PT

Tutorial - 2.

Date
Page

Bishal Poudel.
191318

6. What is JIT compilation ? Differentiate JVM, JRE and JDK.

⇒ JIT (Just in time) are advanced part of Java virtual machine which optimize byte code to machine instruction conversion part by compiling similar byte code at same time and thus reducing overall time. It is the part of JVM and also performs several other optimizations such as inlining function.

Difference between JVM, JRE & JDK:

JRE = Java Runtime Environment
JVM = Java Virtual Machine
JDK = Java Development Kit

- JRE and JDK comes as installer while JVM are bundled with them.
- JRE only contain environment to execute java program but doesn't contain other tool for compiling java program.
- JVM comes along with both JDK and JRE and created when we execute Java program by giving java command.
- JVM provides platform-independent way of executing code.

Date
Page

7. How error is differ from exception. Write a java program to illustrates the use of try, catch, throw, throws and finally keyword.

⇒ Error are ~~the~~ generally can not be handled and usually refer catastrophic failure.
eg. running out of system resources.
It can not be handled using programming.

The error which can be catch and deal with it is called exception. It can be avoid by good programing technique
eg. ArithmeticException etc.

```
class Division {
    int x, y;
    Division (int p, int q) {
        x=p; y=q;
    }
    void result() throws ArithmeticException {
        try {
            if (y==0)
                throw new ArithmeticException();
            else
                system.out.println("ans = " + x,y);
        }
        catch (ArithmeticException e) {
            system.out.println("caught" + e);
        }
```

```
        finally
            System.out.println("This is finally Block");
    }
}
public class ExcepDemo {
    public static void main(String[] args) {
        Division d = new Division(4, 0);
        d.result();
    }
```

O/P.

caught java.lang.ArithmeticException
This in finally block.

8. Write a Java program to demonstrate the concept of user defined exception taking any example scenario.

Ans Acount.java

```
class NotSuffBlncException extends Exception {
    NotSuffBlncException(double b) {
        System.out.println("Unsufficient balance :=" + b);
    }
}
```

```
public class Account {
    double balance = 0;
    void deposite (double b) {
        balance += b;
        System.out.println("New balance = "+ balance);
    }
    void withdraw (double b) throws NotSuffBlncExceptn
        try {
            if  b> balance
                throw new NotSuffBlncException (b-balanu

            else {
                System.out.println("Withdrawable balance ="
                                    balance);
                balance -= b;
                System.out.println("new balance"+ balance);
            }
        }
}
```

UserDfnd Excp.java

```
public class UserDfnd Excp {
    public static void main (String [] args) {
        Account a = new Account ();
        a.deposite (50000);
        try {
            a.withdraw (40000);
        }
```

```
catch (NotSuffBlncException e) {
        System.out.println("caught" + e) }
}
try
    a.withdraw (30000);
catch( NotSuffBlncExcepnim e)
        system.out.println(" caught" + e);

}
}
```

Q18.

```
new balance = 50000
withdrawable balance = 50000.0
new balance = 10000.0
unsufficient balance = 20000.0
caught Userdfndexcep.NotsuffBlncExcepnion.
```

Here we can withdraw and deposite money in account. It throws NotSuffBlncExcepnion if withdrawa we try to withdraw amount more than the amount we have in account.

Date
Page

9. What is debugging ? Explain different debugging techniques with suitable example section of code.

⇒ Error of thinking while writing code which make our product weak and vulnurable is called bug. and the technique of removing bug is debugging.

Debugging Techniques :

- while, do-while & for loop does not terminate as long the controling boolean evaluates to true.
  Bug in loop always causes due to
  use of = in place of ==
  error in updating value.
  e.g.

```
for i=0; i>10; i++){
}
```

- we can debug by tracing variables.
  e.g. printing variable at every points.
- we can try by removing code and find exact place of bug.
- We can use comment either single line or double line
- we can use debugger for debugging.
  e.g. GDB debugger.

10. Explain each of the following with relevant example program:
How Java programming provides the basic object oriented features like encapsulation, abstraction, polymorphism, and inheritance.

⇒ Java provides encapsulation by the use of class. Encapsulation is the capsulating different type of variables & functions or any programming section in single unit. Java provides it in each code.

polymorphism can be achive simply using function or constructor overloading.

```
class Box {
    double width, height;
    Box() {
        width = height = 5;
    }
    Box (double l) {
        width = height = l;
    }
    Box (double l, double b) {
        width = l;
        height = b;
    }
    double area() {
        return width * height;
    }
}
```

```
public class BoxTester {
    public static void main (String [] args) {
        Box b1 = new Box();
        Box b2 = new Box(10);
        Box b3 = new Box (5,10);
        double a
        S.o.p("Area of box1"=+b1.area());
        S.o.p(" Area of box2 =" + b2.area());
        S.o.p ("Area of box 3 =" + b3.area());
    }
}
```

o/p

```
Area of box1 = 25.0
Area of box2 = 100.0
Area of box3 = 50.0
```

Inheritence is the extending new class using existing class.

eg.

```
class A {
    int i, j;
    void showij() {
        S.o.p ("i & j =" + i + j);
    }
    void setij (int p, int q) {
        i = p;
        j = q;
    }
}
```

Date
Page

```
class B extends A {
    int k;
    void showK() {
        S.o.p("K = " + k);
    }
    void setK(int p) {
        k = p;
    }
    void sum() {
        S.o.p("i+j+k = " + (i+j+k));
    }
}

class SimpleInh {
    public static void main(String[] args) {
        B obj = new B();
        obj.setIJ(10, 20);
        obj.setK(10);
        obj.showIJ();
        obj.showK();
        obj.sum();
    }
}
```

O/P. i & j = 10 20
k = 10
i+j+k = 40

In Java abstraction can be achieved by the use of abstract class.

eg.

```
abstract class First{
    abstract void callme();
}

class Second {
    void callme(){
        s.o.p(" This is callme function of abstract class");
    }
}
class AbstractDemo{
    p.s.v.m(String[] args){
        Second s = new Second();
        s.callme();
    }
}
```

o/p. This is callme function of abstract calss.

→ In Java complete abstraction is achived by using interface. It have only abstract methods.

Date
Page

11. What are the significance of using interface in Java? Explain with suitable example program.

⇒ significance of using interface in Java:-

- Member of interface can only have constants.
- It have only abstract method.
- It gives complete abstraction, and polymorphism.
- class implementing any interface must implement all the methods.
- Interface can be used to achieve the feature of multiple inheritance.

eg.

```
interface Area {
        final static double PI = 3.14;
        double compute ( double x, double y);
}

class Rectangle implements Area{
        public double compute (double x, double y)
                return x*y;
}

class Circle implements Area {
        public double compute (double x, double y).
                return PI*x*x;
}
```

Date
Page

```
class InterfaceTest {
    public static void main (String [] args) {
        Rectangle r = new Rectangle ();
        Circle c = new Circle (); Area a = r;
        s.o.p (" Area of Rectangle =" + a. compute (4
        a = c;
        s.o.p (" Area of circle =" + a. compute (10, 0))
    }
}
```

O/P. Area of Rectangle = 2000
Area of circle = 314

Date
Page

12. Write a program to implement following functionality of Bank Account. Assume necessary member variable required. (consider exception handling too)

class name : BankAccount

member methods:

public BankAccount (long accountNumber) : use parameterized constructor

public void withdraw (double amount) : Withdraw an amount (check balance before withdraw)

public double getBalance () : Return an available balance

public void transfer (double amount, BankAccount targetAccount) : Transfer amount to targetAccount.

Sol?

BankAccount.java

```
class NotSuffBlncExcp extends Exception {
    NotSuffBlncExcp ( double b ) {
        System.o.p.("NotSufficientBalance:" + b);
    }
}

public class BankAccount {
    double balance;
    long accountNumber;
    public BankAccount (long accountNumber) {
        this.accountNumber = accountNumber;
    }
```

2. (X)

```
public void withdraw (double amount) throws
NotSuffBlncExcp {
    if (balance < amount)
        throw new NotSuffBlncExcp (amount-balance);
    else {
        balance -= amount;
        S.o.p (" Withdrawn Amount :" + amount);
        S.o.p (" Account Number :" + accountNumber +
        "\n New Balance :" + balance);
    }
}

public double getBalance()
    return balance;
}

public void transfer ( double amount, BankAccount
targetAccount ) throws NotSuffBlncExcp {
    if (balance < amount)
        throw new NotSuffBlncExcp (amount - balance);
    else {
        balance -= amount;
        targetAccount.deposite (amount);
        S.o.p ("Transfered Amount :" + amount);
        S.o.p (" From :" + accountNumber +
        "\n To :" + target.accountNumber);
    }
}
```

```
public void deposite (double amount) {
    balance += amount;
    S.o.p (" Deposited Amount :" + amount );
    S.o.p. (" Account Number : " + accountNumber +
        "\n New Balance : " + balance));
}

public static void main (String [] args) {

    BankAccount user1 = new BankAccount( 200001 );
    BankAccount user2 = new BankAccount ( 300001 );
    user1. deposite (50000);
    try {
        user1. transfer (30000, user2);
    }
    catch ( NotSuffBlncExcp e) {
        S.o.p("caught " + e);
    }
    S.o.p (" Balance of user1 =" + user1. getBalance() );
    S.o.p (" Balance of user2 =" + user2. getBalance());

    try {
        user1. withdraw ( 20001);
    }
    catch ( Not Sufficient NotSuffBlncExcp e){
        S.o.p ("caught " + e);
    }
}
}
```

Date
Page

output

Deposited Amount : 50000.0
Account Number : 200001
New Balance : 50000.0
Deposited Amount : 30000.0
New Balance : 30000.0
Transfered Amount : 30000.0
From : 200001
To : 300001
Balance of user1 = 20000.0
Balance of user2 = 30000.0
NotSufficient Balance : 1.0
caught ns.bankaccount.NotSuffBlnc.Excp

13. Make a Java application with these features:

- An interface called RegularPolygon with two abstract methods: getNumSides ~~return 3~~ and getSideLength
- A class EquilateralTriangle that implements the interface, has getNumSides return 3 and getSideLength return an instance variable that is set by the constructor.
- A class Square return an ~~instance variable that is~~ that implements the interface, has getNumSides return 4 and getSideLength return an instance variable that is set by constructor.

13.1 Add a static totalSides method, that given a RegularPolygon[], returns the sum of the number of sides of all the elements.

13.2 Add two default methods:

- getPerimeter (n * length, where n is the number of sides)
- getInteriorAngle ((n-2)π/n in radians)

13.3 Make a few test cases.

Sol^n

RegularPolygon.java

```
public interface RegularPolygon {
    double PI = 3.1416;
    int getNumSides();
    double getSideLength();
    static int totalSides() {
        return 7;
    }
}
```

Date
Page

EquilateralTriangle.java

```
public class EquilateralTriangle implements RegularPolygon {
    double length;
    EquilateralTriangle(double l) {
        length = l;
    }
    @Override
    public int getNumSides() {
        return 3;
    }
    @Override
    public double getSideLength() {
        return length;
    }
    double getPerimeter() {
        return getNumSides() * length;
    }
    double getInteriorAngle() {
        return ((getNumSides() - 2) * PI) / getNumSides();
    }
}
```

Date
Page

Square.java

```
public class Square implements RegularPolygon {
    double length;
    Square(double l) {
        length = l;
    }
    @Override
    public int getNumSides() {
        return 4;
    }
    @Override
    public double getSideLength() {
        return length;
    }
    double getPerimeter() {
        return getNumSides() * length;
    }
    double getInteriorAngle() {
        return ((getNumSides() - 2) * PI) / getNumSides();
    }
}
```

Date
Page

PolygonTester.java

```
public class PolygonTester {
    public static void main(String [] args) {
        EquiliteralTriangle et = new EquiliteralTriangle(3.3);
        Square s = new Square(5.4);
        S.o.p("side of eq. triangle = " + et.getNumSides());
        S.o.p("side length of square = " + s.getSideLength());
        S.o.p("perimeter of square = " +
              s.getPerimeter());
        S.o.p("Interior angle of triangle = " +
              et.getInteriorAngle());
        S.o.p("Total num. of sides = " +
              RegularPolygon.totalSides());
    }
}
```

o/p

```
side of eq. triangle = 3
side length of square = 5.4
perimeter of square = 21.6
Interior angle of triangle = 1.0471975511965976
Total num. of sides = 7
```

Date
Page

1. Explain the seven key pillars of Asp.net.

→

(i) Asp.net is integrated with the .NET Framework

→ .NET framework is divided into an almost painstaking collection of functional parts, with tens of thousands of types. Each one of the thousands of classes in the .NET framework is grouped into a logical, hierarchical container called namespace.

(ii) Asp.net is compiled, not interpreted.

→ In first stage, the C# code is compiled into intermediate language called Microsoft Intermediate Language (MSIL) or just IL. The compiled file with it is an assembly. The second level of compilation happens just before the page is actually executed. At that point IL is compiled into low-level native machine code. This stage is known as just in time (JIT) compilation.

(iii) Asp.net is multilanguage

→ Many programmers will probably opt to use one language over another when one develop an application, that choice won't determine what can accomplish with web applications. That's because no matter what language, the code is compiled into IL.

(iv) Asp.net is hosted by the Common Language Runtime

→ perhaps the most important aspect of the ASP.NET engine is that it runs inside the runtime environment of the CLR. Some benefits of CLR are

automatic memory management, garbage collection, type safety, extensible metadata etc.

(v) Asp.net is object oriented

→ It provides a small set of objects; these objects are really just a thin layer over the raw details of HTTP and HTML. On the other hand, ASP.NET is truly object oriented.

(vi) Asp.net supports all Browsers.

→ It supports all browsers so don't need to be worry. Although we can retrieve info. about the client browser & its capabilities in Asp.net.

(vii) Asp.net is easy to deploy and configure.

→ One of the biggest headaches a web developer faces during a development cycle is deploying a completed application to a production server. Asp.net simplifies this process considerably.
Deploying an Asp.net app. is relatively simple.

Date
Page

2. Clearify your understanding and uses of ASP.NET file types and ASP.NET application directories.

=) ASP.NET file types :- →

(i) .aspx

→ These are asp.net web pages. They contain the user interface and, optionally the underlying application code. User request or navigate directly to one of these pages to start web application.

(ii) .ascx

→ These are asp.net user controls. It allows us to develop a small piece of user interface & reuse it in as many web forms.

(iii) web.config

→ This is the XML based config. file for asp.net app.

(iv) global.asax

→ This is global app. file. We can use this file to define global variables and react to global events.

(v) .cs

→ These are code-behind files that contain C# code. They allow us to separate the app. logic from the user interface of the web page.

Date
Page

ASP.NET application directories :-

(i) App_Browsers

→ contains .browser files that asp.net uses to identify the browsers that are using our application & determine their capabilities.

(ii) App_Code

→ contains source code files that are dynamically compiled.

(iii) App_GlobalResources

→ stores global resources that are accessible to every page in the web app.

(iv) APP_LocalResources

→ serve the same purpose as above, except these are accessible to a specific page only.

(v) APP_WebReferences

→ store ref. to web services, which is a remote code routines.

(vi) App_Data

→ stores data, including SQL server express database files.

(vii) App_Themes

→ store themes that are used to standardize and reuse formatti

(viii) Bin

→ Contains all the compiled .NET components (DLLs).

Date
Page

3. What are the server controls in ASP.net? Explain the life cycle of Asp.net ~~app~~ page.

=) ASP.net introduces a remarkable new model for creating web pages. We don't need to write our html. Controls are created and configured as objects. They run on the web server and they automatically provide their own HTML output.

Asp.net actually provides two sets of serverside controls that we can incorporate into our web forms.

(i). HTML server controls
- Generate their own interface
- Retain their state
- Fire server side events

(ii) web controls
- provides a rich user interface
- provide a consistent object model.
- tailor their o/p automatically
- provide high-level features.

Life cycle of Asp.net page:-

In the life cycle following stages involve:

(i) Page request

→ Page request occurs before the page life cycle begins. When a page is requested by a user, asp.net determines whether the page needs to be parsed and compile or whether a cached version of the page can be sent in response without running the page.

(ii) Start

→ page properties such as Request and Response are set.

(iii) Initialization

→ controls on the page are available and each control's uniqueID property is set.

(iv) Load

→ If the current request is a postback, control property are loaded with info. recovered from view and control state.

(v) postback event handling

→ If the request is postback, control event handlers are called. After that validate method of all validator controls is called.

(vi) Rendering

→ Before rendering, view state is saved for the page and all controls. During rendering state, page calls the Render method for each control, providing a text writer that writes the its o/p to OutputStream object of the page's Response property.

(vii) Unload

→ It is raised after the page has been fully rendered.

Q. What is connection string? Explain the architecture of ADO.NET.

⇒ When we create a Connection object, we need to supply a connection string. The connection string is series of name/value settings separated by semicolons (;). Although connection strings vary based on the RDBMS and provider we are using, few piece of info are almost always required:

→ The server where the database is located
→ The database we want to use
→ How the database should authenticate

ADO.NET architecture:-

ADO.NET is a .NET Framework's own data access technology. That consists of managed classes that allow .NET applications to connect to data sources, execute commands, and manage disconnected data. The small miracle of ado.net is that allows us to write more or less the same data access code in web application that we write for client-server desktop apps; or even single-user apps. That connect to a local database.

ADO.NET uses multilayered arch. That resolves around a few key concepts, such as Connection, Command, and Data Set objects.

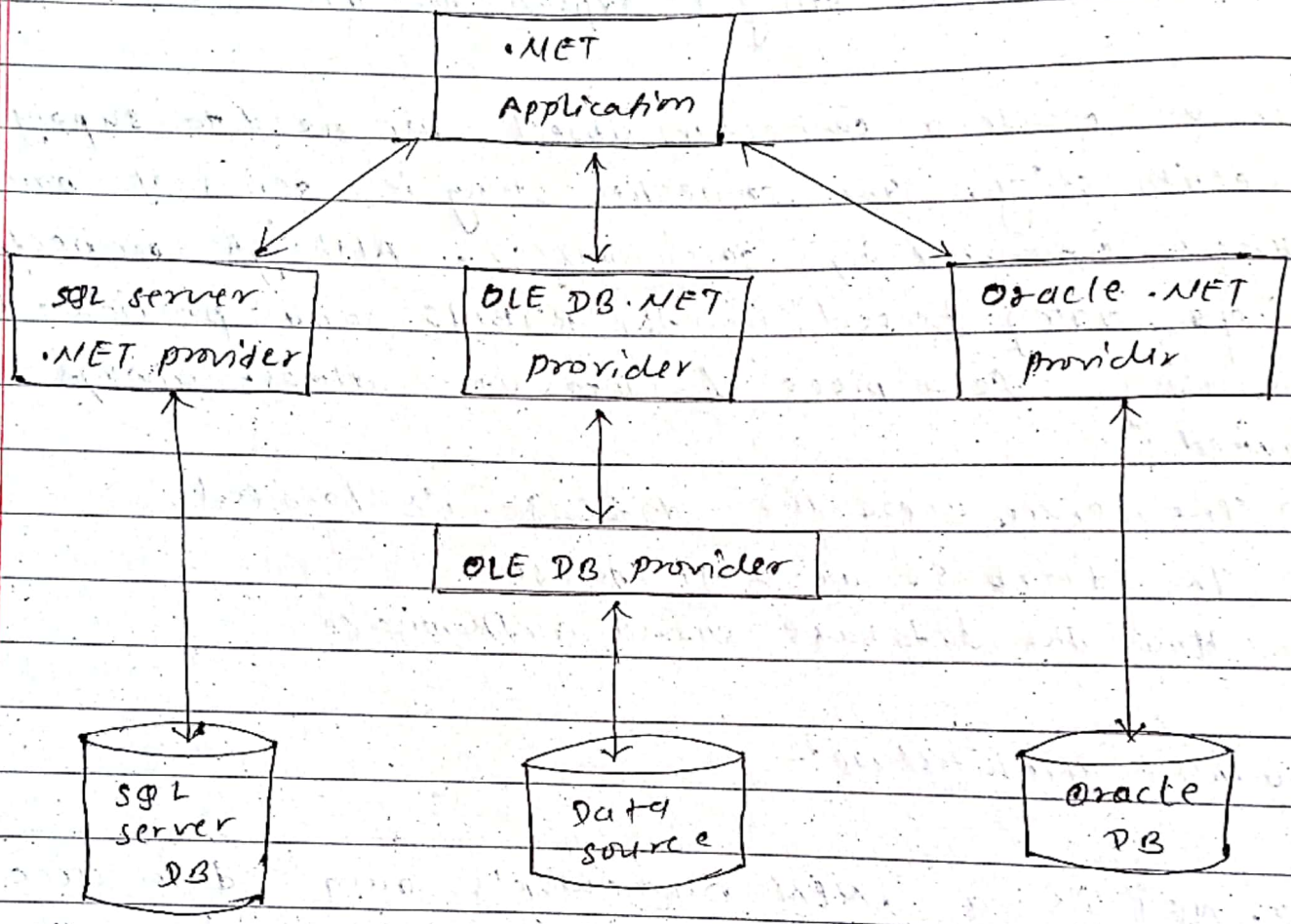

Fig. Architecture of ADO.NET.

Date
Page

5. Write a simple asp.net program in order to demonstrate the connectivity to the database server. (use connection string).

=> Once we chosen our connection string, managing the connection is easy, we simply use Open() and Close() methods.
We can use following code in Page.Load event handler to test a connection and write its status to a label.
First we need to import
System.Data.SqlClient namespace.

```
// Create the Connection object
string connectionString = WebConfigurationManager.
ConnectionStrings["DBConnectionString"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);

try {
  // Try to open the connection
  con.Open();
  lblInfo.Text = "<b>Server Version:</b>" +
                 con.ServerVersion;
  lblInfo.Text += "<br><b>Connection Is:</b>" +
                  con.State.ToString();
}
catch (Exception err){
  // Handle an error by displaying the information.
```

```
        lblInfo.Text = "Error reading the database." +
                       ex.Message;
    }
    finally {
        // Make sure the connection is properly closed.
        con.Close();
        lblInfo.Text += "<br><b>Now Connection is:</b> " +
                        con.State.ToString();
    }
```