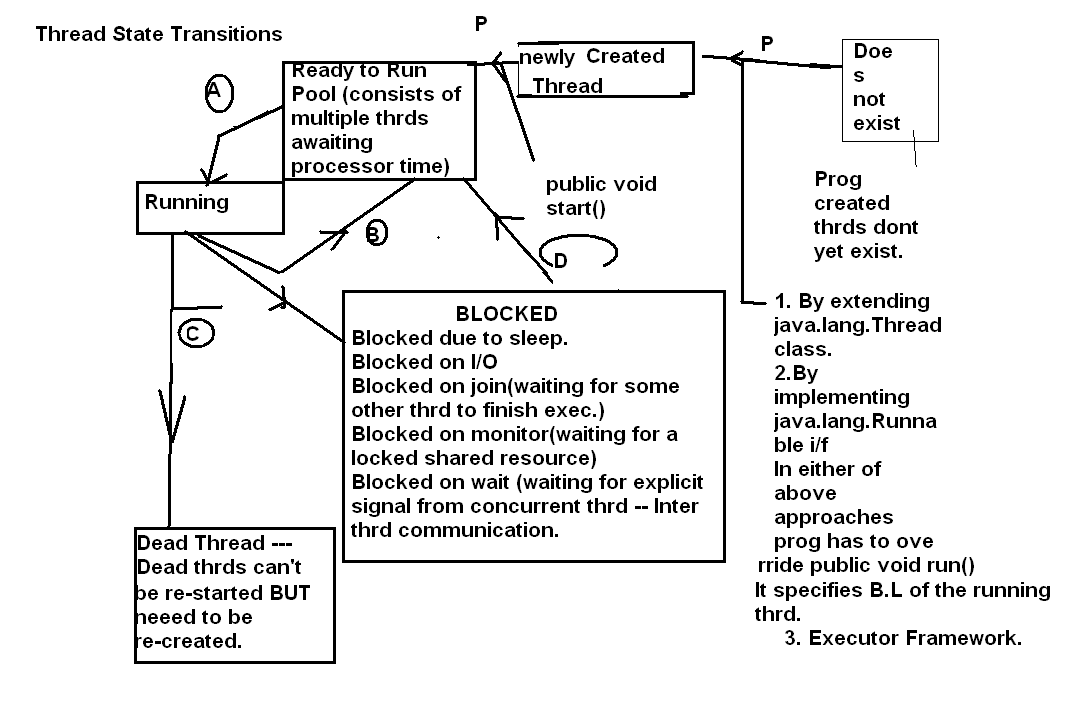
**extends\_vs\_implements**

****

**process\_vs\_thrds**



**Thread-state-transitions**



**Read me Threads**

Refer to Thread state transitions figure

A -- transition from rdy to run -----> Running

Triggered by --- in time slice based scheduling --- time slot of earlier thrd over OR in pre-emptive multitasking -- higher prio thrd pre-empts lower prio thrd.

B --- transition from running ---> ready to run

Reverse of earlier transition OR

public static void yield()----

Requests underlying scheduler to swap out current thrd SO THAT some other lower prio thrd MAY get a chance to run. (to avoid thrd starvation -- i.e co-operative multi-tasking.)

C --running state --- Only in this state --- run() method gets executed.

running --->dead --- Triggers -- run() method returns in healthy manner . OR run() aborts due to un-handled , un-checked excs.

D -- blocked ---> rdy to run --- when any of blocking condition is removed --- blocked thrd enters rdy pool & resumes competition among other thrds.

API Involved

1. Thread class constructor to be used in extends Thread scenario

1.1 Thread() --- A new thrd is created BUT with JVM supplied name.

1.2 Thread(String nm) -- creates named thread.

2. Thread class constructor to be used in implments scenario

2.1 Thread (Runnable target/inst) --- Creates a new thrd --- by passing instance of the class which implements Runnable i/f.

Run time significance -- Whenever this thrd gets a chance to run --- underlying task scheduler -- will invoke(via JVM) this class's run() method.

2.2 Thread (Runnable inst,String name)

public class Myclass extends Thread --- start()

Vs

public class Myclass implements Runnable --- This class simply represents a runnable task. Prog MUST create a thrd class inst BY attaching Runnable task to it.

Thread class API

1.public String getName()

2. public void setName(String nm)

3. public static Thread currentThread() -- rets ref of the invoker thrd.

4. public int getPriority() -- rets current prio.

Prio scale -- 1---10(MIN\_PRIORITY,MAX\_PRIORITY)

NORM\_PRIORITY ---- 5

4.5 public void setPriority(int prio) --- must be invoked before start()

DO NOT rely on priority factor -- since it is ultimately specific to underlying OS prio range.-- may cause loss of portability.

t1 --- max-prio ---run() --- begin up-counter -- obs val after 10 secs

t1 --- min-prio ---run() --- begin up-counter -- obs val after 10 secs

t1 --- max-prio & t2 -- min prio

5. public String toString() --- to ret -- name,prio & thrd grp name

6. public static void sleep(long ms) throws InterruptedException

Objective -1. to test concurrency of thrd : in extends thrd scenario.

Create a Thrd class, add simple loop with dly in run method to display the exec. sequence.

Write main method : which will instantiate thrds(multi-thrded system) & test the concurrent exec. of main thrd along with other thrds.

To ensure that main thrd terminates last : no orphan thrds in the system.

Thread class Method :

1. public void join() throws InterruptedExc.

The invoker thrd gets blocked until the specified thrd joins it(i.e specified thrd becomes dead)

eg : 2 thrds t1 & t2 are running concurrently.

If u invoke : in run() method of t1 :

t2.join() ----> t1 gets blocked until t2 is over.

t3 -- t1.interrupt()

t1 : Blocked on join

Unblcking triggers -- t2 over,getting interrupt signal.

2. public void join(long ms) throws InterruptedExc

waits max for specified tmout .

Unblocking triggers -- t2 over,getting interrupt signal, tmout exceeded.

Objective -- Ensure no orphans , w/o touching child thread class.

Objective

Replace for loop from thrds, by indefinite loop (while true) & still ensure -- no orphans.

Start --- main + 3 child thrds. --- main(parent thrd waits patiently for 5 secs. & then somehow forces termination of child thrds & then terminates last.

API ---

public void interrupt() -- sends interrupt signal to the specified thread. If specified thrd has invoked any method (sleep,join,wait)-- having throws clause of InterruptedExc --- then only thrd gets UNBLOCKED due to InterruptedExc.

NOTE -- Thread which is blocked on I/O -- CANT be un-blocked by interrupt signal.

Threads blocked by invoking any method -- having InterruptedException (sleep,join,wait) can be unblocked by interrupt signal.

2. Objective - to test concurrency of thrds : in implements Runnable scenario.

2.5 implements scenarion

3.

Objective : create 3 thrds with 3 random sleep durations(range is 500ms-5sec)

& start them conc. & ensure that main terminates last.

For random nos--- java.util.Random() , nextInt,nextInt(int upperLim)

Objective : create 3 thrds & start them conc. & ensure that main terminates last.

2nd thrd should accept data from console, dont supply data & observe .

How to unblock a thread , which is blocked on I/O?

Objective : Apply multi threading to Swing application.

Create Swing application -- with start & stop buttons, in south region.

Create JPanel in center region , with some default color.

When start button is clicked, center panel should start changing color(random color) periodically.

When stop button is clicked, stop changing color.

----------------------------------------

Objective -- To store emp details , dept wise in SAME data file.(text buffered manner--- PrintWriter )

Design :

0. Emp class --- id,deptid,name,sal

1. Write Utils class

d.m --- pw

constr --pw inst ---

void writeData(Emp e) {...}

void cleanUp() {...}

constr ---create PW inst --

PW pw=new PW(new FW("emps.data"),true);

---add writeData(....) : instance method to write Emp dtls (first name, last name ,deptId of the emp)

to the file with small dly in between.(why dly ? --- for simulating practical scenario & also to add randomness to code)

do u need clean up method- -- yes -- close pw.

2. Write Dept Handler runnable task class --- implements

Override run() method which will invoke writeData method till 'exit' condition is encountered.

Add stop/exit method to enable 'exit' flag.

....

{

Emp e;

Utils u;

constr(u,e)

{this.e=e;

//u=new Utils(); ------

public void run()

{

while(!exit)

u.writeData(e);

}

3. Write Tester class : accept some emp dtls . Create depthandler task per dept ,attach thrds & start them.

Wait for key stroke ---

System.in.read();

stop all child thrds

ensure no orphans

clean up -- pw

Wait for the key stroke : upon key stroke --- stop all child thrds & then finally terminate main.

Observed o/p ------garbled display or garbled data written in file.

Reason : multiple thrds trying to access the shared resource concurrently.

eg of shared resource : Console or file device,socket,DB table

Solution : Any time when asynch thrds need to access the SHARED resource : LOCK the shared resource --- so that after locking -- only single thread will be able to access the resource concurrently

When is synchronization(=applying thread safety=locking shared resource) required?

In multi-threaded java applns -- iff multiple thrds trying to access SAME copy of the shared resource(eg -- reservation tkt,db table,file or socket or any shared device)

How to lock the resource?

Using synchronized methods or synchronized blocks.

In either approach : the java code is executed from within the monitor & thus protects the concurrent access.

Note : sleeping thrd sleeps inside the monitor(i.e Thread invoking sleep(...) , DOESn't release the ownership of the monitor)

eg classes :

StringBuilder : thrd-unsafe.--- unsynchronized --- if multiple thrds try to access the same copy of the SB, SB may fail(wrong data)

StringBuffer --- thrd -safe ----synchronized internally--- if multiple thrds try to access the same copy of the SB, only 1 thrd can access the SB at any parti. instance.

which is reco class in single threaded appln? --- StringBuilder

multiple thrds -- having individual copies -- StingBuilder

multiple thrds -- sharing same copy -- StringBuilder --

identify code to be guarded -- sb 's api -- invoke thrd unsafe API -- from inside synched block.

ArrayList(inherently thrd un-safe) Vs Vector(inherently thrd safe)

HashMap(un-safe) Vs Hashtable(thrd safe)

synchronized block syntax --- to apply synchro. externally.

synchronized (Object to be locked--- shared resource)

{

//code to be synchronized --methods of shared res. -- thrd safe manner(from within monitor)

}

1. If any thrd is accessing any synched method of 1 obj, then same thrd or any other thrd CANT concurrently access same method of the same obj.(Tester1.java)

2. If any thrd is accessing any synched method of 1 obj, then same thrd or any other thrd CANT concurrently access same method or any other synchronized method of the same obj.(Tester2.java)

2. If thrds have their own independent copies of resources, synch IS NOT required.(Tester2.java)

3.If u are using any thrd un-safe code(ie. ready code without source) --& want to apply thrd safety externally --- then just wrap the code within synched block to use locking.(Tester3.java)

Objective : Create Producer & Consumer thrds .

Producer produces data samples & consumer reads the same.

For simplicity : let the data be represented by : single Emp record

Producer produces emp rec sequentially & consumer reads the same.

Rules : 1 when producer is producing data , consumer thrd concurrently should not be allowed to read data & vice versa.

Any more rules?????????????

Yes --- correct sequencing is also necessary in such cases.

Rule 2 : Producer must 1st produce data sample ---consumer reads data sample & then producer can produce next data sample. Similarly consumer should not be able to read stale(same) data samples .

ITC --- API level

Object class API

1. public void wait() throws IE ---thrd MUST be owner of the monitor(i.e invoke wait/notify/notifyAll from within synched block or method) --- othewise MAY get IllegalMontitorStateExc

---causes blocking of the thrd outside montitor.

UnBlocking triggers --- interrupt(not reco --- since it may cause death of thrd) , notify/notifyAll --- reco.

2. 1. public void wait(long ms) throws IE

UnBlocking triggers --- interrupt(not reco --- since it may cause death of thrd) , notify/notifyAll --- reco.,tmout exceeded

2.2 public void notify() -- MUST be invoked from within monitor , ow may get IllegalMonitorStateExc

Un-blocks ANY waiting thread , blocked on SAME MONITOR

2.3 public void notifyAll() -- Un-blocks ALL waiting threads , blocked on SAME MONITOR

notify/notifyAll--- DOESN't BLOCK the thread & Doesn't release lock on monitor. --- send wake up signal -- to thrd/s waiting on same monitor.

wait --- Blocks the thread --- Releases lock on the monitor.

volatile --- java keyword, applicable at data member.

typically used in multi-threaded scenario only when multiple thrds are accessing the same data member.

Use --- to specify-- that data var. is being used by multiple thrds concurrently -- so dont apply any optimizations(OR the value of the variable can get modified outside the current thrd) . With volatile keyword -- its guaranteed to give most recent value.

The volatile modifier tells the JVM that a thread accessing the variable must always get its own private copy of the variable with the main copy in memory

**Day 15**

**Read me synchronization**

Regarding synchronization

1. Only methods (or blocks) can be synchronized, not variables or classes.

2. Each object has just one lock.

3. Not all methods in a class need to be synchronized. A class can have both

synchronized and non-synchronized methods.

4. If two threads are about to execute a synchronized method in a class, and both threads are using the same instance of the class to invoke the method,only one thread at a time will be able to execute the method. The other thread will need to wait until the first one finishes its method call. In otherwords, once a thread acquires the lock on an object, no other thread can enter ANY of the synchronized methods in that class (for that object).

5. If a class has both synchronized and non-synchronized methods, multiple

threads can still access the class's non-synchronized methods. If you have

methods that don't access the data you're trying to protect, then you don't

need to synchronize them. Synchronization can cause a hit in some cases (or

even deadlock if used incorrectly), so you should be careful not to overuse it.

6. If a thread goes to sleep(or invokes join,yield,notify) , it holds any locks it has—it doesn't release them.

7. A thread can acquire more than one lock. For example, a thread can enter a

synchronized method, thus acquiring a lock, and then immediately invoke

a synchronized method on a different object, thus acquiring that lock as

well. As the stack unwinds, locks are released again. Also, if a thread acquires

a lock and then attempts to call a synchronized method on that same

object, no problem. The JVM knows that this thread already has the lock for

this object, so the thread is free to call other synchronized methods on the

same object, using the lock the thread already has.

eg :

class A {

private B b1;

synched void test()

{

...

b1.testMe();

}

}

class B

{

synched void testMe()

{

//some B.L

}

}

8. You can synchronize a block of code rather than a method.

When to use synched blocks?

Because synchronization does hurt concurrency, you don't want to synchronize

any more code than is necessary to protect your data. So if the scope of a method is

more than needed, you can reduce the scope of the synchronized part to something

less than a full method—to just a block. OR when u are using Thread un-safe(un-sunchronized eg -- StringBuilder or HashMap or HashSet) classes in your appln.

-----------------------------

Regarding static & non -static synchronized

1. Threads calling non-static synchronized methods in the same class will

only block each other if they're invoked using the same instance. That's

because they each lock on "this" instance, and if they're called using two different

instances, they get two locks, which do not interfere with each other.

2. Threads calling static synchronized methods in the same class will always

block each other—they all lock on the same Class instance.

3. A static synchronized method and a non-static synchronized method

will not block each other, ever. The static method locks on a Class

instance(java.lang.Class<?>) while the non-static method locks on the "this" instance—these actions do not interfere with each other at all.

**regarding race condition**

A race condition is a special condition that may occur inside a critical section. A critical section is a section of code that is executed by multiple threads and where the sequence of execution for the threads makes a difference in the result of the concurrent execution of the critical section.

When the result of multiple threads executing a critical section may differ depending on the sequence in which the threads execute, the critical section is said to contain a race condition.

Race condition means that the threads are racing through the critical section, and that the result of that race impacts the result of executing the critical section.

Critical Sections

Running more than one thread inside the same application does not by itself cause problems. The problems arise when multiple threads access the same resources. For instance the same memory (variables, arrays, or objects), systems (databases, web services etc.) or files.

In fact, problems only arise if one or more of the threads write to these resources. It is safe to let multiple threads read the same resources, as long as the resources do not change.

Here is a critical section Java code example that may fail if executed by multiple threads simultaneously:

public class Counter {

protected long count = 0;

public void add(long value){

this.count = this.count + value;

}

}

Imagine if two threads, A and B, are executing the add method on the same instance of the Counter class. There is no way to know when the operating system(scheduler) switches between the two threads. The code in the add() method is not executed as a single atomic instruction by the Java virtual machine. Rather it is executed as a set of smaller instructions, similar to this:

Read this.count from memory into PC register.

Add value to PC register.

Write register to memory.

Observe what happens with the following mixed execution of threads A and B:

this.count = 0;

A: Reads this.count into a register (0)

B: Reads this.count into a register (0)

B: Adds value 2 to register

B: Writes register value (2) back to memory. this.count now equals 2

A: Adds value 3 to register

A: Writes register value (3) back to memory. this.count now equals 3

The two threads wanted to add the values 2 and 3 to the counter. Thus the value should have been 5 after the two threads complete execution. However, since the execution of the two threads is interleaved, the result ends up being different.

In the execution sequence example listed above, both threads read the value 0 from memory. Then they add their individual values, 2 and 3, to the value, and write the result back to memory. Instead of 5, the value left in this.count will be the value written by the last thread to write its value. In the above case it is thread A, but it could as well have been thread B.

Race Conditions in Critical Sections

The code in the add() method in the example earlier contains a critical section. When multiple threads execute this critical section, race conditions occur.

More formally, the situation where two threads compete for the same resource, where the sequence in which the resource is accessed is significant, is called race conditions. A code section that leads to race conditions is called a critical section.

Preventing Race Conditions

To prevent race conditions from occurring you must make sure that the critical section is executed as an atomic instruction. That means that once a single thread is executing it, no other threads can execute it until the first thread has left the critical section.

**threads\_sequence**

Revise

process vs thrds

thread states.

extends vs implements

solve even/off no generator assignment

To explain need of synchronization, set up

1. Create Account class --with balance.

paramterized constr.

B.L

1.1 public void update(double amount) throws Exc

update balance

cancel updation

1.2 public double getBalance() throws Exc

return balance

2. Create UpdateBalance task

run -- invoke Accout's updateBalance continuously. --till exit

3. Create MonitorBalance task

run -- invoke Accout's getBalance continuously. --till exit

4. Tester

create Account class instance.

Pass the same instance to these 2 tasks. Attach threads & start them.

Wait for a keystroke.

enable exit flags.

ensure no orphans.

------------------------------------

1. Create a Printer class

public void print(String mesg) throws Exc

{

sop(" { ");

for(String s : mesg.split()) {

sop(s)

sleep(10);

}

sop("}");

}

2. Create a PrintableTask --Runnable

run -- while(!exit)

p.print(mesg);

3. Create a Tester --main based

3.1 Create Printer instance.

3.2 create 2 tasks --init different mesgs --share SAME Printer instance

3.3 Attach n strt threads.

3.4 wait for keystroke

stop the thrds (quit) --join --main over

-----------------

Regarding anonymous inner classes

They can't access method local var, from its enclosing scope --if its not final or effectively final(i.e value is not getting changed)

-----------------

2 ways of applying synchronization(=applying thrd safety)

1. synchronized method

2. synchrnoized block

syntax

synchronized(shared resource instance)

{

//methods of the resource --in thrd safe manner

}

eg :

HashMap<K,V> hm=new HM<>();

If u have shared this HM across multiple thrds --- for r & w -- in absence of synchro block --- java.util.ConcurrentModificationExc

Soln ---

synchronized(hm)

{

hm.put(....);

}

In other thrd

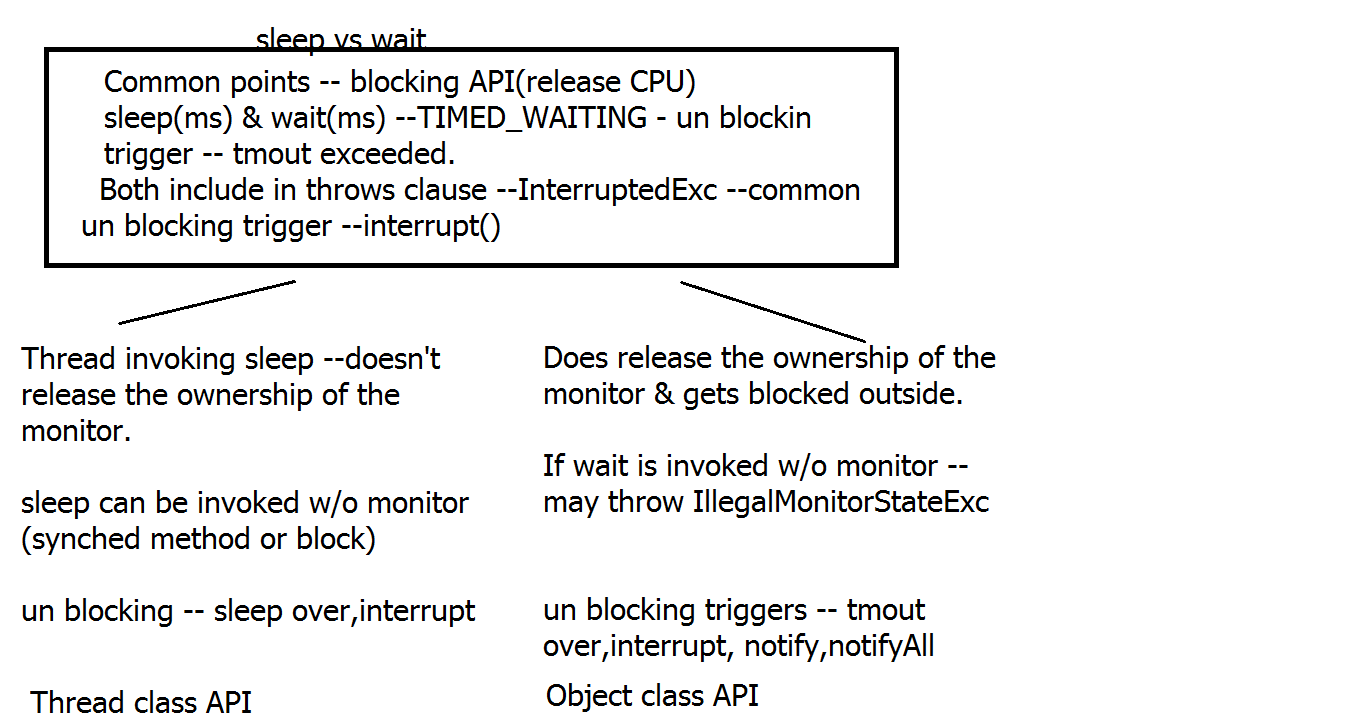
synchronized(hm)

{

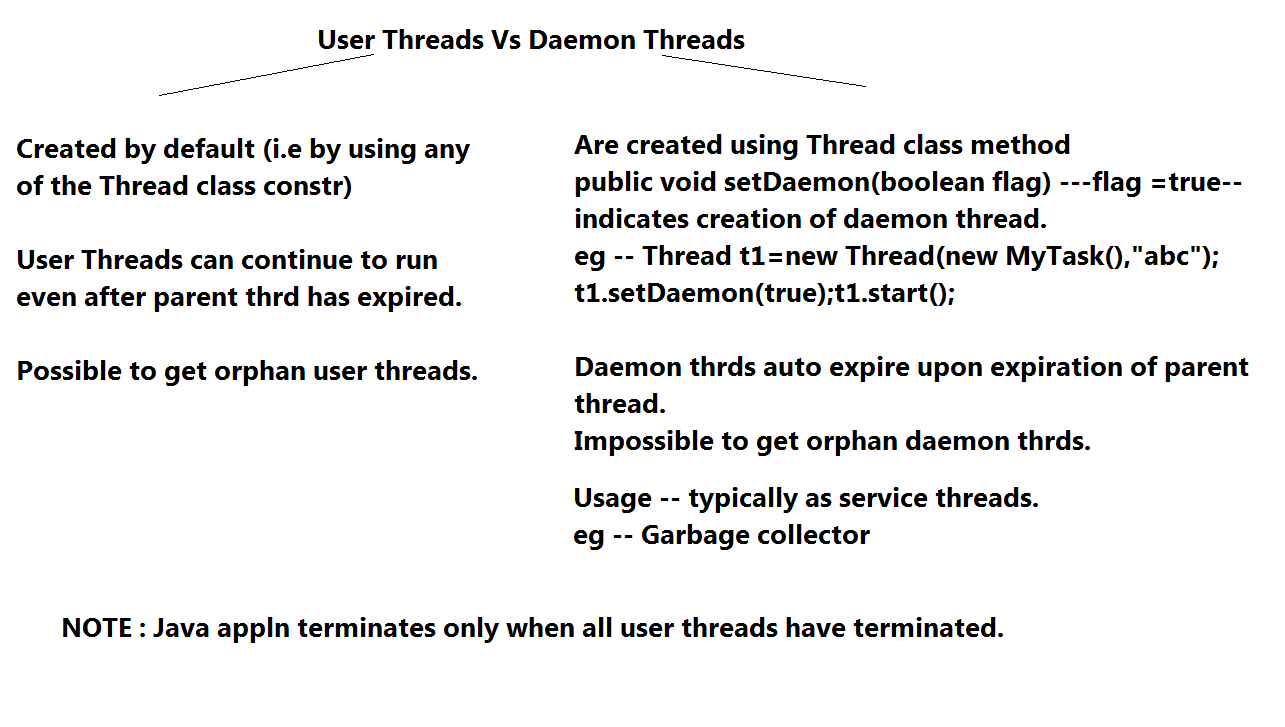
hm.get(....);

}

**Sleep vs Wait**

****

**user\_vs\_daemon thrds**



**Volatile**

volatile --keyword in java

Applied to data members

Typically used in multi threaded environment.

The Java volatile keyword guarantees visibility of changes to

variables across threads.

In a multithreaded application where the threads operate on non-volatile variables, each thread may copy variables from main memory into a CPU cache while working on them, for better performance . If computer contains more than one CPU, each thread may run on a different CPU. That means, that each thread may copy the variables into the CPU cache of different CPUs.

With non-volatile variables there are no guarantees about when the Java Virtual Machine (JVM) reads data from main memory into CPU caches, or writes data from CPU caches to main memory. This can cause several problems

Imagine a situation in which two or more threads have access to a shared object which contains a counter variable declared like this:

public class SharedObject {

public int counter = 0;

}

Imagine too, that only Thread 1 increments the counter variable, but both Thread 1 and Thread 2 may read the counter variable from time to time.

If the counter variable is not declared volatile there is no guarantee about when the value of the counter variable is written from the CPU cache back to main memory. This means, that the counter variable value in the CPU cache may not be the same as in main memory.

The problem with threads not seeing the latest value of a variable because it has not yet been written back to main memory by another thread, is called a "visibility" problem. The updates of one thread are not visible to other threads.

By declaring the counter variable volatile all writes to the counter variable will be written back to main memory immediately. Also, all reads of the counter variable will be read directly from main memory.

public class SharedObject {

public volatile int counter = 0;

}

Declaring a variable volatile thus guarantees the visibility for other threads of writes to that variable.

The Java volatile Happens-Before Guarantee

Since Java 5 the volatile keyword guarantees more than just the reading from and writing to main memory of variables. Actually, the volatile keyword guarantees this:

If Thread A writes to a volatile variable and Thread B subsequently reads the same volatile variable, then all variables visible to Thread A before writing the volatile variable, will also be visible to Thread B after it has read the volatile variable.

When a thread writes to a volatile variable, then not just the volatile variable itself is written to main memory. Also all other variables changed by the thread before writing to the volatile variable are also flushed to main memory. When a thread reads a volatile variable it will also read all other variables from main memory which were flushed to main memory together with the volatile variable.

Thread A:

sharedObject.nonVolatile = 123;

sharedObject.counter = sharedObject.counter + 1;//volatile var

Thread B:

int counter = sharedObject.counter; //reading volatile var

int nonVolatile = sharedObject.nonVolatile;

Since Thread A writes the non-volatile variable sharedObject.nonVolatile before writing to the volatile sharedObject.counter, then both sharedObject.nonVolatile and sharedObject.counter are written to main memory when Thread A writes to sharedObject.counter (the volatile variable).

Since Thread B starts by reading the volatile sharedObject.counter, then both the sharedObject.counter and sharedObject.nonVolatile are read from main memory into the CPU cache used by Thread B. By the time Thread B reads sharedObject.nonVolatile it will see the value written by Thread A.

When volatile is Not Enough ?

Even if the volatile keyword guarantees that all reads of a volatile variable are read directly from main memory, and all writes to a volatile variable are written directly to main memory, there are still situations where it is not enough to declare a variable volatile.

The situation where multiple threads are incrementing the same counter is exactly such a situation where a volatile variable is not enough.

If two threads are both reading and writing to a shared variable, then using the volatile keyword for that is not enough. You need to use a synchronized in that case to guarantee that the reading and writing of the variable is atomic. Reading or writing a volatile variable does not block threads reading or writing. For this to happen you must use the synchronized keyword around critical sections.

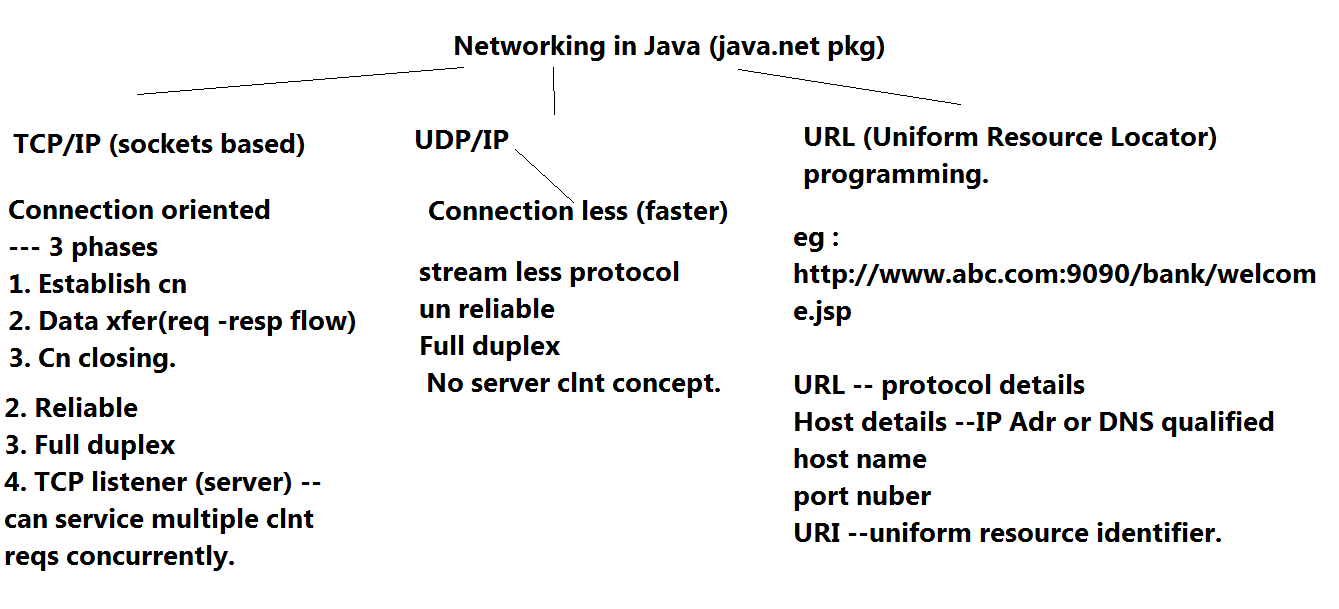
In case only one thread reads and writes the value of a volatile variable and other threads only read the variable, then the reading threads are guaranteed to see the latest value written to the volatile variable. Without making the variable volatile, this would not be guaranteed.

The volatile keyword is guaranteed to work on 32 bit and 64 variables.

**Day 16**

**Networking**

**Networking overview**

****

**Read me socket**

Pre-requisite : Visit & understand TCP-IP primer.

TCP

Connection oriented ---connection established ---data xfer ----conncetion closed.

Reliable -- ACK/NACK -- based upon CRC 16(checksum)

Stream based --- can attach data streams to socket(java.io)

full duplex---can send & receive concurrently

Server (listener ) -- can concurrently handle multpile client requests.

API : java.net Pkg

Any error in networking is represented by java.net.SocketException or IOException/ConnectExc./EOFExc

1. Server side steps

1.1

Create a server socket instance. (Control socket)

API : java.net.ServerSocket

Constructor

1.ServerSocket(int portNo) throws IOExc

portNO : 1024 onwards : local port No.

Def. backlog = 50.(backlog ---> The maximum queue length for incoming connection indications (a request to connect) . If a connection indication arrives when the queue is full, the connection is refused.

2. ServerSocket(int portNo,int backlog) throws IOExc

Desc : SS represents the port used for the purpose of the con. establishment.

When the con. is established bet clnt & server , TCP server side process will auto. route the data traffic to newly created data socket.

1.2 Place SS in 'wait for clnt req' mode.

API : ServerSocket has a blocking method :

public Socket accept() throws IOExc.

Above method will be blocked till : valid conn. establishment with the clnt occurs.

It will raise IOExc : if cn. establishment fails.

If cn estblshment succeeds :

Ret Type : Socket : representing Data socket : to be used for data xfer bet. clnt & server.

Cn is established

1.3

Attach suitable data streams.

Socket class API

To read the data from a socket :

InputStream getInputStream() throws IOExc.

To write data to a socket :

OutputStream getOutputStream() throws IOExc

U can't attach DIRECTLY Reader or Writer(i.e char strms) to a socket device.

1.4

As an example scenario : If u want to xfer strings in a buffered manner bet. server & clnt.

: Can use PrintWriter(OutputStream out) for sending data in char. buffered manner

& can use BufferedReader or Scanner to read the data in char mode.

1.5 Upon data xfer completion from a clnt, close cn (i.e close the data socket asso. with the clnt)

1.6 When data xfer from all clnts over or while exiting from server side appln, close SS & exit.

2. Client side steps

2.1

Establish the TCP conn. with a server by specifying the rem. Host(server host) IP addr/DNS qualified Host name & remote port no.

API : java.net.Socket(String hostName/ipAdr,int remPort) throws UnknownHostExc,IOExc,ConnectExc

If this rets : without the exc. : This marks the successful cn. establishment bet. clnt & server.

2.2

Attach suitable data streams.

Socket class API

To read the data from a socket :

InputStream getInputStream() throws IOExc.

To write data to a socket :

OutputStream getOutputStream throws IOExc

U can't attach DIRECTLY Reader or Writer(i.e char strms) to a socket device.

2.3 As an example scenario : If u want to xfer strings in a buffered manner bet. server & clnt

: Can use PrintWriter(OutputStream out) for sending data in char. buffered manner

& can use BufferedReader or Scanner to read the data in char mode.

2.4 When data xfer is complete, close socket & exit.

Objectives

1. Establish conn. bet server & clnt , create streams , exchange data(clnt says Hello Server,Server gets this & send response) & exit.

2. Clnt -- Establish connection with server. Accept numbers from user --- using JOptionPane.showInputDialog. After numbers , accept command (avg,sum,product) . Send collection of numbers , cmd to server . Wait for result , display result & exit.

Objective --- Server --- Accepts connection from the clnt. accepts collection of

numbers & command(add/multiply) --- performs operation & rets result to client & terminates.

2.1 Modify server only , clnt remains the same ---Prompts to user(server -side user) --- Want to service client? If yes --accepts cn from client & waits to accept data & cmd from client, upon accepting , computes result & sends the same to clnt. Server now should be able to support concurrent clients till --- server side user specifies end of server side appln.

Ensure that no orphan thrds are created.

1.5 Modify above for supporting full-duplex communication.

User should see received data on console.

Open i/p dialog box to send data. press ok & receiver should get it.

If user presses cancel, stop transmission. Take it as application termination signal, so close sockets & terminate

How to ?

For actual full-duplex support , xmit operation SHOULDN't be blocked by receive.

Must create thrds for asynch working of receiver & xmitter.

transmit thread -- responsible for send data

receiver thrd --responsible for rec data

main thrd -- creates above thrd , starts them & then get blocked till both thrds are over(join -- no orphans)

Then main() should ret terminating application.

, close socket, that should auto. terminate , receiver from remote end .

Main thread should stop last.

2.2

Objective --- In continuation with earlier EmpUtils --

Clnt.java --- GUI clnt (tcp clnt) --- in init -- connect to srvr , manipulate emp records & upon closing of the frm --send emp collection to the server & then clnt appln terminates.

Server --- sets up server side net. ---- receives collection from indi. clnts (max no of clnts) , adds the same in thrd safe manner to srvr side collection , & stores it to srvr side bin file before termination.

3. Customer care center ---Prompt for Center Name to user.

Establish cn to server .

clnt acccepts complaint from user , creates complaint object & stores it into suitable collection -- upon 'Register Complaint' button.

When clnt closes appln ---send center name & then upload collection of cust complaints to server & then exit.

Server --- accepts cn from clnts --- center name & complaints from all centers in suitable collection , save the same in bin file.

Write a separate tester on server side to confirm uploaded contents

3. Client side --- Represents Bank Branch -- GUI frm.-- Accept Branch code 1st. Create GUI to accept Bank account details, store them in a collection,view & remove.

Upon closing frm--- clnt should send collection to srvr & then terminate.

Srvr side --- non-GUI --- create Suitable collection to store all branch details. When all clnts terminate,display branch details & terminate.

3. Clnt sends emp id to the srvr, srvr gets the details from db & sends to the clnt

(covers clnt : GUI + Networking & for srvr : net +DB)

4.clnt side : emp data insertion form. add button : for adding it to the local coll.Send button to send the data to server, where server displays rcvd emp records. (gui+coll+i/o+net)

4. single chat clnt & srvr./ muliti chat

5. Client side --- JFrame using JFileChooser & JMenu , allow user to open any file (bin or text) & upload the same on send Button to the server.

Server side --- receives uploaded content & stores the same in suitable server side folder .

**socket\_closing**

NOTE :

1. When u ABORT(i.e w/o proper closing socket inst) clnt or srvr --AND other remote end is using br.readLine -- other end gets Socket exc --conn lost by peer(at br.readLine)

If u are using din.readUTF() --- u get EOFExc

NOTE that if other end is not reading or writing data from/to socket u will not get any exc.

1.5 When u ABORT(i.e w/o proper closing socket inst) clnt or srvr --AND other remote end is using sc.hasNextLine --it will ret false & then u can end receiver thread.

2.When u propery close socket inst on srvr or clnt & if other end is using br.readLine or any other read method --- then it rets null. BUT doesn't throw exc.

3.When u propery close socket inst on srvr or clnt & if other end is using sc.hasNextLine() or any other checking method (hasNext() or hasNextInt() ...) --- then it rets false. BUT doesn't throw exc.

**Day 17**

**JDBC**

**Read me jdbc**

WHY JDBC ?

JDBC ensures DB vendor independence to Java Applications or Applets.

HOW it grants DB independence?

1. JAR supplied by DB vendor or Driver vendor --- consists of JDBC driver -- ie. a converter for Java Data <-----> Native DB types & implementation classes , vendor specific.

2. JDBC API (java.sql) consists of largely --- Interfaces.

Sun supplies specifications or WHAT (i/fs ) & leaves implementation to DB vendors or 3rd party JDBC drvr vendors.

Generalized steps for DB connectivity

1. Place the JDBC driver in the Java classpath

Typically JDBC drivers are in form of JAR(Java archival format : compressed bundle of pkged Java classes) :

Oracle supplies Type IV Thin Client type of the Driver : ojdbc14.jar/classes12.jar/ojdbc6.jar

How to add JDBC drvr's JAR to the classpath(w/o IDE)

set classpath=g:\oracle\jdbc\lib\ojdbc6.jar;

With IDE --- simply Add external Jar.

2. Load & register JDBC driver

2.1 Load the JDBC driver in JVM's memory.

API :

pkg : java.sql,oracle.jdbc.\*

Class : DriverManager

DriverManager API :

2.2

public static void registerDriver(Driver instance) throws SQLException

eg : DriverManager.registerDriver(new OracleDriver());

OR

2.3

java.lang.Class<T>

public static Class forName(String F.Q clsName) throws ClassNotFoundException

Class.forName("oracle.jdbc.OracleDriver");--- can raise ClassNotFoundExc

3. Get the fixed DB connection thro' the JDBC driver.

API : java.sql.DriverManager (class)

public static Connection getConnection(String dbURL,String userName,String password) throws SQLException

Params : dbURL : URL to reach DB thro the drvr.

jdbc:oracle:thin:@HostDetails --- for oracle Type IV thin clnt driver

HostDetails = DBServerHost:1521:SID

eg : jdbc:oracle:thin:@localhost:1521:orcl

For WiMC lab PC -- jdbc:oracle:thin:@localhost:1521:xe

4.Create the JDBC statement

Connection i/f method

public Statement createStatement() throws SQLException :

creates an empty JDBC stmt to hold the query & exec.

5. Fill in the query & execute the same.

Statement i/f method

If query is : select : u must use executeQuery method.If the query is DML other than select(i.e insert,delete,update) or DDL then use the method executeUpdate

5.1 For select query :(for result set returning query)

public ResultSet executeQuery(String sql) throws SQLException

Returns the result set consisting of selected rows & cols.

5.2 For others : (queries not returning RST)

public int executeUpdate(String sql) throws SQLException

Returns the updated row count : indicating how many rows were affected.

6.1 Process the ResultSet

API : ResultSet I/f method

public boolean next() throws SQLException

Advances the RST cursor to the next row & returns true : if valid data or false if no results.(end of results)

If valid data exists : then read row data

Methods from ResultSet i/f

Type getType(int colPosition) throws SQLException( colPos : as it appears in RST)

or

Type getType(String colName) throws SQLException

Type=JDBC data type.

Mapping bet. Oracle Data types & JDBC data type

varchar/varchar2 : String

number(n) : integer

number(m,n) : double/float

date : java.sql.Date

TimeStamp : java.sql.TimeStamp

7 : Insertion of a row to the table (any query returning updateCnt)

Only changes are : query & replace executeQuery by executeUpdate.

8. If Java appln is exiting : close RST,close ST & then close Cn from the finally block or finalize method.(Typically closing Cn , closes all stmts & rsts.)

What is the type of the ResultSet so far created? : Forward type only & read only

To such a RST : previous() or abs. positioning meths(absolute(n) or relative(n) positioning meths will raise SE.

How to create a scrollable RST?

Replace step 4 by the following.

4.Create the JDBC statement to support scrollable RST

Connection i/f method

public Statement createStatement(int resultSetType,int concurrencyType) throws SQLException :

resultSet type : forward type(ResultSet.TYPE\_FORWARD\_ONLY) or scrollable(ResultSet.TYPE\_SCROLL\_INSENSITIVE OR ResultSet.TYPE\_SCROLL\_SENSITIVE) :

Difference between these 2

A result set that is TYPE\_SCROLL\_INSENSITIVE does not reflect changes made while it is still open and one that is TYPE\_SCROLL\_SENSITIVE does. Both types of result sets will make changes visible if they are closed and then reopened:

resultset Concurrency : read only result set(ResultSet.CONCUR\_READ\_ONLY) or updatable result set.(i.e can make changes to RST & same changes can also be applied to the DB table. can replace thus : insert,update,delete queries)

Additional API of scrollable ResultSet :

boolean absolute(int n) throws SE : tries to place the RST cursor on the nth row.

relative,afterLast,beforeFirst,first,last,previous,getRow

Why use PreparedStatement ?

1. PST represents pre-parsed & pre-compiled Stmts. At the time of creation of the PST, 3 steps out of 4(i.e parsing ,syntax chking,compiling ) take place. So when User reqs for data(eg : via a button click) the only remaining step is : fill in user data & exec query.

2. U can't pass the IN params to Statement , but can pass IN params to PST.

How to create PST?

1. Use Connection i/f method :

public PreparedStatement prepareStatement(String sql) throws SE

eg : PreparedStatememt pst=cn.prepareStatement("select \* from my\_emp where id=?");

? : IN param. to be filled prior to query exec.

The RST associated with above PST is of : TYPE\_FORWARD\_ONLY & CONCUR\_READ\_ONLY

How to make it scrollable?

API : Connection i/f

public PreparedStatement prepareStatement(String sql,int resultType,int concurrencyType)

throws SE

3. How to set IN params of PST?(to be invoked : in event listener : after user gives i/p or in server side code after clnt sends request)

API : PreparedStatement i/f

void setType(int placeHolderPos,Type value) throws SE

Type : JDBC data type

PlaceHolder pos : 1.....counted from left

eg : to set emp id .

pst.setInt(1,....);

4. exec the query

rst=pst.executeQuery();

5. process rst in the standard manner.

RMI clnt : sends emp id & RMI srvr contact DB : if emp exists ---sends emp info , ow. raises exc empnot found -- via pst.

CallableStatement : i/f from java.sql

Statement <--- PST <--- CallableStatement

why CST ?

1. Use CST to execute stored procedures & stored funs exisitng on DB

2. To pass IN,OUT,IN OUT type of params

Steps to invoke & exec. the stored proc/fun

1. Create CST

API : Connection i/f

public CallableStatement prepareCall(String invocationSyntax) throws SqlException

invocationSyntax for stored proc : "{call procName(?,?.....?)}"

invocationSyntax for stored fun : "{?=call funcName(?,?.....?)}"

? : represents IN,OUT or IN OUT param

{} : represent the esc seq. for the JDBC drvr. JDBC drvr will translate this invocation to a native DB invocation form.

2. Set IN params : methods inherited from PST

void setType(int placeHolderPos,Type value) throws SE

3. Register OUT / IN OUT params (i.e specify the JDBC data type of the OUT/IN OUT param to JVM)

void registerOutParameter(int paramPosition,int type) throws SE

paramPosition : placeHolder pos 1....

type : java.sql.Types : class constant

3.5 For IN OUT PARAM : invoke step 2 & 3 (ie. set IN val & register out param data type)

4. Execute the stored proc or a fun

public boolean execute() throws SE

Ret val is ignored.

5. Extract the results from OUT/IN OUT

CallableStatement methods

Type getType(int paramPos) throws SE

type : JDBC data type

Objective : Using scanner : accept sid,did,amt for funds transfer, exec the st.proc & disp the results.

DB Transactions

Functionally grouped SQL stmts : representing a B.L.

Tx => all the stmts from a Tx either fail or succeed.

i.e If any stmt fails : entire Tx has to be discarded.

The changes made by the Tx will be made permanent : IFF all the stmts succeed.

How to do it from JDBC API?

1. Start a Tx

Connection i/f method

void setAutoCommit(boolean false)

ie. unset the auto-commit flag.

eg : cn.setAutoCommit(false);

2. Wrap entire Tx within a separate try-catch block.

3. If the entire try block succeds (i.e at the end of try) ---> commit the Tx

API : cn.commit();

4.But if u reach inside the catch clause(due to system exc or custom exc) : rollback the Tx

API : cn.rollback();

5. To continue : set auto-commit to true again.

Updatable ResultSet :

How to create a PST which supports scrollable & updatable RST?

1. API : Connection i/f

public PreparedStatement prepareStatement(String sql,int resultType,int concurrencyType)

throws SE

resultSet type : TYPE\_SCROLL\_INSENSITIVE/SENSITIVE

concurrencyType : CONCUR\_UPDATABLE

2. Alternative to update query

2.1 Get the updatable RST.(eg : via pst.executeQuery())

2.2 Place the RST cursor on the row to be updated.(via absolute/relative meths of RST)

2.3 Update the col. vals-- on the RST

ResultSet API

public void updateType(int colPosition,Type newVal) throws SE

type--- JDBC data type

OR

public void updateType(String colName,Type newVal) throws SE

type--- JDBC data type

2.4 Once all changes to a particular row are done invoke :

API : public void updateRow() throws SE

to apply these changes to the underlying DB table.

3. Alternative to insert query

3.1 Get the updatable RST.(eg : via pst.executeQuery())

3.2 Place the RST cursor on the new row to be inserted.

API

ResultSet : void moveToInsertRow() throws SE

This places the RST cursor on the newly created row.

3.3 Update the col. vals-- on the RST copy

Invoke update methods (mandatory for NOT NULL constraint) : as in step 2.3

ResultSet API

public void updateType(int colPosition,Type val) throws SE

type--- JDBC data type

3.4 Once all col vals are inserted :

API : public void insertRow() throws SE

to apply these changes to the underlying DB table. (i.e new row gets inserted in DB)

3.5 To place cursor back to original row

API : public void moveToCurrentRow() throws SE

4. Alternative to delete query

4.1 Get the updatable RST.(eg : via pst.executeQuery())

4.2 Place the RST cursor on the row to be deleted (via absolute/relative)

4.3 Delete row :

ResultSet API

void deleteRow() throws SE (NOTE : deletes row from RST & DB too!!!!! use it with care!)

For date/time handling from JDBC

classes to be used from java.sql are :

Date,Time & TimeStamp

{d 'yyyy-mm-dd'}

{t 'hh:mm:ss'}

{ts 'yyyy-mm-dd hh:mm:ss'}

steps : for handling date

1. Create a table with col. type=date

2. Create a PST

3. Use java.sql.Date API

method :

public static Date valueOf(String dateFormat)

dateFormat : yyyy-mm-dd

4. Use PST's method

public void setDate(int pos,Date val) throws SE.

Meta data associated with JDBC

1. Database meta data : holds the info like : DB version,DB drvr version, Tx are supported or not, scrollable/updateble rsts, names of all tables from DB.....,max conns available

To get D.M.D

API : Connection i/f

DatabaseMetaData getMetaData() throws SE

DatabaseMetaData : i/f

Has methods : getVersion(),getTables().....

How to get all the table names for the current user?

Use DMD : method

ResultSet getTables(String catalog,String schemaPattern,

String tableNamePattern,

String[] types)

throws SQLException

Usage

DatabaseMetadata dmd=cn.getMetaData();

ResultSet rst=dmd.getTables(null,null,null,new String[] {"TABLE"});

//to retrieve table name

invoke : rst.getString(3) ; //3 => table name

2. ResultSetMetaData : metadata about the RST

How to get it?

Method in ResultSet API

ResultSetMetaData getMetaData() throws SE

eg :

ResultSetMetaData rmd=rst.getMetaData();

2.1 Methods of RMD

int getColumnCount() throws SE

String getColumnLabel(int colPos) throws SE

int getColumnType(int colPos) throws SE

Dirty Read --Enables un-committed tx data, to read from current tx.

Un-repeatable reads -- Enables to read committed data from concurrent tx, may lead to un repeatable results.

Phantom reads-- Enables to read committed data from concurrent tx, may lead to additional rows appearing in same tx.

Handling BLOBs with JDBC API

How to store BLOB data?

1. Create DB table having blob type of column.

eg create table my\_images(id int(3) primary key auto\_increment,name varchar(30),snap blob);

drop table my\_images;

2. Accept bin file from user to store on DB.

3. Use PreparedStatement API method -- to store BLOB on DB

API

public void setBinaryStream(int placeholderPos, InputStream in, int length) throws SqlException

4. Use executeUpdate to insert row data.

How to restore BLOB data from DB ?

1.Use API of ResultSet to read BLOB.

public Blob getBlob(int colPos) throws SqlException

2. Use java.sql.Blob i/f method

public byte[] getBytes(long pos,int length)

NOTE : pos begin with 1 .

3. Once u have byte[] , u can store the same on File(bin) using FOS or send it over sockets using Socket.getOutputStream()

Reference for MySQL connectivity

1.install MySQL

2. Clnt i/f

create database testjdbc;

use testjdbc;

create table Employee( empId int primary key, name varchar(25), deptId int, isPermanent boolean,sal double);

insert into Employee values(1,'aa',123,true,2000);

insert into Employee values(2,'ab',101,true,3000);

Driver class name : com.mysql.jdbc.Driver

To load/register driver ---- Class.forName(String F.Q className) throws ClassNotFoundExc

DB URL - jdbc:mysql://hostname:3306/databaseName

root -- user name

root -- password

example code for conn to MySQL ----

Class.forName("com.mysql.jdbc.Driver");

String dbURL="jdbc:mysql://localhost:3306/testjdbc";

//use DM.getConnection(url,username,pass)

Objective ---- RMI & JDBC integration

Func requirement --1. disp emp dtls --- if present , ow. raise cust exc.

2. Insert new emp record --- ret success msg or raise cust exc in case failure.

Server side steps

1. B.I --- method decl ---

String getEmpDtls(int empId) throws RE,EmpNotFoundExc

2. String insertEmp(emp specific dtls) throws RE,EmpInsertExc

2. Create impl class --- rem obj

constr --- cn,psts

B.M ---get ----

insert

HOW TO make JDBC applns/applets completely DB independent?

1.Create text based properties file.

key & value pair.(keys --- arbitrary values---changing as per DB setting)

2.Create empty java.util.Properties<K,V> --- sub-class of HashTable

Key & values must be --- String

Can load Properties directly from any stream.

Properties API

public void load(Reader r) throws IOExc

3.Can access the Property value using API

Properties API

String getProperty(String key)

ret type=value asso with key.

eg--

Properties props = new Properties();

FileInputStream in = new FileInputStream("database\_mysql.properties");

props.load(in);

in.close();

String drivers = props.getProperty("jdbc.drivers");

Class.forName(drivers);

String url = props.getProperty("jdbc.url");

String username = props.getProperty("jdbc.username");

String password = props.getProperty("jdbc.password");

return DriverManager.getConnection(url, username, password);

Regarding jar cmd line utility

0. For runnable jars --- create manifest.txt --- 1liner having Main-Class: tester.Test, new line & save file

1.cd to folder where ur classes are(eg bin)

1. From bin --- jar cvfm test.jar manifest.txt \*

2. To run jar

java -jar test.jar

**sql\_for\_mysql.cmds**

#to check DB status

status

#to show all databases

show databases;

# to create database

create database test;

#to use db

use test

# to show all tables

show tables

# to list all procedures

show procedure status

# to list all functions

show function status

#to view a stored procedure

show create procedure update\_account

#JDBC details

Driver class - com.mysql.jdbc.Driver

URL - jdbc:mysql://localhost:3306/test

User name - root

Password - root

#DDL

create table my\_emp (

empid int(3) primary key auto\_increment,

name varchar(20),

addr varchar(20),

salary double(8,2),

deptid varchar(10),join\_date date);

insert into my\_emp (name,addr,salary,deptid,join\_date) values('aa','pune',1500,'rnd','2011-11-18');

insert into my\_emp (name,addr,salary,deptid,join\_date) values('ab','chennai',2500,'prod','2015-1-20');

insert into my\_emp (name,addr,salary,deptid,join\_date) values('cc','mumbai',3500,'rnd','2004-11-18');

insert into my\_emp (name,addr,salary,deptid,join\_date) values('dd','delhi',3000,'prod','2005-1-20');

create table items

(id int(3) primary key auto\_increment,dscr varchar(30),price double(6,1));

insert into items (dscr,price) values('Core Java Books',500);

insert into items (dscr,price) values('Java EE Books',600);

insert into items values('XML Books',400);

insert into items values('.NET Books',700);

insert into items values('CORBA Books',300);

create table stock\_users (id int(3) primary key,pin int(6),name varchar(20),bal double(6,1));

insert into stock\_users values(101,1234,'aa',1000);

insert into stock\_users values(102,1235,'ab',2000);

insert into stock\_users values(103,1236,'ac',3000);

create table stock\_info(id int(3) primary key,name varchar(20),dscr varchar(30),price double(6,1),quantity int(5));

insert into stock\_info values (1,'tata','low-risk,medium-gain',100,150);

insert into stock\_info values (2,'tcs','medium-risk,medium-gain',150,250);

insert into stock\_info values(3,'reliance','high-risk,high-gain',200,300);

create table bank\_users(id int(3) primary key,name varchar(20) unique,

password varchar(20));

insert into bank\_users values(1,'a','b');

insert into bank\_users values(2,'c','d');

insert into bank\_users values(3,'e','f');

create table bank\_accounts(ac\_no int(3) primary key,id int(3) ,type varchar(10),bal double(6,1),CONSTRAINT fk\_users

FOREIGN KEY (id)

REFERENCES bank\_users(id));

insert into bank\_accounts values(101,1,'NRO',1000);

insert into bank\_accounts values(102,1,'NRE',2000);

insert into bank\_accounts values(103,2,'SAV',3000);

#Below will not work as integrity constraint parent key not found.

# insert into bank\_accounts values(104,20,'SAV',3000);

//to select all account's info for given user id

select a.ac\_no,a.type,a.bal from bank\_users u,bank\_accounts a where

u.id=1 and u.id=a.id;

CREATE TABLE person (

id INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT primary key,

firstName VARCHAR(45) NOT NULL,

lastName VARCHAR(45) NOT NULL);

create table accounts(id int(3) auto\_increment primary key ,name varchar(20),type varchar(10),bal double);

insert into accounts (name,type,bal) values ('aa','sav',1500);

insert into accounts (name,type,bal) values ('bb','sav',2500);

#stored procedure

drop procedure update\_account;

DELIMITER $$

create procedure update\_account (

in sid int ,

in did int,

in amt double ,

out sbal double,

out dbal double

)

begin

update accounts set bal=bal-amt where id=sid;

update accounts set bal=bal+amt where id=did;

select bal into sbal from accounts where id=sid;

select bal into dbal from accounts where id=did;

end$$

DELIMITER ;

DROP FUNCTION IF EXISTS F\_TEST

DELIMITER $$

CREATE FUNCTION F\_TEST(PID INT) RETURNS VARCHAR

BEGIN

DECLARE NAME\_FOUND VARCHAR DEFAULT "";

SELECT EMPLOYEE\_NAME INTO NAME\_FOUND FROM TABLE\_NAME WHERE ID = PID;

RETURN NAME\_FOUND;

END$$

DELIMITER ;

#stored function

CREATE FUNCTION CustomerLevel(p\_creditLimit double) RETURNS VARCHAR(10)

DETERMINISTIC

BEGIN

DECLARE lvl varchar(10);

IF p\_creditLimit > 50000 THEN

SET lvl = 'PLATINUM';

ELSEIF (p\_creditLimit <= 50000 AND p\_creditLimit >= 10000) THEN

SET lvl = 'GOLD';

ELSEIF p\_creditLimit < 10000 THEN

SET lvl = 'SILVER';

END IF;

RETURN (lvl);

END

DELIMITER $$

create function update\_account\_fn (sid number ,did number,amt number) returns number

DETERMINISTIC

begin

DECLARE dbal number;

update accounts set bal=bal-amt where id=sid;

update accounts set bal=bal+amt where id=did;

select bal into dbal from accounts where id=did;

return dbal;

end$$

DELIMITER ;

create table my\_customers (

id int(5) primary key auto\_increment,

deposit\_amt double(6,1),

email varchar(20) unique,

name varchar(20),password varchar(20),

reg\_date date,role varchar(10));

#my\_customers table data

insert into my\_customers values(1,1500,'rama@gmail','rama','1234','2010-1-23','admin');

insert into my\_customers values(2,1200,'kir@gmail','Kiran','123','2014-5-29','customer');

insert into my\_customers values(3,1800,'shek@gmail','Shekhar','1235','2016-3-4','customer');

#billing table

create table billing(mob\_no varchar(20) primary key,name varchar(10),plan varchar(10),call\_duration int(11) , total\_bill double(6,1));

insert into billing values('12345','aa','plan-A',100,50);

insert into billing values('12344','bb','plan-B',150,70);

insert into billing values('12346','cc','plan-A',80,40);

insert into billing values('12347','dd','plan-C',180,240);

delete from my\_emp where empid > 6;

1. mysql -u root

mysql> UPDATE mysql.user SET Password = PASSWORD('root') WHERE User = 'root';

mysql> FLUSH PRIVILEGES;

shell> mysql -u root

mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('root');

mysql> SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('newpwd');

mysql> SET PASSWORD FOR 'root'@'%' = PASSWORD('newpwd');

CREATE USER 'root'@'localhost' IDENTIFIED BY 'root';

#To alter column name/type

ALTER TABLE contacts CHANGE COLUMN contact\_type ctype varchar(20) NOT NULL;

# A trigger before insert for checking book price >= 0

drop trigger chk\_stats;

delimiter $$

create trigger chk\_stats before insert on test.book

for each row

begin

if NEW.price < 100 then

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Book Price must be >= 100';

end if;

end;

$$

delimiter ;

insert into book values('abc','a1',20);

# A trigger before update for checking book price >= 0

drop trigger chk\_stats;

delimiter $$

create trigger chk\_stats before update on test.book

for each row

begin

if new.price < 100 then

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Book Price must be >= 100';

end if;

end;

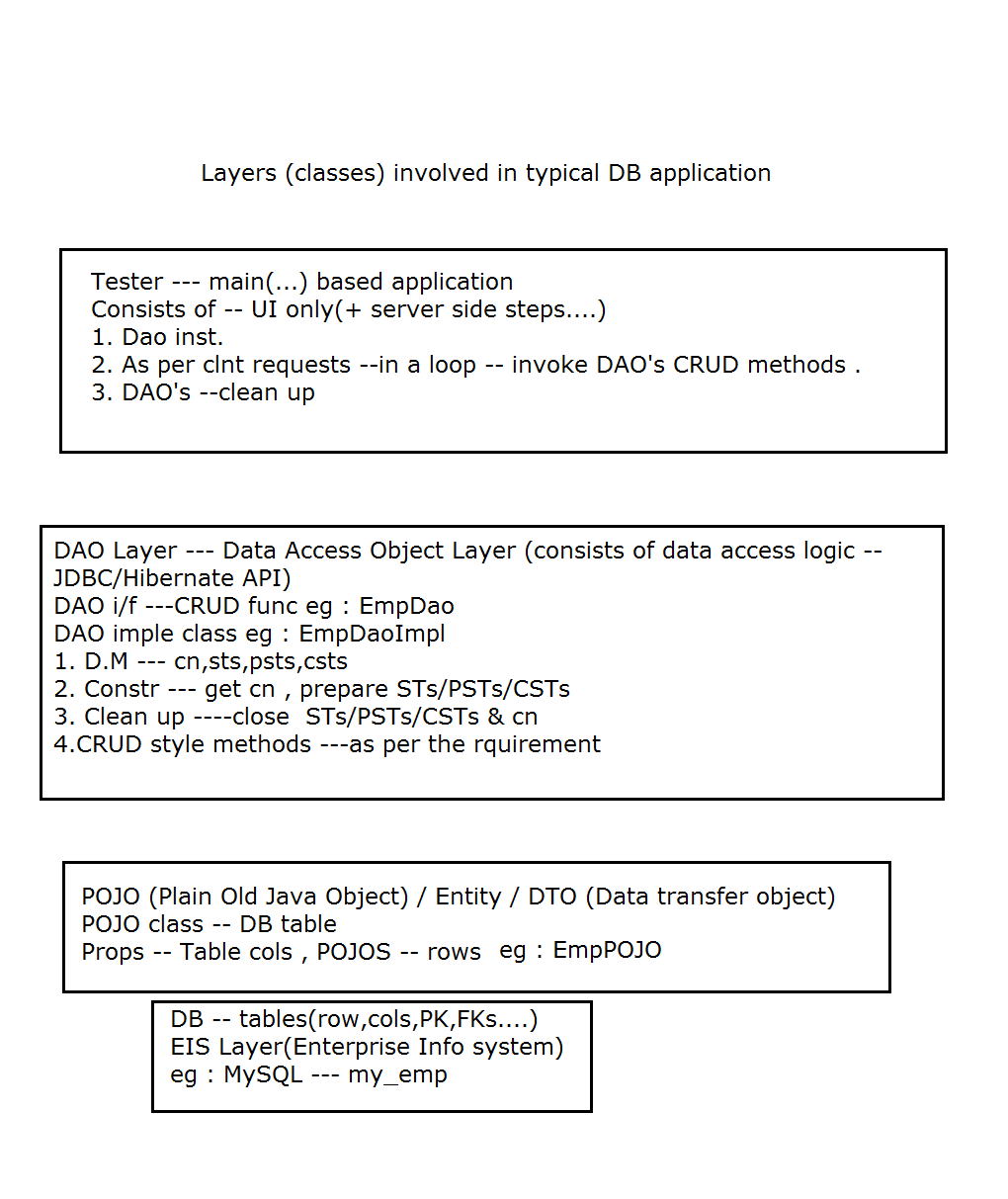
$$

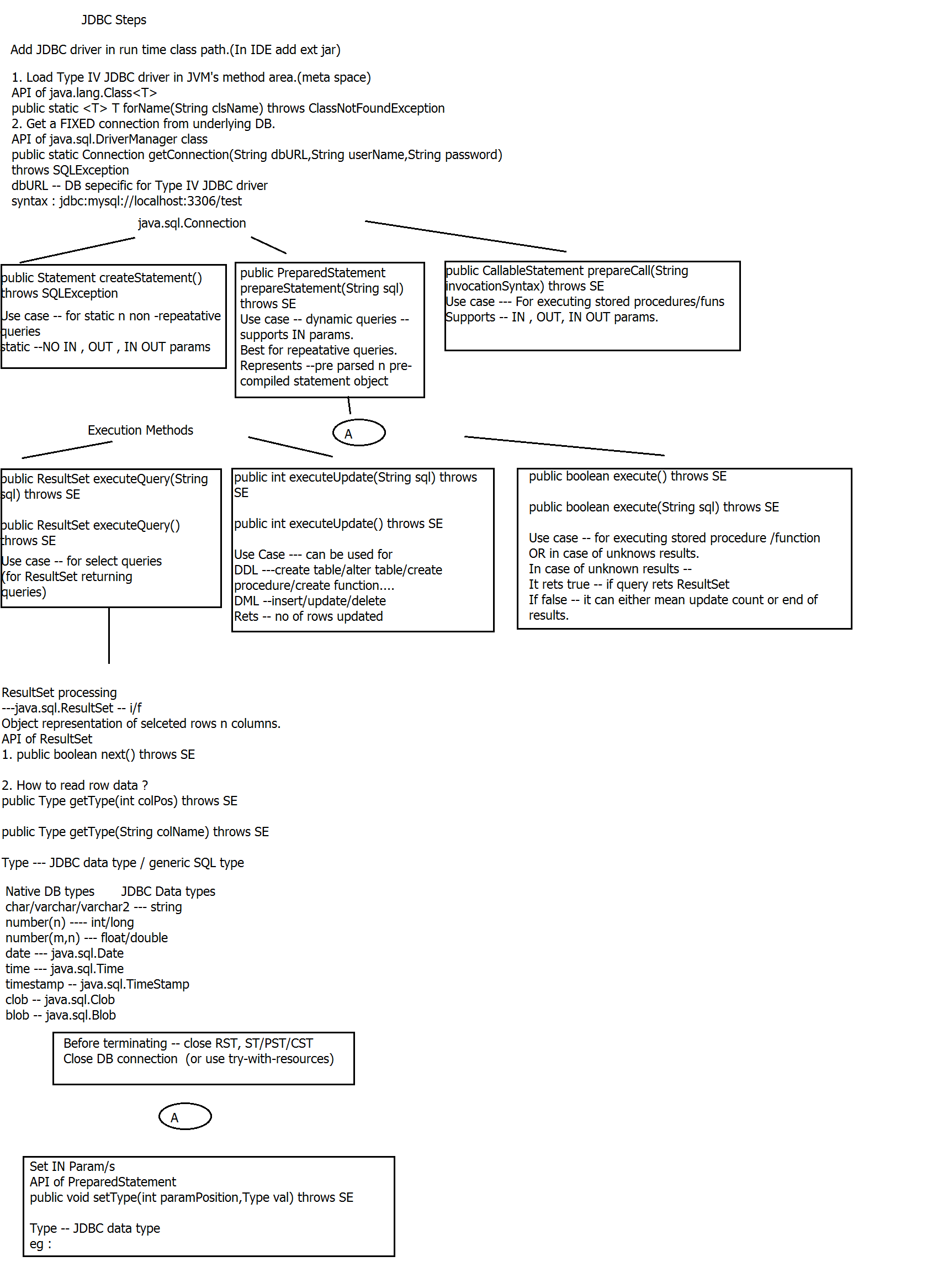
delimiter ;

update book set price=10 where isbn='1';

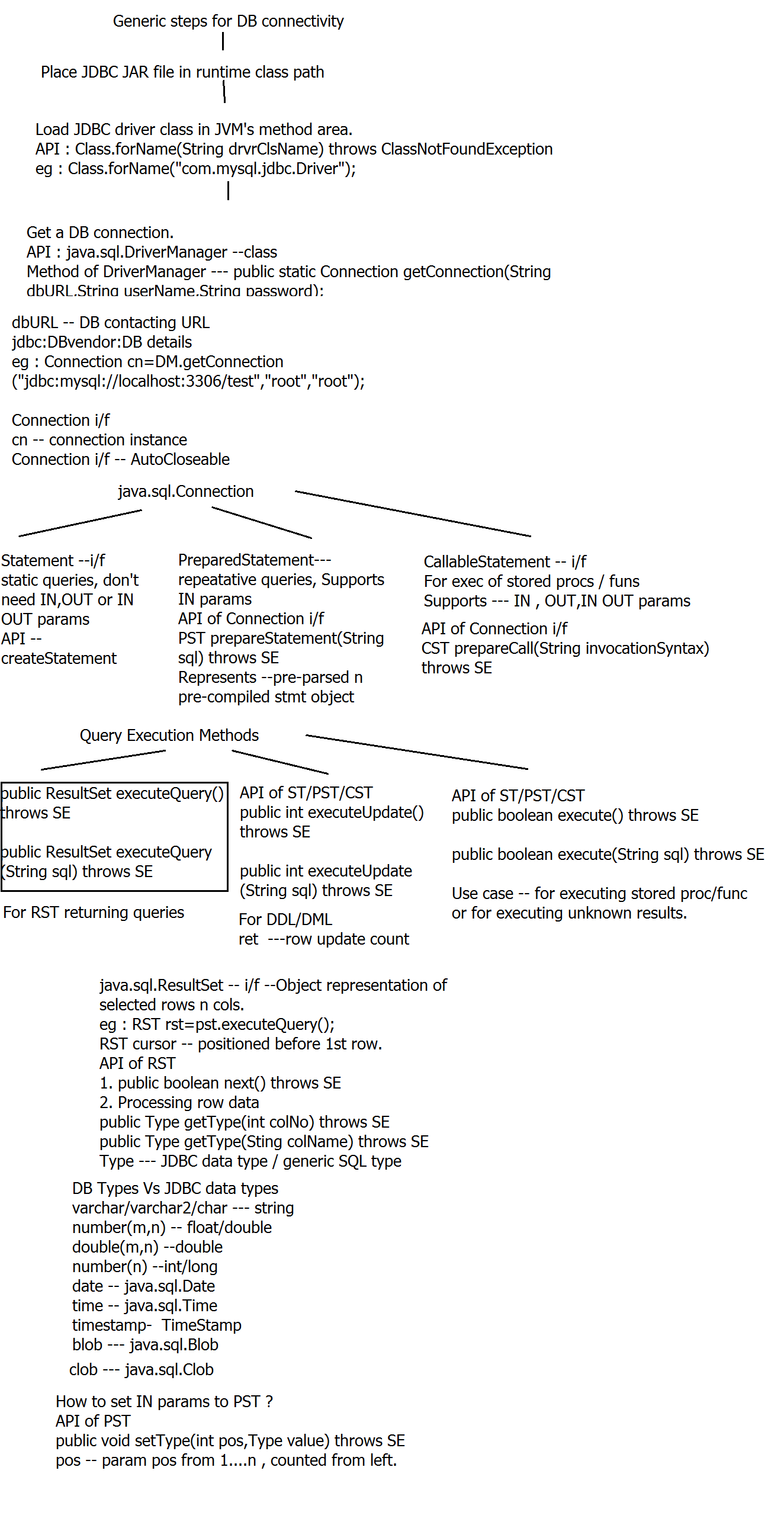
**Day 18**

**JDBC layers**

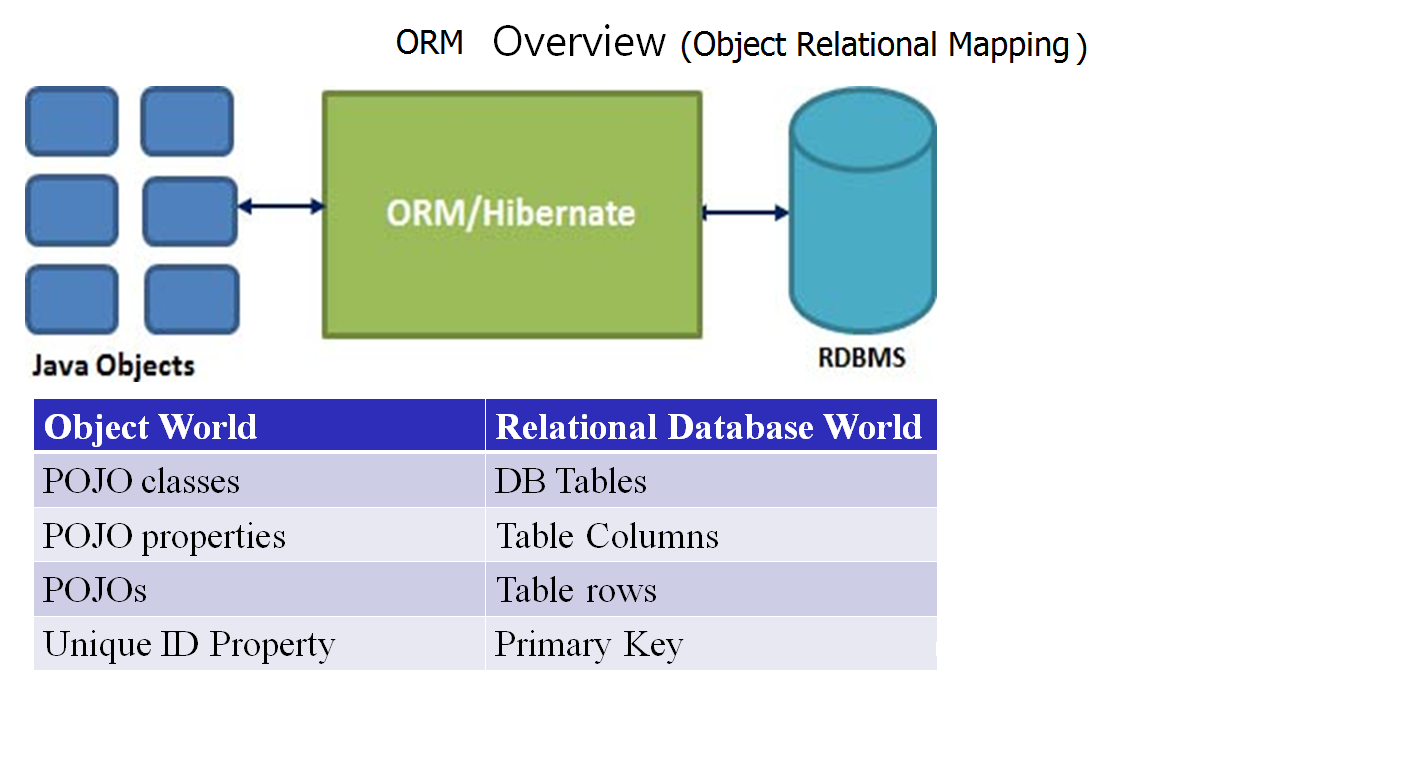
****

**JDBCOverView**

**JDBC steps**



**ORM overview**



**Multithreaded server design**

sequence

1. assignment status

2. Modify server to handle multiple clients concurrently.

Design of multi threaded server to handle concurrent clnts

1. main thread

1.1 init BL (eg : restore info , connect to dn & load info...)

1.2 Create SS with sufficienlty high backlog factor.

1.3 Create AL<Thread> to store clnt handling threads.--l1

1.4 for(int i=0;i<num;i++) or while (true)

{

Socket ds=ss.accept();

l1.add(new ClntHandlerThread(ds));

}

1.5 no new clnt connections --- ss.close

1.6 wait for clnt handling thrds to finish exec

for(Thread t : l1)

t.join();

1.7 clean up (store info/close db conn)

exit

2. ClntHandler Thread --inner class --extends Thread

why --so that it can access outer class's members directly.

constr

2.1 CH(Socket ds)

{

save ds

out

in

start();

}

2.2 public void run()

try{

while(true)

service(hm,in,out);

} catch (EOFExc )

{

//clnt terminated

ds.close

}

Complete ITC assignment

------------------------

Enter JDBC

What is it ?

Why ?

JDBC overview & steps.