Web Application Development Using Java

Nalluri Varun
CS 560
Computer Science Department
Illinois Institute of Technology
Chicago, Illinois 60616
vnalluri@hawk.iit.edu

Abstract – The drawbacks of C++ programming language led to the development of Java language. C++ is a platform dependent and won’t support internet and www. So James Gosling developed java language in the year 1991 at sun Microsystems. The web development has become easier with java because it is simple, Object oriented, Robust, platform independent, portable, Multithreaded, secure and Dynamic. Even today lot of the web development applications are done in java as it is easier and cheaper when compared to other technologies.

Index Terms – OOPS, Multithreading, JDBC, Servlets, JSP

INTRODUCTION

Java is an Object Oriented Programming Language. This paper gives a detailed description of all the java concepts including OOPS.

OOPS

Encapsulation:

Data hiding:

One of the sound object-oriented programming techniques is hiding the data within the class by declaring them with private accessibility modifier. Making such variables available outside the class through public methods. Data hiding says “restrict access and modification of internal data items through the use of getter and setter methods”.

Method abstraction:

Another sound object-oriented programming techniques is “Hiding method implementation complexity from outside users” is called as method abstraction. Encapsulation is nothing but Data hiding and method abstraction. Encapsulation combines (or bundles) data members with method members together (or the facility that bundles data with the operations that perform on that data is called as encapsulation).

Example:

```java
public class Authentication{
    private String username;
    private String password;

    public void setUsername(String name){
        if((name != null) && name!="")
            username = name;
    }

    public void setPassword(String pwd){
        if((pwd != null) && pwd!="")
            Password = pwd;
    }

    //Method abstraction
    public boolean isValidUser(){
        //Use JDBC API for obtaining connection.
        Connection con =
            DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",
                "scott", "tiger");
        Check with database whether user is already registered or not.
        If valid user then
            return true;
        else
            return false;
    }

    //Client program
    class AccessSite{
        public static void main(String[] args){
            Authentication auth = new Authentication();
            //set website username and password
            auth.setUserName("aspire");
            auth.setPassword("aspire123");
            boolean success= auth.isValidUser();
            if(success){
                System.out.println("Logged into INBOX…");
            }else{
                System.out.println("Invalid username / password…");
            }
        }
    }
}
```

Inheritance

Inheritance is a way to reuse code of existing objects, or to establish a subtype from an existing object, or both, depending upon programming language support. In classical inheritance where objects are defined by classes, classes can inherit attributes and behavior from pre-existing classes called base classes, superclasses, parent classes or ancestor classes. The resulting classes are known as derived classes, subclasses or child classes. The relationships of
classes through inheritance gives rise to a hierarchy. In prototype-based programming, objects can be defined directly from other objects without the need to define any classes, in which case this feature is called differential inheritance. Define common methods in base class and specific methods in subclass. The inheritance concept was invented in 1968.

As per the diagram here the rectangle class can make use of the move and resize methods which are inherited from its parent class Figure.

**Polymorphism** Polymorphism translates from Greek as many forms (poly - many morph - forms). Overloading and overriding are the two different types of polymorphism.

**Method Signature** It is the combination of the method name and its parameters list. Method signature does not consider return type and method modifiers.

**Method overloading**
Two or more methods are said to be overloaded if and only if methods with same name but differs in Parameters list (Differs in Number of parameters, Parameter type, or Parameter order), and without considering return type, method modifiers i.e. Overloaded methods must have different method signature irrespective of their return type, modifiers.
Example: public int add(int x, int y){}, public int add(int x, int y, int z){}
The overloaded methods are resolved by the compiler at compilation time. Hence overloading is also Known as ‘Static Polymorphism’ or ‘Early Binding’.

**Method overriding** A subclass can re-implement superclass methods to provide subclass specific implementation. The method in superclass is called as overridden method, whereas method in subclass is called as overriding method.
Method overriding rules:
The accessibility modifier of subclass method must be same or less restrictive. The superclass and subclass methods must have same return type. The superclass and subclass method signature must be same (i.e., method name, the number of parameters, parameter types, parameter order). The throws clause in overriding method can throw all, none or a subset of the checked exceptions which are specified in the throws clause of the overridden method, but not more checked exceptions. The method overriding is resolved at runtime based on actual object type but not Reference type. Hence method overriding is also known as Late Binding or dynamic polymorphism.

**Modular programming**: It is a software design technique that increases the extent to which software is composed of separate, interchangeable components called modules by breaking down program functions into modules, each of which accomplishes one function and contains everything necessary to accomplish this. Conceptually, modules represent a separation of concerns, and improve maintainability by enforcing logical boundaries between components. Modules are typically incorporated into the program through interfaces.
A module interface expresses the elements that are provided and required by the module. The elements defined in the interface are detectable by other modules.
The implementation contains the working code that corresponds to the elements declared in the interface.

**Message passing**: In computer science, it is a form of communication used in parallel computing, object-oriented programming, and interprocess communication. In this model, processes or objects can send and receive to other processes. By waiting for messages, processes can also synchronize.

**Constructors**: The purpose of constructor is initialization and instantiation. It is used for initializing the variables and for creating objects. The name of the constructor must be same as class name. The constructor must not return any type. We don’t need to call the constructor explicitly, rather it is triggered automatically.
Example:
Public class Box{
    Int width;
    Int height;
    Int depth;
    public Box(int w, int h, int d){
        width = w;  //initialization
        height = h;
        depth = d;
    }
}  
Box b = new Box(1,2,3);  //Instantiation

**Exceptions**: An unexpected problem occurs while running our application at runtime is called as an exception. This shouldn’t be a problem at compile time. So these exceptions are to be handled carefully for the smooth run of our program without any issues at run-time. There are different types of exceptions; some of them are interrupted exception, runexception, IOexception. The runexceptions are called as un-checked exceptions. Handling of checked exceptions is mandatory.
Example: division by zero is an arithmetic exception which is caused at run-time.
MULTITHREADING

A program which is ready to execute is called as process. A large program is split into multiple tasks, assigning a thread for each task and execute them concurrently is called as multithreading. A thread is a flow of control with in a program. It is a smallest unit of dispatchable code. The code which we want to run is mentioned in run() method. Defining a thread can be done in two ways, 1. Extending a java.lang.Thread class or by implementing a java.lang.Runnable interface.

Synchronization: If a method is declared as a synchronized, then only one thread is allowed to execute at a time. Every object in java has an implicit lock. If a thread wants to execute synchronized method, first it has to make a lock on the object. Once the thread got an object lock, it is allowed to execute the synchronized method. After completion of the method execution, then the object lock is released. Best example of application of synchronization is online booking of movie tickets.

Inter thread communication: Threads can communicate with each other. There are three methods which are used to communicate among threads. They are wait(), notify(), notifyAll(). These three methods are defined in java.lang.object class but not in java.lang(thread class. These methods must be called from synchronized context; otherwise we will get a Runtime error. Whenever a thread calls wait method, then it has to release the lock first and then gets into a wait state. If it wants to notify other threads it uses notify and notifyall depending on the need.

JAVA DATABASE CONNECTION

JDBC API is used to interact with databases for storing application specific information from Java application in database. JDBC Driver is capable of accessing database, spreadsheets and flat files.

JDBC Architecture:

Currently there are more than 220 JDBC Drivers available in the market. All of them are categorized into four types. They are JDBC_ODBC bridge driver, Native-API partly-java driver, Java Native Net protocol Driver, Java to Database protocol. The basic steps to use JDBC are 1) Declaring resources- the database resources are to be declared. 2) Load and Registering Driver class with DriverManager ex: Class.forName("Oracle.jdbc.Driver,OracleDriver"). 3) Defining JDBC URL. Usually it is Type 4, which is jdbc:oracle::thin:@<server hostname>:1521:xe. 4) Establishing connection with database using connection con. 5) Preparing SQL Query. 6) Create one of the JDBC statements. 7) Execute SQL commands. 8) Process the results such as number of records inserted, deleted or updated. 9) Handle Exceptions to make sure no error occurs while runtime.10) Release Resources to avoid memory leak problems.

Advanced JDBC Concepts:

BatchUpdates: It allows us to submit multiple DML operations without SELECT to the database as a single unit. It improves the performance by reducing the number of hits to the database i.e., minimizes network traffic. Batch Updates are supported since JDBC 2.0 / JDK1.2. The methods used in batch updates are, 1) addBatch() 2) executeBatch() 3) ClearBatch().

Rowset: It is an interface in javax.sql package. The RowSet is a thin wrapper on ResultSet object. RowSet object follows the JavaBeans model for properties and event notifications, i.e., it is a JavaBeans component. The RowSet interface provides a set of JavaBean properties such as: I. Connection properties (url, username, password) II. DataSource property (dataSourceName) III. Transaction Isolation property (transactionIsolation) IV. Command property for executing query (command) V. ResultSet type (type) VI. Concurrency (concurrency) VII. Event properties – Allowing other components in an application to be notified when an event occurs on a rowset. The different RowSet implementation classes are: JdbcRowSet, CachedRowSet, WebRowSet, etc. By default, all RowSet implementation classes are Scrollable and Updatable ResultSets, hence they are used when the underlying JDBC Driver does not support Scrollable &
Updatable ResultSets. The RowSet have many different implementations and these will fall into two broad categories: Connected & Disconnected.

**ConnectionPooling:** The connection pooling mechanism is used to establish pool of connections (i.e., group of connections) before the client makes a request. The connection is retrieved from the pool, used by the application, and finally returns back to the pool. The Connection Pool contains physical connections. With Connection Pooling, the connections are reused but not recreated every time when a connection is needed. The advantage of Connection Pooling is which improves Performance. Java application will not establish connection here, rather it will just get a connection from connection pooling which is called logical connection. It will be used and finally connection is returned back to the pool.

![Connection Pooling Diagram](image)

**Transaction Management:** Unit of work is called as Transaction. The transaction properties are ACID properties.

1) **Atomicity:** Either All or None of the database operations are executed within a transaction is called as atomicity. In JDBC, the atomicity is implemented by using setAutocommit(boolean), commit(), rollback() methods on connection object. By default, auto commit mode is enabled, means every database operation is executed in its own transaction. Hence explicit transaction management is not required. But sometimes, we may need to group more than one SQL statements into a single transaction by disabling auto commit.

2) **Consistency:** If the data used in transaction is consistent before starting transaction, then it will be consistent even after the end of the transaction (success or failed). Atomicity automatically supports consistency. Isolation intern supports consistency.

3) **Isolation:** Hide (or isolate) the common data from other transactions until the current transaction ends. This is done by the database by using locks on data.

4) **Durability:** If the transaction succeeds, the database system guarantees that its updates will persist even if the database system got crashed. One the system is up and running, it should be able to load previously crashed data.

**SERVLETS**

Servlet is a server-side java program (called as Web component), runs on Servlet Container (part of Web Server), which accepts HTTP (mostly) Request from client browser, process it and generates dynamic content, and finally sends generated content as HTTP Response back to the client browser. Servlets are used for developing web applications. Servlet containers are used to execute Servlets or JSP components. Servlet container is also called as Servlet engine. The following steps shows how web server and web browser communicate each other, 1) Network connection is established between End User machine (Web browser) and Client machine (WebServer). 2) Web browser sends HTTP Request Message to Web Server. 3) The Servlet which is running inside Web Server will process the request, generates dynamic content, and finally sends generated content back as HTTP Response Message to the Web browser.

**Servlet Container** It is also called as Servlet engine, which provides runtime environment for Java Servlet (JSP) components. The services provided by the Servlet Container as follows: 1) Manage Servlet life-cycle. 2) Session Management. 3) Resource Management. 4) Security service

**Servlet API** We can develop servlet applications using following two packages: I. javax.servlet package This package contains mostly interfaces and few classes to develop servlet applications irrespective of the protocol such as SMTP, POP, FTP, etc. II. javax.servlet.http package , This package contains classes and interfaces to develop http based servlet applications. We can write a servlet class by implementing Servlet interface directly. But every method of the Servlet
interface must be implemented irrespective of whether they required or not. Hence, it is recommended to write a servlet class by extending GenericServlet class. This class is inherited from Servlet interface, and provides implementation for every method except service() method. Hence, it is declared as an abstract class.

**JAVA SERVLET PAGES**

JSPs are used to create Dynamic web pages in web applications. JSPs are preferred over servlets to create views, since JSPs are tag based. JSP page contains HTML tags plus JSP tags. HTML tags are used to create static content and JSP tags are used to add dynamic content to web pages. We can use java code to add dynamic content by using different JSP tags. Programming in JSP is easier than in Servlets because the introduction of tag-based approach in JSP.

JSP Life-cycle: The server side pages in other languages are parsed every time they are accessed and hence expensive. But in J2EE, the expensive parsing is replaced by generating java class from the JSP using JSP Translator or JSP Engine. A typical JSP is interpreted in the web container and the consequent generation of HTML. This HTML is then rendered on web browser. JSPs are called server side pages because it is the JSP Engine which interprets them to generate HTML. The generated HTML is sent to the client browser. Every JSP is parsed at run time to generate servlet class using JSP Translator or JSP Engine. Hence most of the JSP life cycles are similar to servlet life cycle. 1) Translation phase 2) compilation phase 3) Loading and Instantiating 4) jspinit() method 5) jspService() 6) jspDestroy method.

**ACKNOWLEDGMENT**

I got to know a lot of new things while writing this paper. So, I would like to thank my Prof. Charles R. Bauer for giving this opportunity.

**REFERENCES**

http://docs.oracle.com/cd/B14099_19/web.1012/b14017/overview.htm
http://www.linuxjournal.com/article/1363
http://www.oracle.com/technetwork/articles/javase/servlets-jsp-140445.html