Java Real Time Tools

by

Mr. RAGHU

Scanned with CamScanner

1. MAVEN

jar : To run any java project we must provide at least one jar to JDK(JVM). jar is a collection of .class files (or even it can contain any data like .java, .xml , .txt etc..)

All basic classes like String, Date, Thread, Exception, List etc.. are exist in rt.jar (runtime jar) given by Sun(Oracle). rt.jar will be set to project by JDK/JRE only. Programmer not required to set this jar.

Every jar is called as dependency. It means program depends on jars. Sometimes one jar may need another jar(child jar), this is called as dependency chain.

ex: Spring jars needs commons logging jar, hibernate jars needs slf4j,java-asst jar

Every jar directly or indirectly depends on rt.jar. Always parent jar version >= child jar version

Dependency Management :

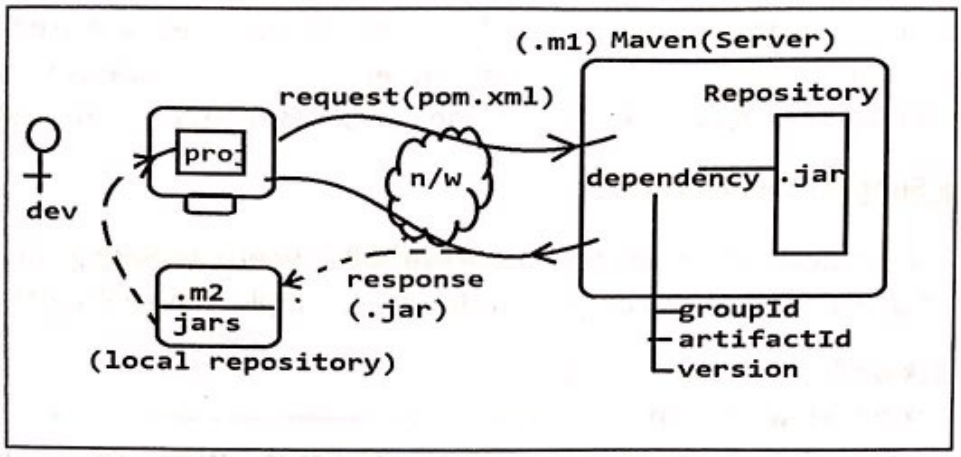
It is a concept of maintain jars of a project, maintain child jars and parent jars (dependency-chain), maintain proper version of all jars (dependency-version)

Library : It is collection of Jars.

Maven is a Tool (software) used to handle dependency-management as automated ie jars, chain jars with version management should be taken care by maven tool. It is developed by Apache. Maven is a Repository server which contains jars(dependencies) only in well-arranged format (provider based, jar name based, version based).

Every jar (depedency) contains 3 details mainly. Those are groupId (jar provider/company name) artifactId (jar name) & version (jar version). Maven maintains all jars as Library also called as Maven Library.

Scanned with CamScanner



Maven project created in local system should make request. Request should be in XML format. That is pom.xml (pom= project object model). Maven server takes pom.xml request and returns jars as response. All jars will be copied to local repository first ie ".m2" folder.

Every jar(dependency)should contain,

groupId (providerName)
artifactId (jar name)
version (jar version).

Format looks as,

```
<dependency>
 <groupId> ___ </groupId>
 <artifactId>___ </artifactId>
 <version> ____</version>
</dependency>
```

Maven Projects :

To create Maven Project we should choose one option given below.

1. Simple Project : Basic Application
2. ArchType Project : Like web-app, framework related etc...

For any Maven project 4 basic folder are provided. Those are,

- **src/main/java**
- **src/main/resource**
- **src/test/java**
- **src/test/resource**

Scanned with CamScanner

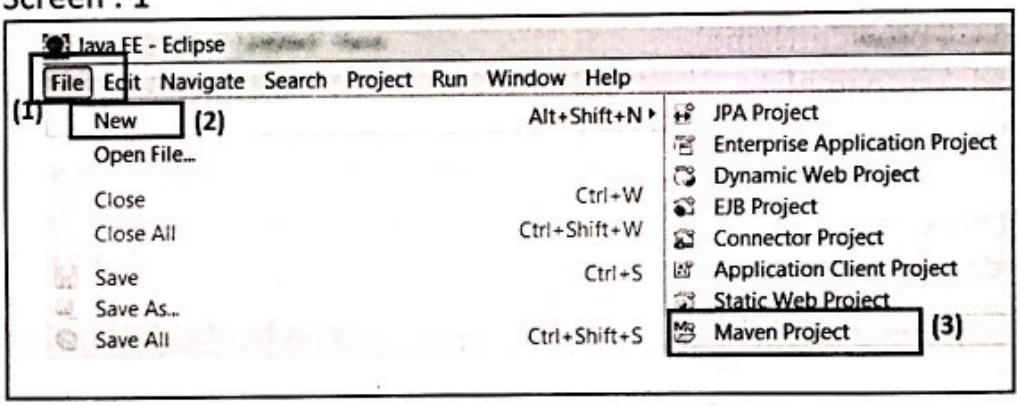
"java" folder is used to store only .java files, where "resources" are used to store non-java files like .properties, .xml, .txt, etc... "main" indicates files related to project developement. "test" indicates files related to UnitTesting.

Creating Simple Maven Project :

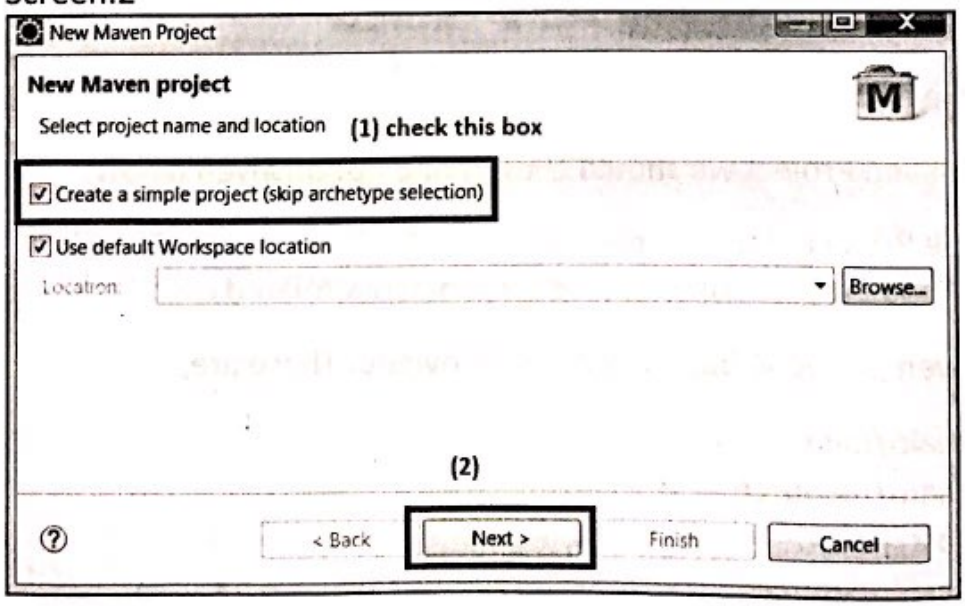
This type of projects are used in java to write JDBC,Hibernate,Spring core or any stand alone applications only. Steps to create Simple Maven Project:-

1. click on "File" menu
2. click on "New" option
3. Choose "Maven Project" option. If not exist goto "other", search using maven project word.
4. Screen looks as below. Here select checkbox "Create Simple Project".
5. on click next button, maven project needs groupId,artifactId,version details of our project.

Screen : 1

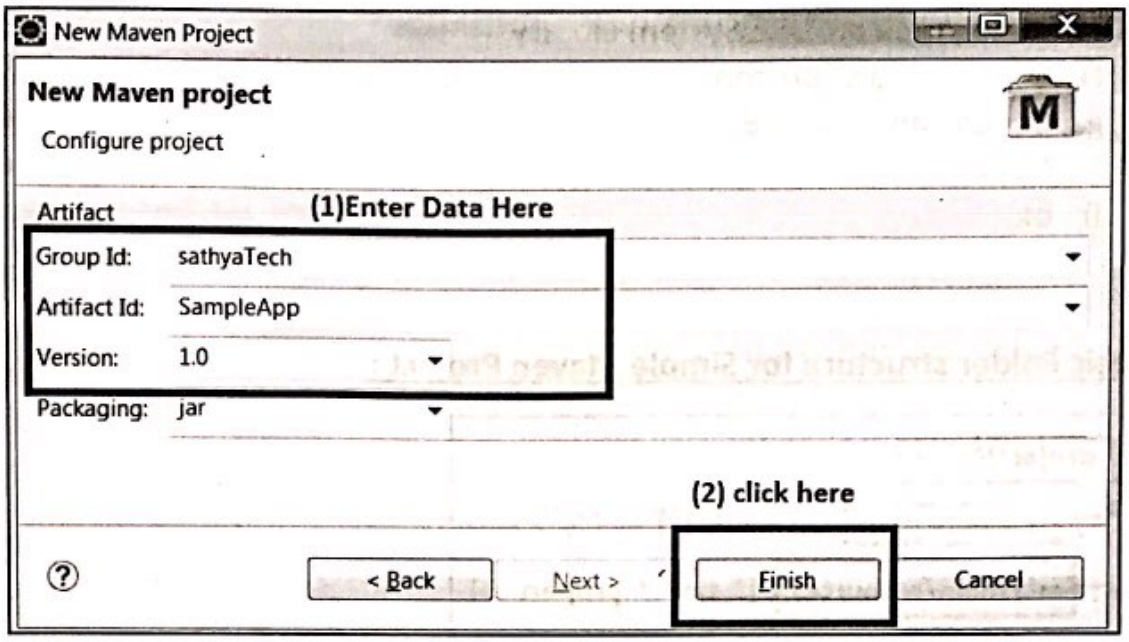


Screen:2



Scanned with CamScanner

Screen:3



At last our project also be converted/exported as .jar (java archive), .war(web archive) or .ear (enterprise archive). Here version can contain numbers, characters and symbols. (Screen looks as below)

Setup JDK/JRE in Eclipse :

A. Window menu
B. preferences option
C. Search with "installed JRE"
D. choose Installed JRE under Java
E. click on "Add" Button to add another
F. Select "Standard VM "
G. next
H. Click on "Directory/Browse"
I. select location (ex: C:/Program Files/Java/jdk1.8)
J. finish/ok
K. finish/ok

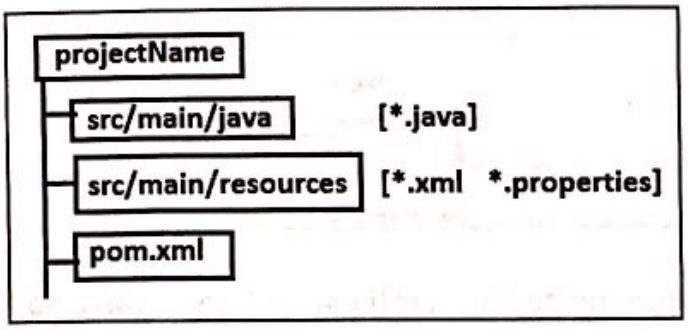
Now Modify JRE Version to Maven

a) Right click on maven project
b) Build path option
c) Click on configure Build path..

Scanned with CamScanner

d) Choose Library Tab
e) Select/click on "JRE System Library"
f) Click on "Edit" Button
g) Choose any one JRE
h) Finish
i) ok

Basic Folder structure for Simple Maven Project :



✓ Open pom.xml file in XML View mode. In this root tag is <project>
✓ Under this tag, add one tag
<dependencies> </dependencies>

Then file looks like

```
<project ....>
 <modelVersion>4.0.0</modelVersion>
 <groupId> ------- </groupId>
 <artifactId> ---- </artifactId>
 <version> ------- </version>
 <dependencies>



 </dependencies>
</project>
```

Here, we must specify jar details under <dependencies> tag example : for spring core programming

<dependency>

Scanned with CamScanner

```
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>4.3.2.RELEASE</version>
</dependency>
```

For mysql-connector jar :

```
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.26</version>
</dependency>
```

*** User Library ***

If any Jar not exist in Maven or unable get from Maven Server (ex: created by One company or by programmer) then we can add jar to project using this concept. User Library has two steps,

1. Create Library in Workspace
2. Add Library to project.

1. Create Library in Workspace

i. Window menu
ii. Preference option
iii. Search with "User Library" word
iv. Select User Library under Java
v. Click on New button
vi. Enter Library Name (ex: MyLibs)
vii. Finish/ok
viii. Click on Library Name
ix. Click on Add External Jars Button
x. Select Jars required from folders
xi. Finish
xii. ok

2. Add Library to project:

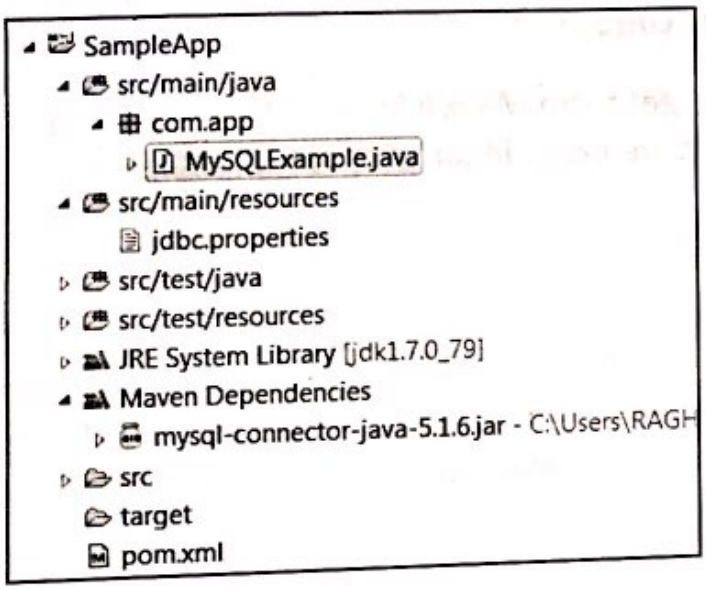
I. Right click on project

Scanned with CamScanner

II. Choose "Build path"
III. Configure Build path ...
IV. Click on Library Tab.
V. Click on "Add Library" Button
VI. Select "User Library" option
VII. Next button
VIII. Choose "MyLibs"
IX. Click on Finish and ok.

EXAMPLE MAVEN APPLICATIONS

1. JDBC USING MYSQL TYPE-4 DRIVER CLASS WITH PROPERTIES FILE



pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>sathyaTech</groupId>
        <artifactId>SampleApp</artifactId>
        <version>1.0</version>
        <dependencies>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <version>5.1.6</version>
                </dependency>
        </dependencies>
```


Scanned with CamScanner

```
</project>
```

Step#1 Right click on src/main/resources folder , choose here new->other option. Search with "file" and choose same. Enter File name as "jdbc.properties".

```
# JDBC MySQL Connection properties

jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/test
jdbc.user=root
jdbc.password=root
```

Step#2 Create one java Class "MySQLExample.java". Here defined 3 static methods to load properties file, to get DB Connection and to select data from employee table.

```
package com.app;

import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

/**
 * @author RAGHU
 * @version JDK 1.7
 */

public class MySQLExample {

	/**
	 * 1. Load Connection Details from Properties file
	 */
	public static Properties loadProperties(){
		Properties props=null;
		try {
			ClassLoader loader = Thread.currentThread().
```


Scanned with CamScanner

```
                                        getContextClassLoader();
            InputStream resourceStream = loader.
            getResourceAsStream("jdbc.properties");
            props=new Properties();
            props.load(resourceStream);

        } catch (Exception e) {
            e.printStackTrace();
        }
        return props;
}

/**
 * 2. Get Connection
 *  @throws Exception
 */
public static Connection getConnection() throws Exception{

            Properties props=loadProperties();
            Class.forName(props.getProperty("jdbc.driver"));
            Connection con = DriverManager.getConnection(
                        props.getProperty("jdbc.url"),
                        props.getProperty("jdbc.user"),
                        props.getProperty("jdbc.password"));
            return con;
}
/**
 * 3. Select Data from employee table
 * @throws Exception
 */
public static void selectEmployee() throws Exception{

        Statement st = getConnection().createStatement();
        ResultSet rs = st.executeQuery("select * from employee");
        while (rs.next())    {
                    System.out.print(rs.getInt(1)+",");
                    System.out.print(rs.getString(2)+",");
                    System.out.println(rs.getDouble(3));
            }
```


Scanned with CamScanner

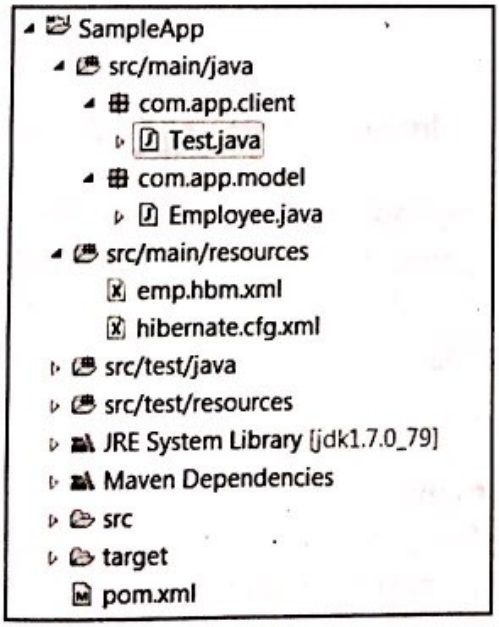
```
    }
    public static void main(String[] args) {
        try {
            selectEmployee();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Create Table & insert Records:

```
employee  CREATE TABLE employee (
    empId int(11) NOT NULL,
    empName varchar(255) DEFAULT NULL,
    empSal double NOT NULL,

  ) ;
insert into employee values(10,'AA',55.23);
insert into employee values(11,'BB',66.24);
```

2. HIBERNATE EXAMPLE APPLICATION USING MYSQL DATABASE.

Folder Structure:-



pom.xml file

Scanned with CamScanner

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>sathyaTech</groupId>
        <artifactId>SampleApp</artifactId>
        <version>1.0</version>
        <dependencies>
          <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>4.3.5.Final</version>
      </dependency>
      <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.6</version>
      </dependency>
     </dependencies>
</project>
```

Model class : Employee.java

```
package com.app.model;

public class Employee {

        private int empId;
        private String empName;
        private double empSal;

        //alt+shift+O (De-select All->OK)
        public Employee() {
        }
        //alt+Shift+S , R (select all) ->OK
        public int getEmpId() {
                return empId;
        }
        public void setEmpId(int empId) {
```


Scanned with CamScanner

```
		this.empId = empId;
	}
	public String getEmpName() {
		return empName;
	}
	public void setEmpName(String empName) {
		this.empName = empName;
	}
	public double getEmpSal() {
		return empSal;
	}

	public void setEmpSal(double empSal) {
		this.empSal = empSal;
	}
	//alt+shift+s,s ->ok
	@Override
	public String toString() {
		return "Employee [empId=" + empId + ", empName=" + empName
+ ", empSal=" + empSal + "]";
	}



}
```

Mapping File : emp.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
	<class name="com.app.model.Employee" table="EMPLOYEE">
		<id name="empId" column="eid"/>
		<property name="empName" column="ename"/>
		<property name="empSal" column="esal"/>
	</class>
</hibernate-mapping>
```


Scanned with CamScanner

configuration file hibernate.cfg.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory>
                <property name="hibernate.connection.driver_class">
                        com.mysql.jdbc.Driver </property>
<property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/test
</property>
 <property name="hibernate.connection.user">root</property>
 <property name="hibernate.connection.password">root</property>

 <property name="hibernate.show_sql">true </property>
<property name="hibernate.dialet">org.hibernate.dialect.MySQLDialect
</property>
<property name="hibernate.hbm2ddl.auto">create</property>
<mapping resource="emp.hbm.xml" />
 </session-factory>
</hibernate-configuration>
```

Test class :

```
package com.app.client;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import com.app.model.Employee;

public class Test {

        public static void main(String[] args) {
                Configuration cfg = new Configuration();
                cfg.configure(); //loads hibernate.cfg.xml file
```


Scanned with CamScanner

```
        /*SessionFactory sf = cfg.buildSessionFactory();*/
        SessionFactory sf = cfg.buildSessionFactory(
                        new StandardServiceRegistryBuilder().
                        applySettings(cfg.getProperties()).build());

        Session ses = sf.openSession();
        Transaction tx = ses.beginTransaction();

        Employee emp=new Employee();
        emp.setEmpId(100);
        emp.setEmpName("ABC");
        emp.setEmpSal(12.36);

        ses.save(emp);

        tx.commit();
        ses.close();

        System.out.println("Record inserted..");
    }
}
```

3. SPRING CORE EXAMPLE :

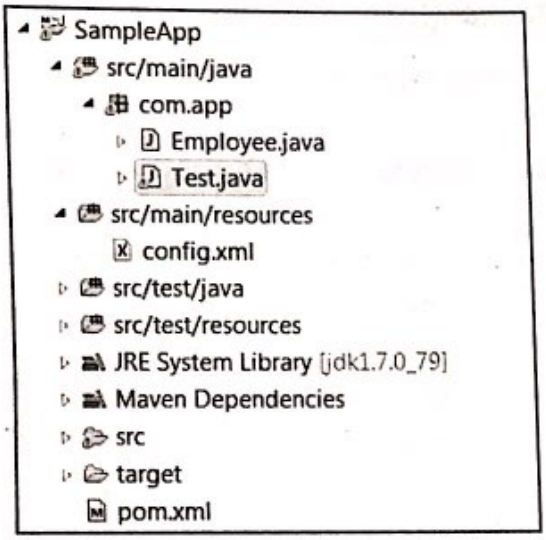
pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>sathyaTech</groupId>
    <artifactId>SampleApp</artifactId>
    <version>1.0</version>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
```


Scanned with CamScanner

```
            <artifactId>spring-context</artifactId>
            <version>4.3.4.RELEASE</version>
        </dependency> 5.1.8.Rel
    </dependencies>
</project>
```

Folder Structure :



Spring Bean:

```
package com.app;

public class Employee {

    private int empId;
    private String empName;
    private double empSal;

    public int getEmpId() {
        return empId;
    }
    public void setEmpId(int empId) {
        this.empId = empId;
    }
    public String getEmpName() {
        return empName;
    }
    public void setEmpName(String empName) {
        this.empName = empName;
```

Scanned with CamScanner

```
	}
	public double getEmpSal() {
		return empSal;
	}
	public void setEmpSal(double empSal) {
		this.empSal = empSal;
	}
	@Override
	public String toString() {
		return "Employee [empId=" + empId + ", empName=" + empName
+ ", empSal=" + empSal + "]";
	}

}
```

Spring Configuration File (XML File) : config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
  ">

	<bean class="com.app.Employee" name="empObj">
		<property name="empId" value="100"/>
		<property name="empName" value="ABC"/>
		<property name="empSal" value="2.36"/>

	</bean>

</beans>
```

Test class:

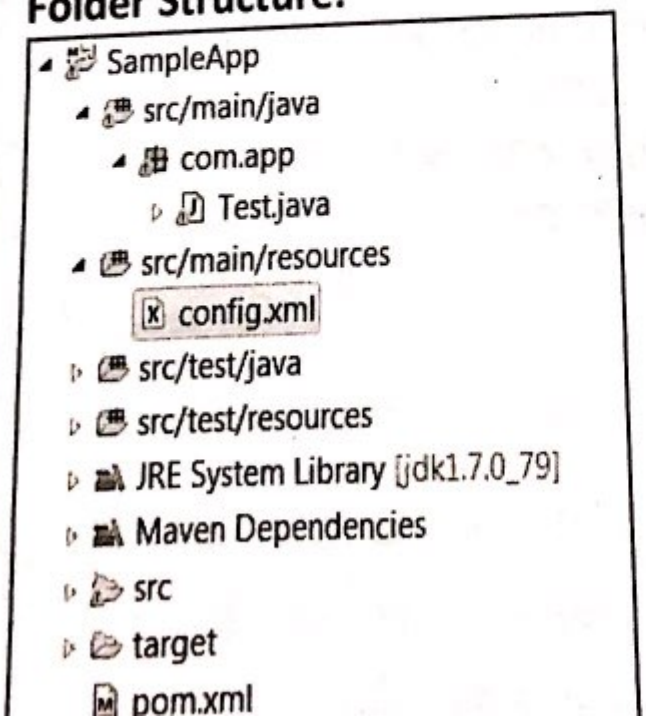
```
package com.app;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
```


Scanned with CamScanner

```
* @author RAGHU
* @version JDK 1.7
*/

public class Test {

        public static void main(String[] args) {
                ApplicationContext context=new
ClassPathXmlApplicationContext("config.xml");
                Object ob=context.getBean("empObj");
                if(ob instanceof Employee){
                        Employee emp=(Employee) ob;
                        System.out.println(emp);
                }else{
                        System.out.println("object is not employee type");
                }

        }
}
```

4. SPRING JDBC EXAMPLE :

Folder Structure:



Scanned with CamScanner

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>sathyaTech</groupId>
        <artifactId>SampleApp</artifactId>
        <version>1.0</version>
        <dependencies>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>4.3.4.RELEASE</version>
                </dependency>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-jdbc</artifactId>
                        <version>4.3.4.RELEASE</version>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <version>5.1.19</version>
                </dependency>
        </dependencies>
</project>
```

Spring Configuration File (XML File): config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:p="http://www.springframework.org/schema/p"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
 ">

        <bean
class="org.springframework.jdbc.datasource.DriverManagerDataSource"
                name="dataSourceObj"
```


Scanned with CamScanner

```
            p:driverClassName="com.mysql.jdbc.Driver"
            p:url="jdbc:mysql://localhost:3306/test"
            p:username="root"
            p:password="root"
        />

        <bean class="org.springframework.jdbc.core.JdbcTemplate"
name="jtObj"
            p:dataSource-ref="dataSourceObj"
        />

</beans>
```

Test class:

```
package com.app;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;

/**
 * @author RAGHU
 * @version JDK 1.7
 */

public class Test {

        public static void main(String[] args) {
            ApplicationContext context=new
ClassPathXmlApplicationContext("config.xml");
            JdbcTemplate jt=(JdbcTemplate) context.getBean("jtObj");
            String sql="insert into employee values(?,?,?)";
            int count=jt.update(sql, 20,"ABC",36.36);
            System.out.println(count);


        }
}
```


Scanned with CamScanner

2. NEED OF LOG4J

BUG/Problem:- After Application is developed then it will be handover to End Client/Customer. At the time of Running Application some problems may occur while processing a request. In programming concept it is called as Exception.

These problems try to identify at the time of development to avoid few. Some will be handled once they are occurred. Those are analyzed and solved by developer. Give priority to Avoid problems if not handle the problems.

To avoid problems concepts are : 1. Validations 2. Standard Coding

To handle the problems concepts are: 1. Error pages **2.Log4J**

1. Validations : At UI page like Register, Data Edit, Login, Feedback form, Comment etc.. at these levels we must use Java Script or equal Programming to avoid invalid input. Example:-

Register Page (JSP)
ID [_____] Only numarics [0-9]
Name [_____] only characters [a-z]
Email [_____] Pattern __@__.__ format

Write Scripting code for above checking.

---

2. Standard Coding : If we use coding standards we can reduce [30-40]% problems in application.

Examples : -

i) String s=(...) input is provided at run time
int len=s.length();
above code is converted to...
String s=....;
if(s!=null) int len=s.length();

Scanned with CamScanner

```
    else sysout("No Valid Data");
```

ii) **int arr[]={....};**
sysout(arr[4]);
may throw ArrayIndexOutOfBounds Exception above code is converted to...

```
int arr[]={....};
if(arr.length > 4){
 sysout(arr[4]);
}else {
 sysout("Array Size is less");
}
```

iii) **Object ob=ses.get(T.class,id);** (T=className provided at runtime)
Address a=(Address)ob;
it may throw ClassCastException to avoid this

```
if(ob instanceof Address){
 Address a=(Address)ob;
} else {  sysout("ob is not address");    }
```

iv) List<Employee> has Employee objects read 1st index employee and find that employee object empName length

code

```
List<Employee> empList=....
Employee e=list.get(0);
String empName=e.getEmpName();
int len=empName.length();
```

above code can be written as.....

```
if(empList!=null && empList.size>0 && empList.get(0)!=null ){
  Employee e=list.get(0);
  if(e!=null && e.getEmpName()!=null ) {
    int len=e.getEmpName().length();
  }
}
```


Scanned with CamScanner

3. Error Pages in Web Applications:- If server has encountered any problem then it will throw An equal HTTP Error. Example 404-Page Not Found, 500-Internal server Error, 400-Syntax Error, 405-Method Not Allowed, 401-UnAuthorized etc...

To show one simple message to end user we must define JSP/HTML page and make them as isErrorPage="true" and configure in **web.xml** as

```
<error-page>
 <error-code>404</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
 <exception-type>java.lang.NullPoniterException</exception-type>
 <location>/showError.jsp</location>
</error-page>
```

Create on "error.jsp" ex:

```
<%@ page ... isErrorPage="true"%>
<html><body> Welcome to Error Page ..Wait 5 sec to redirect to Home Page..
<%response.setHeader("Refresh", "5;index.jsp"); %> </body> </html>
```

Log4J

1. It is a Tracing or Logging Tool used in Specially in Production Environment. It is used to find success messages, information, warnings, errors in application while using it.
2. By Default any Message/Error will be printed on Console, which is temporary location, it can show only few lines like last 100 lines.
3. To see all problems from beginning to till date use Log4j concept.

Scanned with CamScanner

4. Log4J can write problems/error to File(.log), Database, Email, Network etc..
5. Log4J provides Error Details like Exception type, class and method with line number it occured, Date and time also other information ..
6. Log4J also supports writing errors to Console also.
7. Using System.out.println prints only on console, but not to file or any other memory.

Log4J has 3 components:

Layout	Appender	Logger

1. Logger (LOG) Object: This object must be created inside the class as a instance variable. It is used to enable Log4J service to current class. If this object is not created in the class then Log4J concpet will not applicable(will not work)for that class It has 5 priority methods with names (along with order)

Order	method	Name
1	debug(obj)	DEBUG
2	info(obj)	INFO
3	warn(obj)	WARN
4	error(obj)	ERROR
5	fatal(obj)	FATAL
-NA-	-NA-	OFF

NA : Not Applicable

a. debug(msg) : It prints a message with data. It is used to print a final result of process. Like EmpId after saved is : 2362.
b. info(msg) : It is used to print a simple message. Like process state-I done, if block end. Email sent etc..
c. warn(msg): It is used to print warning messages. Like Collection is not with Generic, local variable not used, resource not closed etc...
d. error(msg): It is used to print Exceptions like NullPointer, ArrayIndex, SQLException etc..

Scanned with CamScanner

e. fatal(mgs) : It indicates very high level problem. Like Server/DB Down, Connection timeout, Network broken, Class Not Found etc...

OFF is used to disable Log4J concept in application. Log4J code need to be deleted.

---

2. Appender : It will provide details for "Where to print Message?". Means in what type of memories, messages must be stored. To write Logger Statements to

i.	File	use	FileAppender
ii.	Database	use	JdbcAppender
iii.	Email	use	SmtpAppender
iv.	Console	use	ConsoleAppender
v.	Network	use	Ftp(Telnet)Appender

In one Application we can use more than one appender also.

---

3. Layout : It provide the format of message to be printed. Possible layouts are:

i. Simple Layout : Print message as it is
ii. HTML Layout : Print message in HTML format(<html><body>....)
iii. XML Layout: Print message in XML format (<Errors><Type>..<Message>..)
iv. Pattern Layout : Prints messages in given pattern. example pattern: Date-Time / Line Number : Class-method :- Message

In development mostly used Appender is FileAppender and Layout is Pattern Layout.

Log4J Example Programming:-

Step#1 Create Maven Project

I. Click on File Menu
II. Choose "new" option

Scanned with CamScanner

III. Choose Maven Project
IV. Select "Create Simple Project" checkbox
V. Click on "Next" Button
VI. Enter GroupId, ArtifactId, version details
VII. Click on "finish" button

Step#2 Specify Log4J dependency in pom.xml

```
<dependency>
 <groupId>log4j</groupId>
 <artifactId>log4j</artifactId>
 <version>1.2.17</version>
</dependency>
```

Step#3 Create one class under "src/main/java" and write Log4J code.

ex: code

```
package com.app;
import org.apache.log4j.Appender;
import org.apache.log4j.ConsoleAppender;
import org.apache.log4j.Layout;
import org.apache.log4j.Logger;
import org.apache.log4j.SimpleLayout;

public class Product {
	private static Logger log=Logger.getLogger(Product.class);
	public static void main(String[] args) {
		Layout layout=new SimpleLayout();
		Appender ap=new ConsoleAppender(layout);
		log.addAppender(ap);
		log.info("Hello");
	}
}
```

Run application "ctrl+F11" to see output.

Note:

Scanned with CamScanner

1. To Create "Logger" class object to current class use static factory method getLogger(_.class). It returns Logger Object and enable the Log4J to current class.

2. Create a Layout(Abstract Class) object using it's implemented classes, list given below

a. SimplLayout (C)
b. XMLLayout (C)
c. HTMLLayout(C)
d. PatternLayout(C)

3. Create Appender(I) object using it's implemented classes and also provide layout details to Appender object.

a. ConsoleAppender (C)
b. FileAppender (C)
c. JdbcAppender (C)
d. SmtpAppender (C)

4. At last Appender object must be added to Logger Object then only it is considered, by using log.addAppender() method.

5. If No appender is provided to Logger object then Log4J throws WARNING as log4j:WARN No appenders could be found for logger (com.app.Product).

Layouts: Layout Defines Format of Message to be displayed on UI/DB/Console. Possible Layout classes are listed below with description.

1. SimpleLayout : This class prints message as it is provided in log methods.

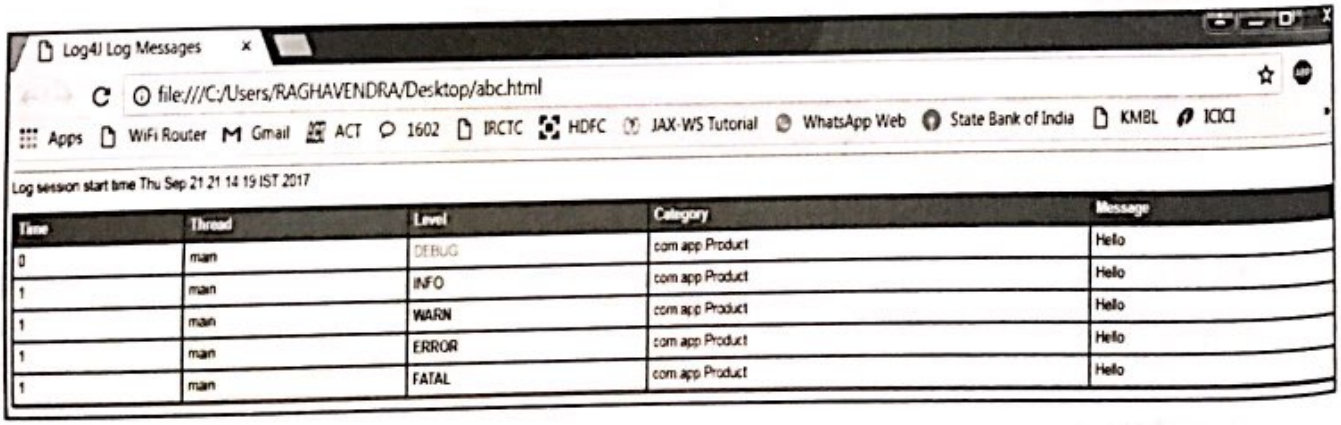
2. HTMLLayout : This class provides message in HTML format. To see full output of this. Copy output to .html file

Example Code:

Scanned with CamScanner

```
package com.app;
import org.apache.log4j.Appender;
import org.apache.log4j.ConsoleAppender;
import org.apache.log4j.HTMLLayout;
import org.apache.log4j.Layout;
import org.apache.log4j.Logger;


public class Product {
        private static Logger log=Logger.getLogger(Product.class);
        public static void main(String[] args) {
                Layout layout=new HTMLLayout();
                Appender ap=new ConsoleAppender(layout);
                log.addAppender(ap);

                log.debug("Hello");
                log.info("Hello");
                log.warn("Hello");
                log.error("Hello");
                log.fatal("Hello");

        }
}
```

Example: screen



3. XMLLayout: It prints log messages in XML format. In above code modify Layout as XMLLayout and run as same.

Example Code:

Scanned with CamScanner

```
package com.app;

import org.apache.log4j.Appender;
import org.apache.log4j.ConsoleAppender;
import org.apache.log4j.Layout;
import org.apache.log4j.Logger;
import org.apache.log4j.xml.XMLLayout;


public class Product {
	private static Logger log=Logger.getLogger(Product.class);
	public static void main(String[] args) {
		Layout layout=new XMLLayout();
		Appender ap=new ConsoleAppender(layout);
		log.addAppender(ap);

		log.debug("Hello");
		log.info("Hello");
		log.warn("Hello");
		log.error("Hello");
		log.fatal("Hello");

	}
}
```

Output Looks as :

```
<log4j:event logger="com.app.Product" timestamp="1506008834191"
level="DEBUG" thread="main">
<log4j:message><![CDATA[Hello]]></log4j:message>
</log4j:event>

<log4j:event logger="com.app.Product" timestamp="1506008834192"
level="INFO" thread="main">
<log4j:message><![CDATA[Hello]]></log4j:message>
</log4j:event>

<log4j:event logger="com.app.Product" timestamp="1506008834192"
level="WARN" thread="main">
<log4j:message><![CDATA[Hello]]></log4j:message>
```


Scanned with CamScanner

```
</log4j:event>

<log4j:event logger="com.app.Product" timestamp="1506008834192"
level="ERROR" thread="main">
<log4j:message><![CDATA[Hello]]></log4j:message>
</log4j:event>

<log4j:event logger="com.app.Product" timestamp="1506008834192"
level="FATAL" thread="main">
<log4j:message><![CDATA[Hello]]></log4j:message>
</log4j:event>
```

4. PatternLayout : This class provides output pattern for a message that contains date, time, class, method, line number, thread name, message etc..

Observe Some example patterns given below, for complete details click on below link

(https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html)

a)Date&Time pattern

- %d = date and time

examples:

a) %d
b) %d {dd-MMM-yy hh:mm:ss SSS}
c) %d {dd-MM-yy hh:mm:ss SSS}
d) %d {HH:mm:ss}
e) Here meaning of every word used
f) in date pattern is,
g) dd = date
h) MMM= Month Name
i) MM = Month number
j) yy = Year last two digitis
k) yyyy = Year in 4 digits
l) hh = Hours in 12 format

Scanned with CamScanner

m) HH = Hours in 24 format
n) mm = Minutes
o) ss = Secs
p) SSS = MillSecs

- %C = Class Name
- %M = Method Name
- %m = Message
- %p = Priority method name(DEBUG,INFO..)
- %L = Line Number
- %l = Line number with Link
- %n = New Line(next line)
- %r = time in milli sec.
- %% = To print one '%' symbol.
- we can also use symbols like - [] , /

One Example Pattern with all matching "%d-%C[%M] : {%p}=%m<%L> %n "

Appender:- An appender provides the destination of log message. Here example appenders are

a) ConsoleAppender
b) FileAppender
c) JdbcAppender
d) SmtpAppender

Example of FileAppender:-

```
package com.app;

import org.apache.log4j.Appender;
import org.apache.log4j.FileAppender;
import org.apache.log4j.HTMLLayout;
import org.apache.log4j.Layout;
import org.apache.log4j.Logger;
```


Scanned with CamScanner

```
public class Product {
        private static Logger log=Logger.getLogger(Product.class);
        public static void main(String[] s)       throws Exception {
                Layout layout=new HTMLLayout();
                Appender ap=new FileAppender(layout,"myapp.log");
                log.addAppender(ap);

                log.debug("Hello");
                log.info("Hello");
                log.warn("Hello");
                log.error("Hello");
                log.fatal("Hello");
        }
}
```

log4j.properties

log4j.properties file: This file is used to specify all the configuration details of Log4J. Especially like Appender Details and Layout Details with Patterns and also root Logger details. This File contains details in key=value format (.properties). Data will be shown in below order like

1. rootLogger
2. appenders
3. layouts

a) In this file (log4j.properties) use symbol '#' to indicates comments.
b) We can specify multiple appenders in log4j.properties file Like 2 File Appenders, one JdbcAppender, One SmtpAppender etc..
c) Every Appender must be connected with layout
d) Appender name must be define before use at rootLogger level.
e) Appender name can be anything ex: abc,hello,sysout,file,db,email etc..
f) log4j.properties file must be created under src folder (normal project) or src/main/resource folder (maven project).
g) Make rootLogger = OFF to disable Log4J completely without deleting code in any class or properties file

Scanned with CamScanner

h) log4j.properties file will be auto detected by Log4J tool. No special coding is required.

Example #1 Console Appender

Create one properties file under "src/main/resources" with name "log4j.properties"

I. Right click on "src/main/resources"
II. Choose "new"
III. Select "other.."
IV. Search and select "file" for option
V. Click on next
VI. Enter file name and click finish.

Example : log4j.properties (code)

```
# Root logger details
log4j.rootLogger=DEBUG,stdout

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd} %p %c:%L - %m%n
```

Test class:-

```
package com.app;
import org.apache.log4j.Logger;
public class Product {
        private static Logger log=Logger.getLogger(Product.class);
          public static void main(String[] srs){
                log.debug("Hello");
                log.info("Hello");
                log.warn("Hello");
```


Scanned with CamScanner

```
            log.error("Hello");
            log.fatal("Hello");

        }
}
```

output:

```
2017-09-21 21:47:59 DEBUG com.app.Product:6 - Hello
2017-09-21 21:47:59 INFO com.app.Product:7 - Hello
2017-09-21 21:47:59 WARN com.app.Product:8 - Hello
2017-09-21 21:47:59 ERROR com.app.Product:9 - Hello
2017-09-21 21:47:59 FATAL com.app.Product:10 - Hello
```

Using of Logger methods:

1. debug() : to specify a message with value (message with result after operation)
2. info() : to indicate current state (block end, method started , service finished, email sent etc..)
3. warn() : to provide warning message (Variable not used, List has no generic, connection not closed)..
4. error/fatal : used in catch blocks. Here error - used for unchecked exception and fatal - checked exception in general

example :- spring controller

```
package com.app.controller;
@Controller
public class HomeController {
        private static Logger log=Logger.getLogger(HomeController.class);

        @RequestMapping("/url")
        public String saveEmp(...){
                log.info("Save Method started");
                try{
                        log.info("Before save employee..");
```


Scanned with CamScanner

```
                int empId=service.save(emp);
                log.debug("emp saved with id :"+empId);
            }catch(Exception e){
                log.error(e); // log.fatal(e);
            }
            log.warn("EMPID not used in program");
            log.info("save service is at end");
            return "Home";
        }
}
```

Log4j Examples

pom.xml (to run all below examples)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>sathyatech</groupId>
        <artifactId>log4jexample</artifactId>
        <version>1.0</version>
        <dependencies>
            <dependency>
                <groupId>log4j</groupId>
                <artifactId>log4j</artifactId>
                <version>1.2.17</version>
            </dependency>
            <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>5.1.6</version>
            </dependency>
            <dependency>
                    <groupId>javax.mail</groupId>
                    <artifactId>mail</artifactId>
                <version>1.4</version>
            </dependency>
    </dependencies>
```

Scanned with CamScanner

```
</project>
```

Example #1 Console Appender Example

Step#1 : Define one Test class with example code with log methods.

```
package com.app;
import org.apache.log4j.Logger;
public class Product {
        private static Logger log=Logger.getLogger(Product.class);
          public static void main(String[] srs){
                  log.debug("Hello");
                  log.info("Hello");
                  log.warn("Hello");
                  log.error("Hello");
                  log.fatal("Hello");

        }
}
```

Step#2 Create log4j.properties file with below code

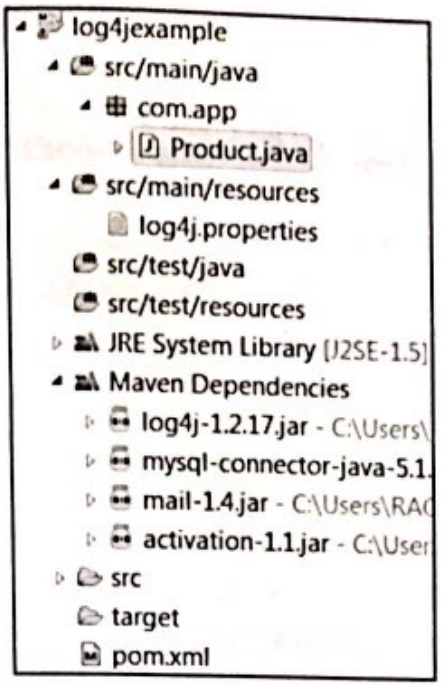
log4j.propertie:

```
# Root logger option
log4j.rootLogger=DEBUG, stdout
# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd } %p
%c:%L - %m%n
```

Step#3 Run above Test class Product.java (ctrl+F11) to see output.

Scanned with CamScanner

Folder Structure:-



Example#2 File Appender Example

Step#1 : Define one Test class with example code with log methods.

```
package com.app;
import org.apache.log4j.Logger;
public class Product {
        private static Logger log=Logger.getLogger(Product.class);
          public static void main(String[] srs){
                  log.debug("Hello");
                  log.info("Hello");
                  log.warn("Hello");
                  log.error("Hello");
                  log.fatal("Hello");

        }
}
```

Scanned with CamScanner

Step#2 Create log4j.properties file with below code

log4j.properties

```
# Root logger option
log4j.rootLogger=DEBUG, file
# Redirect log messages to a log file
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=logs/myapp.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}
%p %c:%L - %m%n
```

Here logs folder Created under Project (F5) Refresh.

Step#3 Run above Test class Product.java (ctrl+F11) to see output.

Example#3 JDBC Appender Example

Step#1 : Define one Test class with example code with log methods.

```
package com.app;
import org.apache.log4j.Logger;
public class Product {
        private static Logger log=Logger.getLogger(Product.class);
          public static void main(String[] srs){
                  log.debug("Hello");
                  log.info("Hello");
                  log.warn("Hello");
                  log.error("Hello");
                  log.fatal("Hello");

        }
}
```

Step#2 Create this table before you run example, all the logs will be stored in below table

```
CREATE TABLE LOGS (
        METHOD VARCHAR(20)  NOT NULL,
        DATED   DATETIME                NOT NULL,
```


Scanned with CamScanner

```
    LOGGER VARCHAR(50)   NOT NULL,
    LEVEL  VARCHAR(10)   NOT NULL,
    MESSAGE VARCHAR(1000)     NOT NULL
);
```

Step#3 Create log4j.properties file with below code

log4j.properties

```
# Root logger option
log4j.rootLogger=DEBUG, db

# Define the Jdbc appender
log4j.appender.db=org.apache.log4j.jdbc.JDBCAppender
log4j.appender.db.driver=com.mysql.jdbc.Driver
log4j.appender.db.URL=jdbc:mysql://localhost:3306/test
log4j.appender.db.user=root
log4j.appender.db.password=root
log4j.appender.db.layout=org.apache.log4j.PatternLayout
log4j.appender.db.sql=INSERT INTO LOGS VALUES ('%M', now()
,'%C','%p','%m')
```

prints DB server Date & Time

Step#4 Run above Test class Product.java (ctrl+F11) to see output.

Example#4 SMTP Appender Example

Step#1 : Define one Test class with example code with log methods.

```
package com.app;
import org.apache.log4j.Logger;
public class Product {
    private static Logger log=Logger.getLogger(Product.class);
      public static void main(String[] srs){
            log.debug("Hello");
            log.info("Hello");
```


Scanned with CamScanner

```
            log.warn("Hello");
            log.error("Hello");
            log.fatal("Hello");

        }
}
```

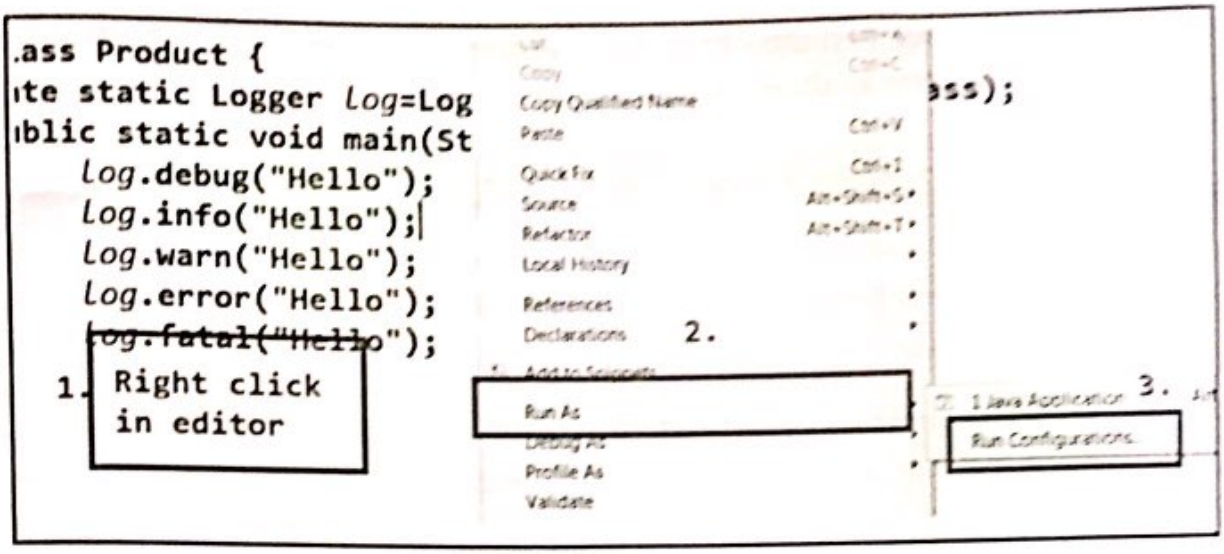
Step#2 Create log4j.properties file with below code

** Modify user Name and password before you run this application

```
# Root logger option
log4j.rootLogger=DEBUG, email
#configuring the SMTP appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.SMTPHost=smtp.gmail.com
log4j.appender.email.SMTPUsername=raghusirjava@gmail.com
log4j.appender.email.SMTPPassword=abdeyhfk@33
log4j.appender.email.From=raghusirjava@gmail.com
log4j.appender.email.To=<your email id>@gmail.com
log4j.appender.email.Subject=Log of messages
log4j.appender.email.Threshold=DEBUG
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern= %m %n
```

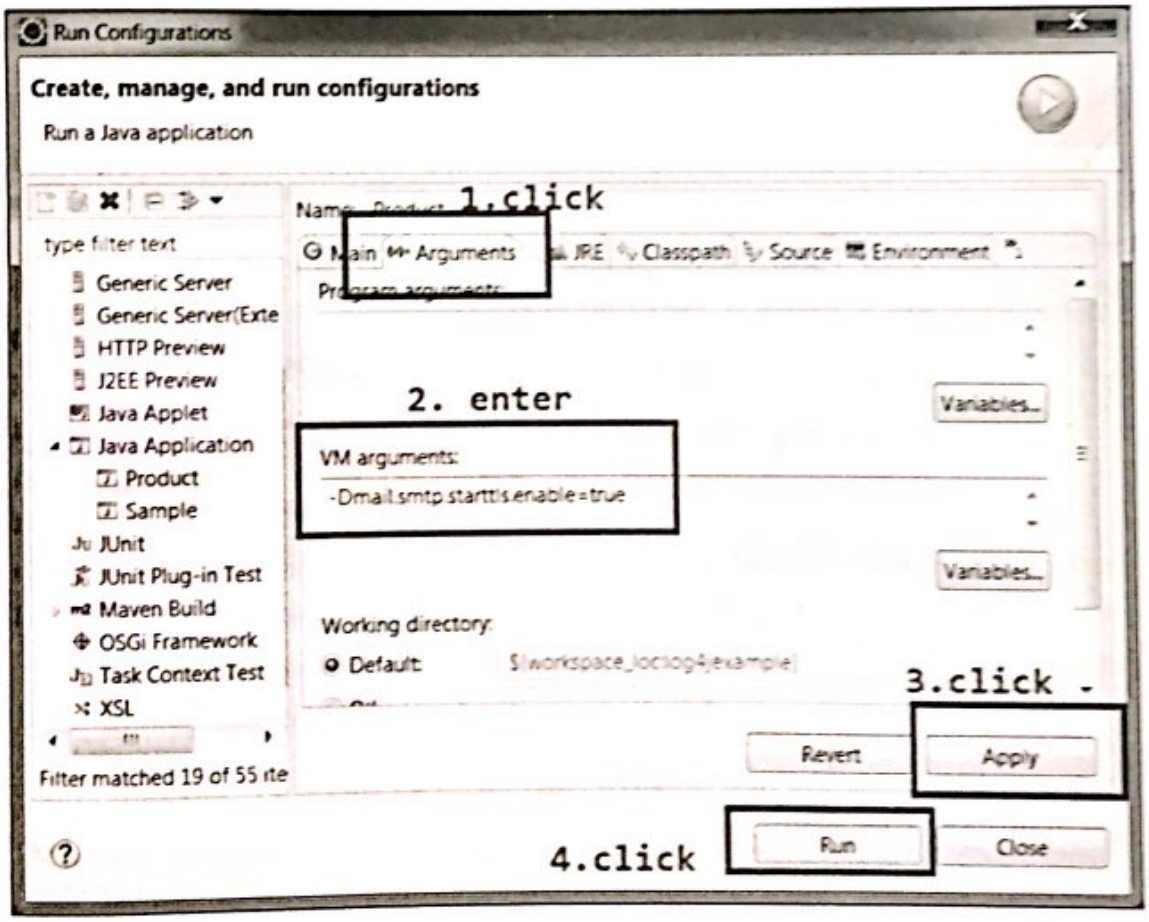
Step#3 Configure this VM Argument before you run application
Right click in Editor and Choose "Run As" => "Run Configuration"

Scanned with CamScanner



Enter this key=value in VM arguments in below screen

```
-Dmail.smtp.starttls.enable=true
```



Scanned with CamScanner

Step#4 Run above Test class Product.java (ctrl+F11) to see output. Or Click on Run Option shown in above screen

Facebook Group: https://www.facebook.com/groups/thejavatemple

Scanned with CamScanner

4. GIT - CODE REPOSITORY TOOL (VERSION MANAGEMENT)

To develop app. we write different types of files in projects (like java, xml, jsp etc..). All these files are stored finally in a central server i.e. Code Repository. It maintains the details of code. like

- ✓ How many times code is modified?
- ✓ Who modified what code(+/- added/removed)
- ✓ Changes done on what date and time? Maintaining these details is known as Version Management/Version Control.

a. In developement mostly used tool is GIT(gitHub).
b. It maintains two level repository process for cross verify and commit.
c. For this we need to create an account in github.com(using email id and verify email after creation).
d. It maintains staging area to select /verify/comment the file while adding to local repository.

Git Process:

Git Repository Types:

a. Public : Free for every one
b. Private : Paid version.

Remote Repository(Steps):

a. goto https://github.com/
b. SignUp(Register) with details
c. Goto Email -and verfiy Link
d. login with un and pwd
e. Create Repository Type Public (Companies uses- Private Type)
f. Copy Git Link: https://github.com/abcd/testVen.git it is secured with un,pwd.

Eclipse Workspace and Local Repository Steps:

Scanned with CamScanner

In eclipse,

1. Go to window menu
2. Show Views search for "GIT" select GitRepository and GitStaging.
3. Click on Window
4. showView
5. History.

Creating local repository:

1. Right Click on Project
2. choose "Team" option
3. Share project
4. select checkbox "create or use repository"
5. select option shown below
6. click create repository button
7. finish

GIT Operations with Flow:

- add file to git staging (Click-drag-drop).
- Commit and Push
- Paste Git Link in URI input box
- enter userName and password.
- next/next/finish.
- right click on project
- Choose team
- Select pull, then ,rebase to update local from remote.

Some Interview Questions over GIT:

1) Can you provide GIT/Client/Apps URLs? No, Sorry.all details what you are asking is private and confidential. I cannot provide those.

2)Who created your git account ?

Git administrator name-"XYZ", in my company. He handles all permission and auth. details.

Scanned with CamScanner

3)What type of Account Repository areyou using?

It is private account handled by company.

4)When you check-in or push your code?

After Implementation of complete module and UnitTesting code in my local, I'll push to remote.

5)How will you find about one file all modification??

We use history option in eclipse

->right click->team->shown in history.

we can also choose any two files to compare with each other ex:select

any history options->right click->compare with each other.

6)How do you identify code modifications in GIT?

Using Symbols (+) code is added (-) code is removed.

7)What is the difference between commit and push?

commit: update code from workspace to Local Repository

push :Update code from local to remote repository.

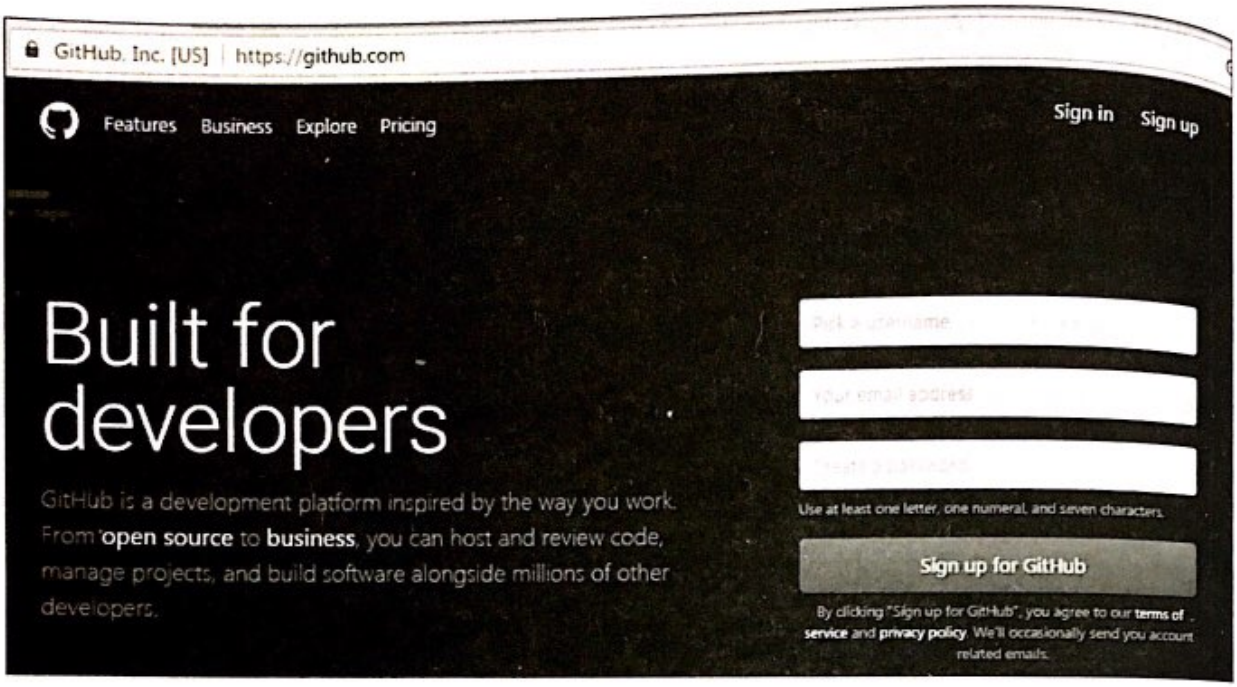
8)How can you update you workspace/local with remote?

By using pull and rebase operations.

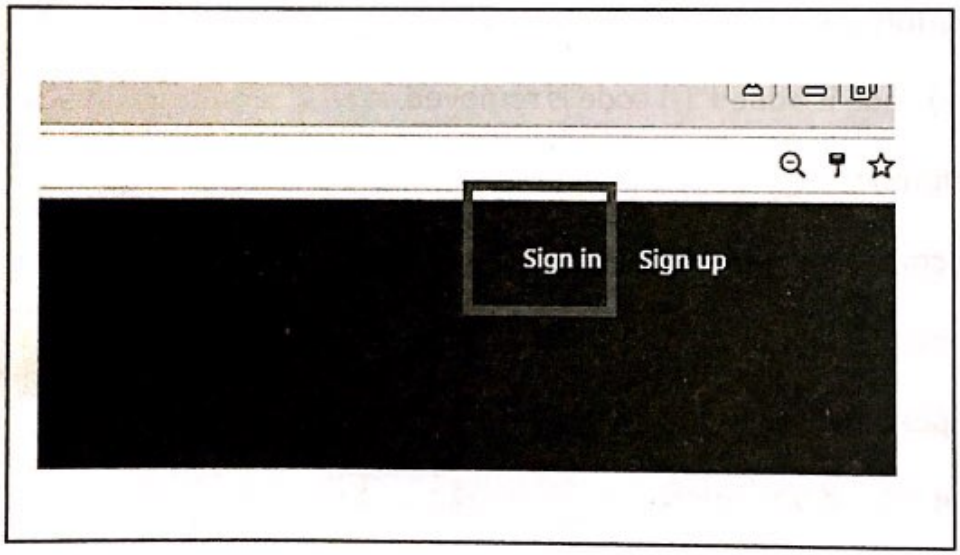
Screens Help For Complete Process:

https://github.com/

Scanned with CamScanner

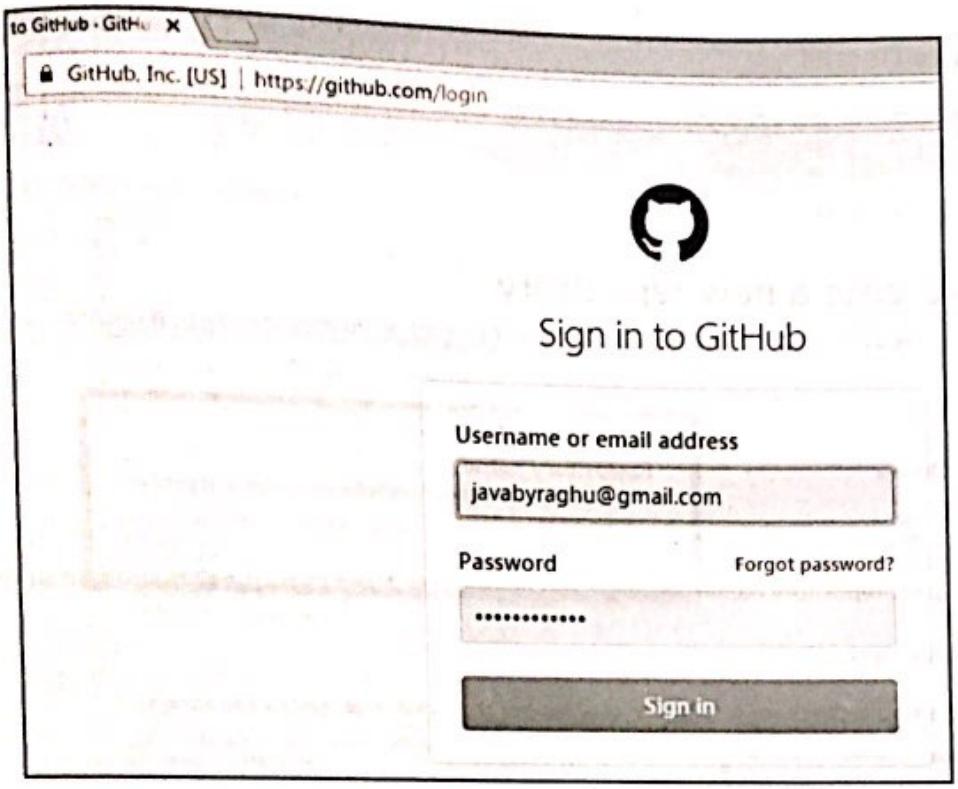


Goto email account and click on verify link. Click on signin

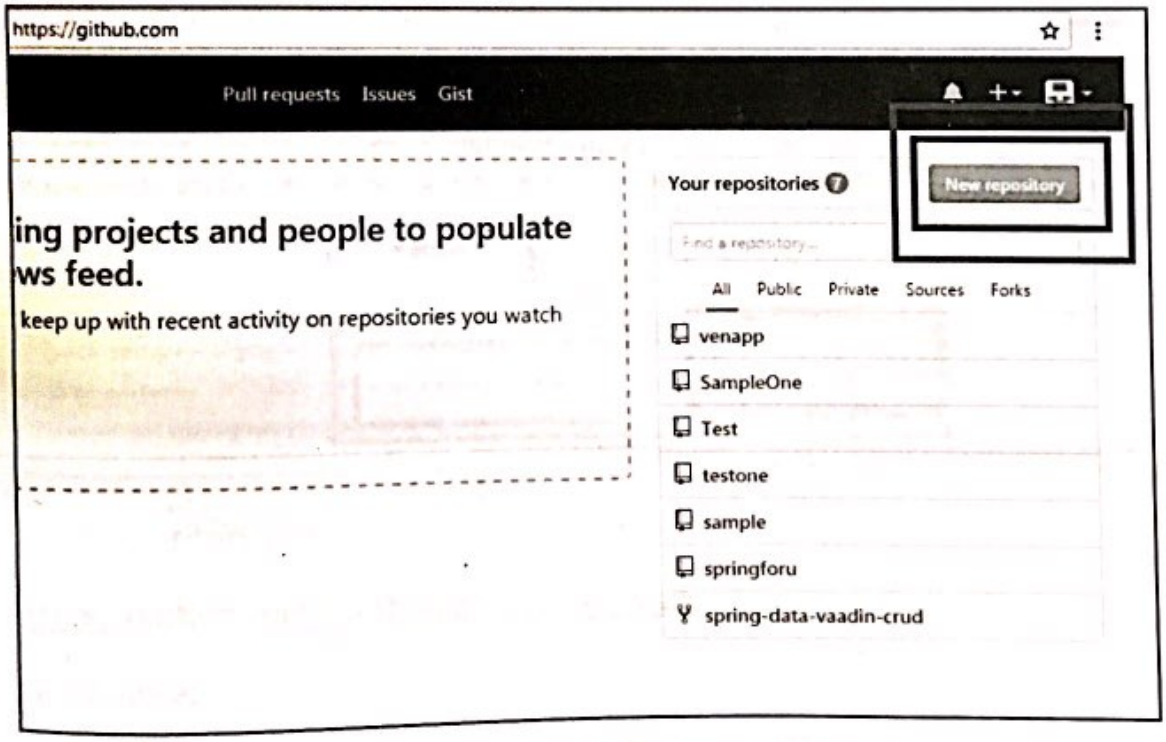


Enter user name and password.

Scanned with CamScanner

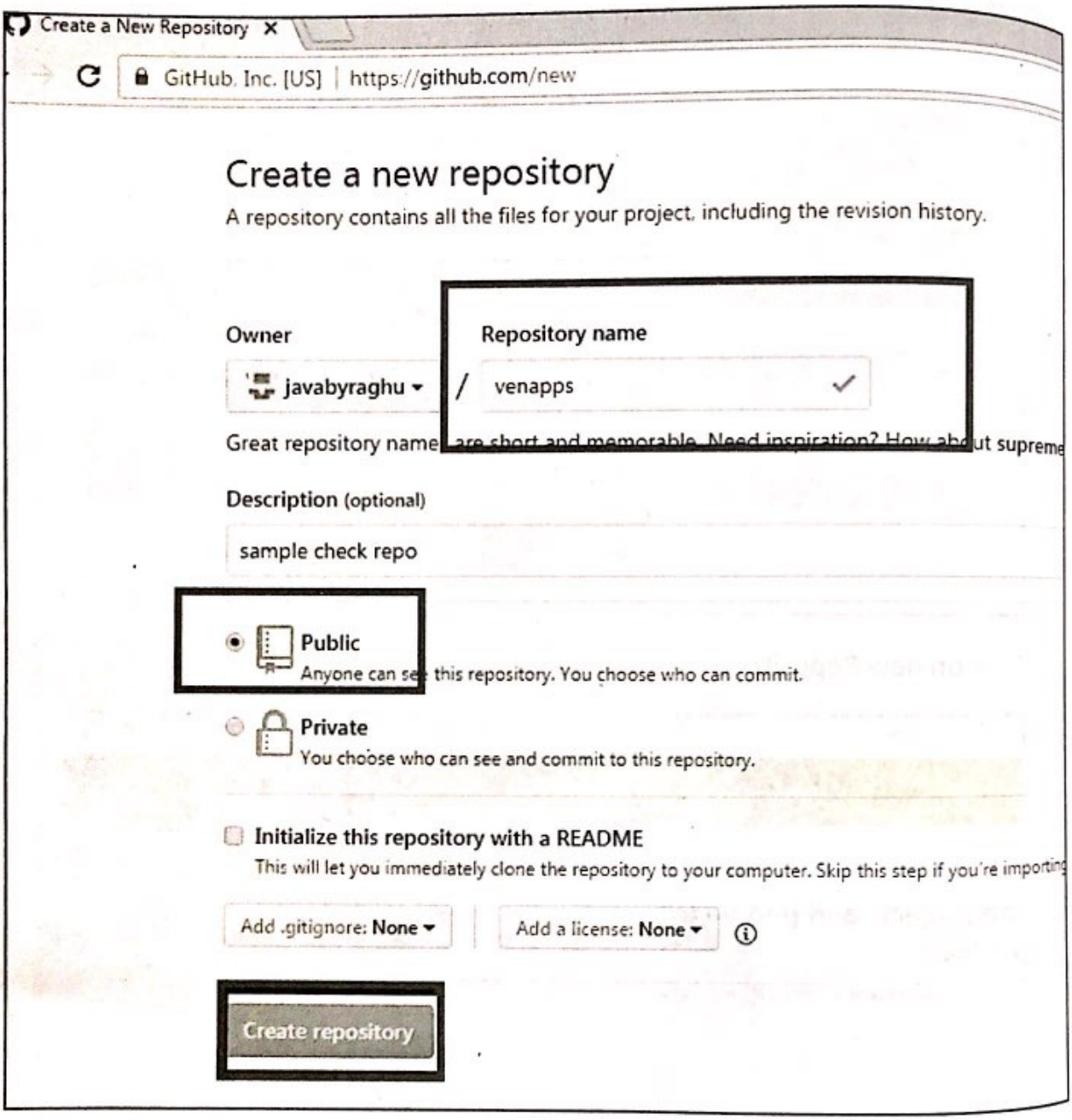


Click on new Repository



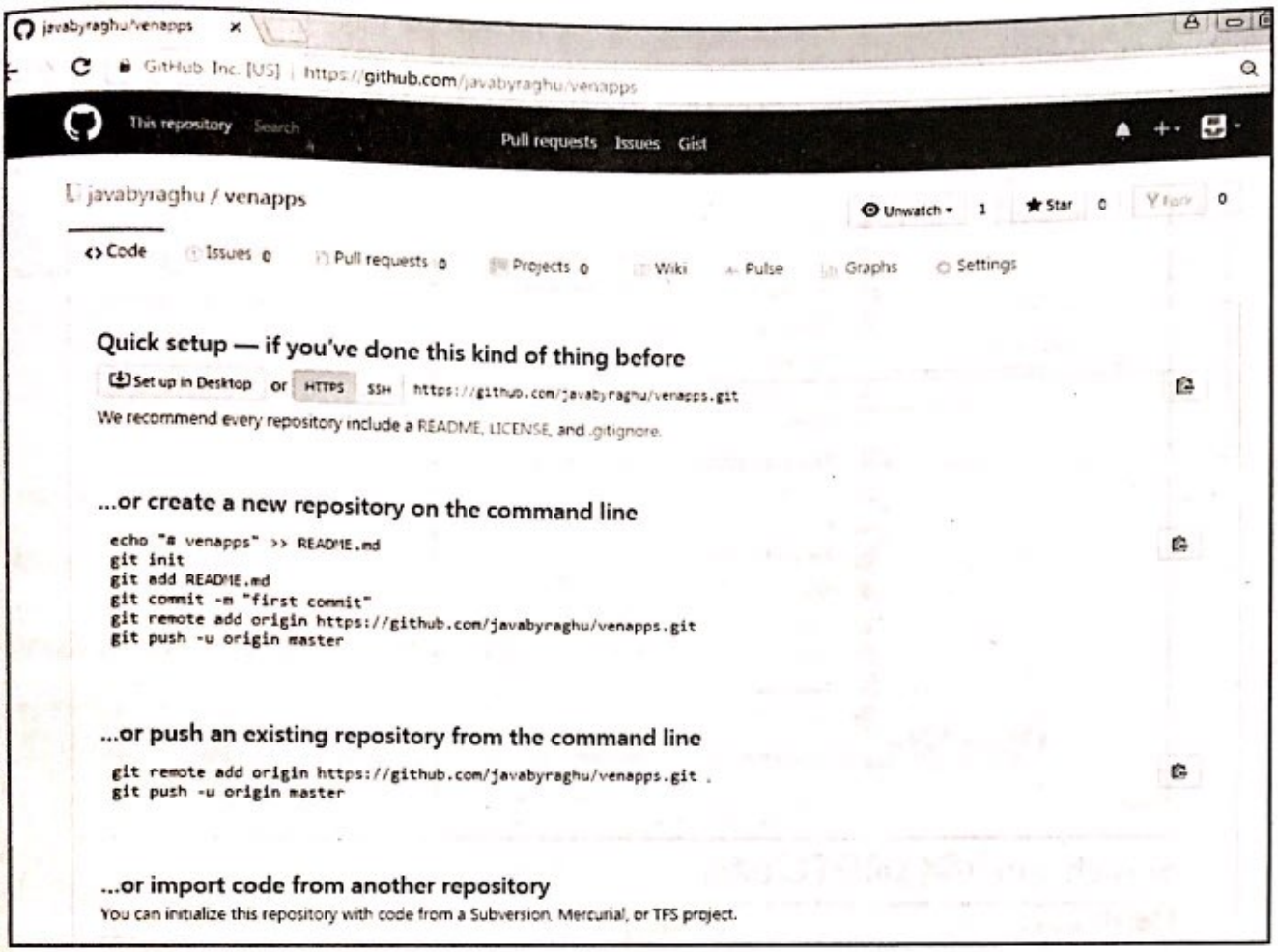
Scanned with CamScanner

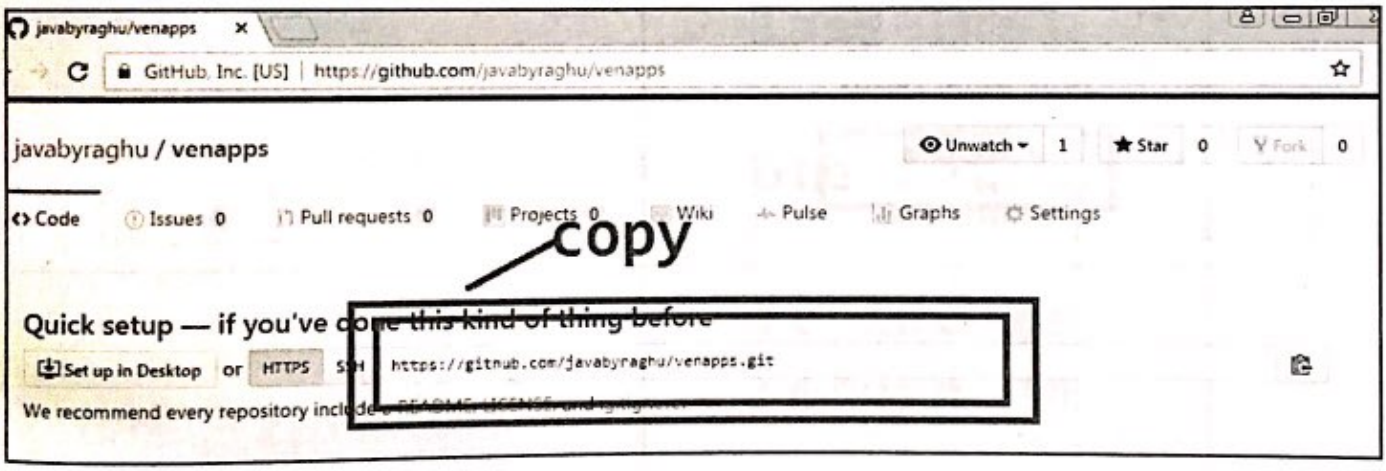
Provide Repository Details



On click create repository

Scanned with CamScanner





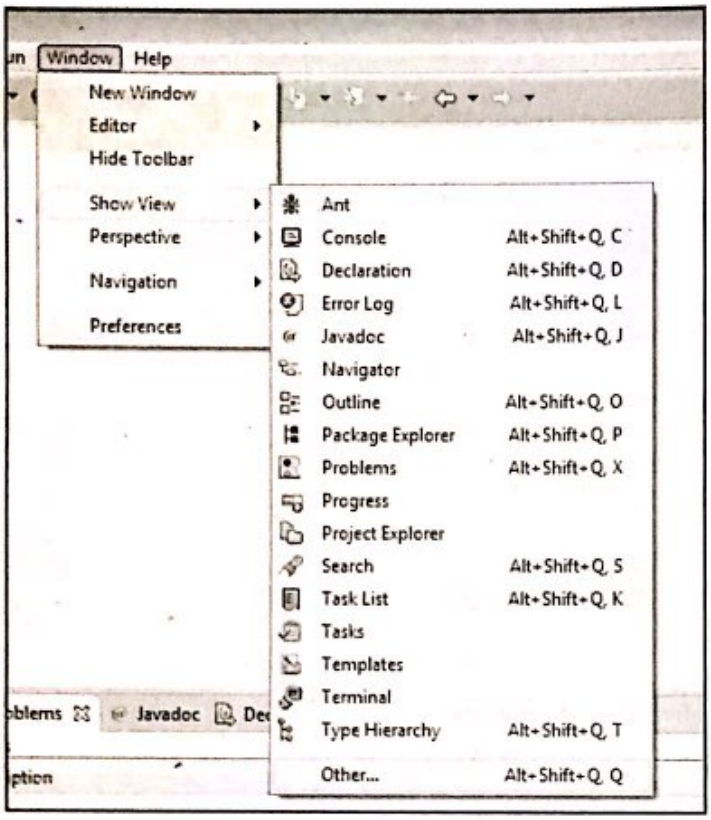
Ex : (Repository Link)

https://github.com/javabyraghu/venapps.git

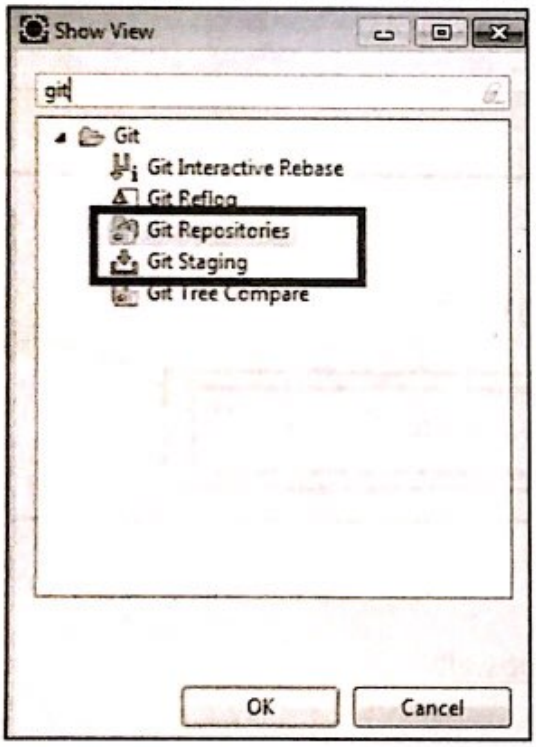
Got Eclipse:

- Create A new Java Project **(project name and repository name Should be same)**
- Add Git Views

Scanned with CamScanner

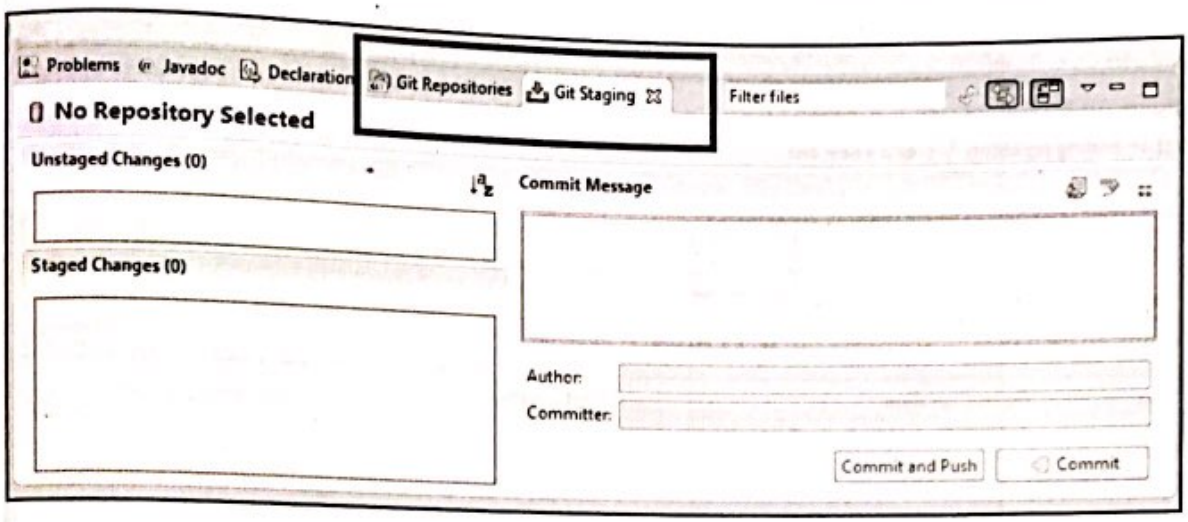


Serach with Git, select Git Repository, Staging



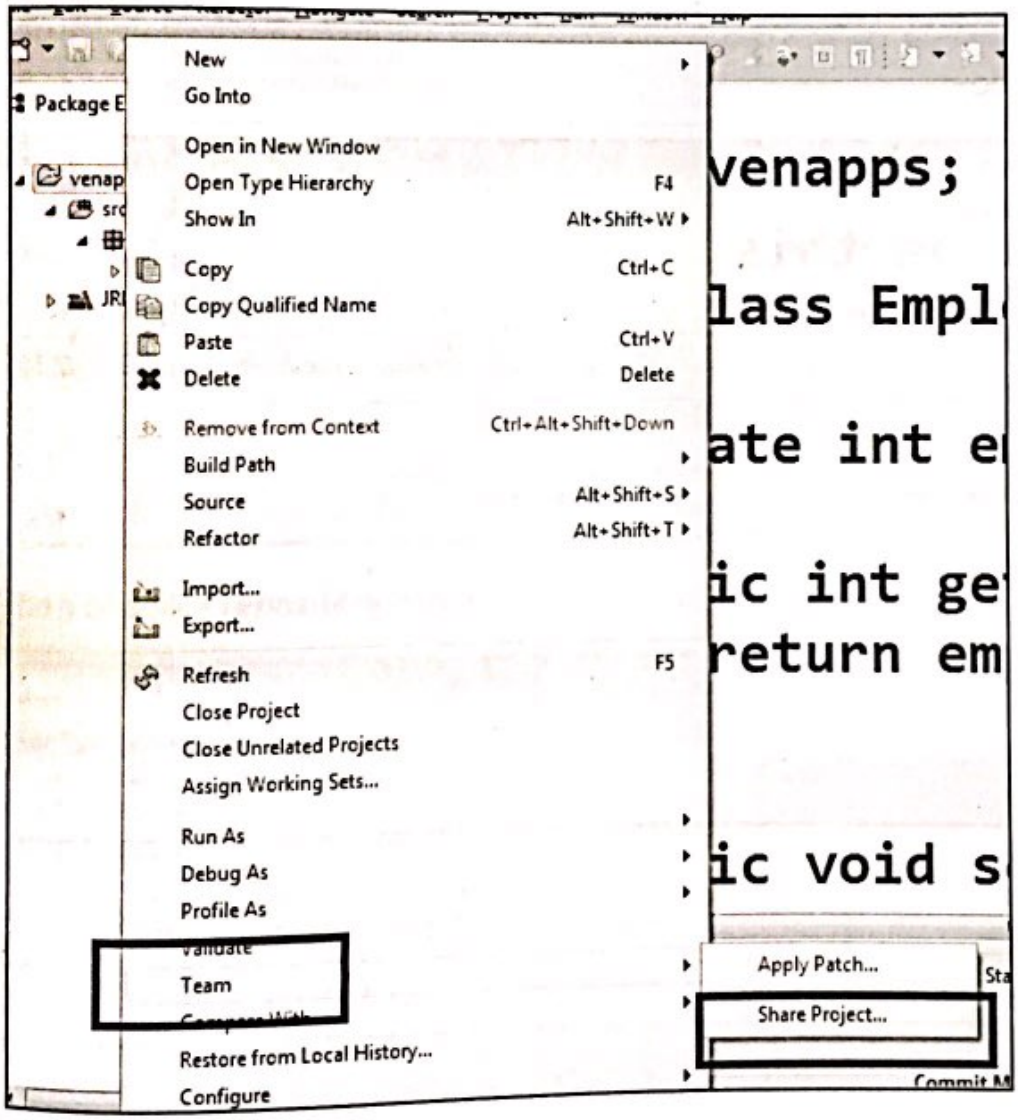
Observer options like

Scanned with CamScanner



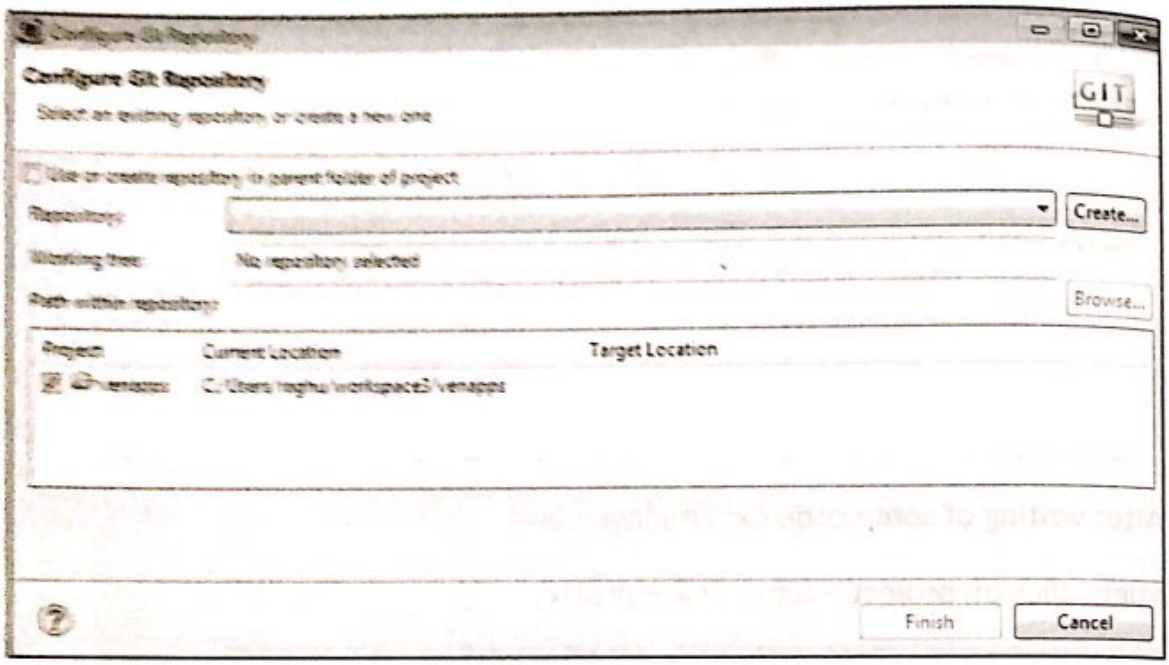
After writing of some code ex: Employee.java

>right click on project > tem > share project

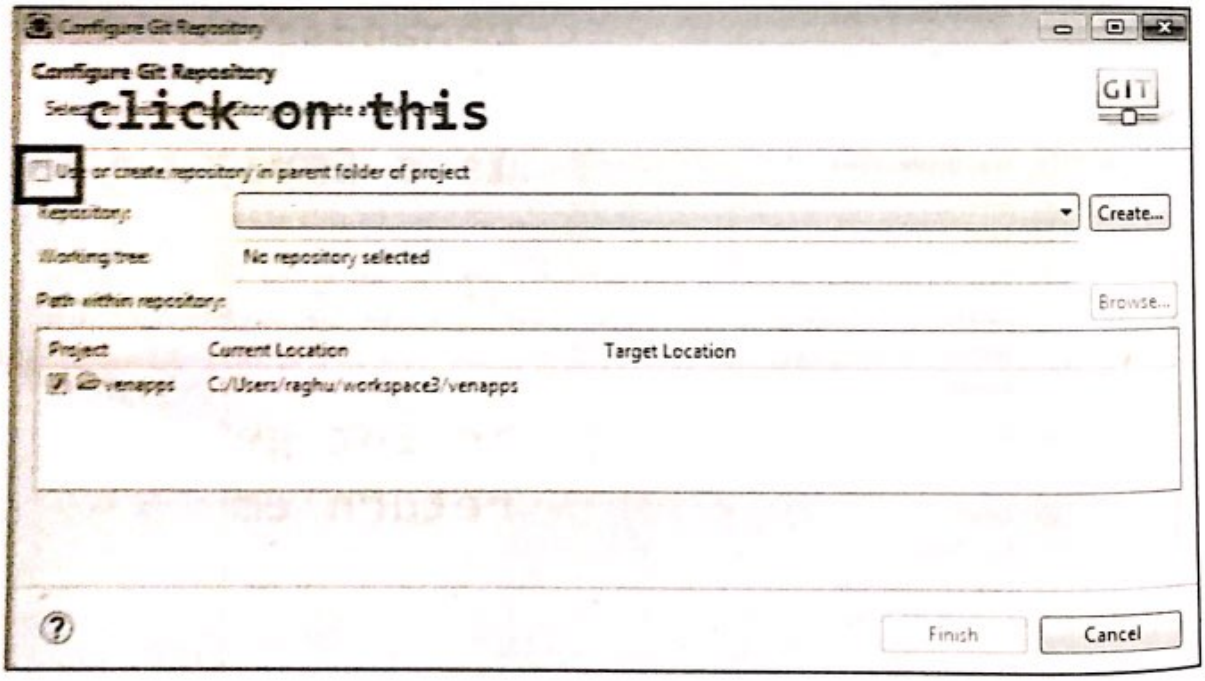


looks as below:

Scanned with CamScanner

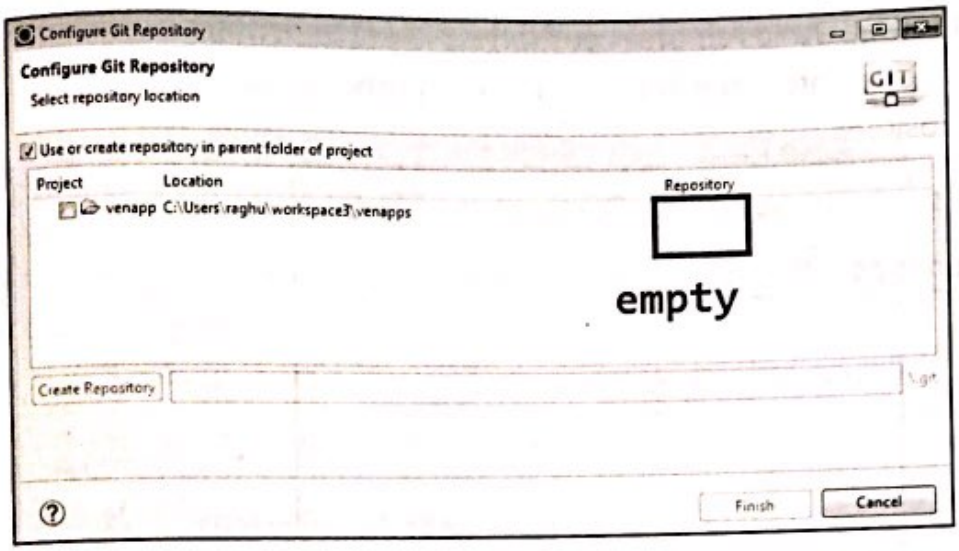


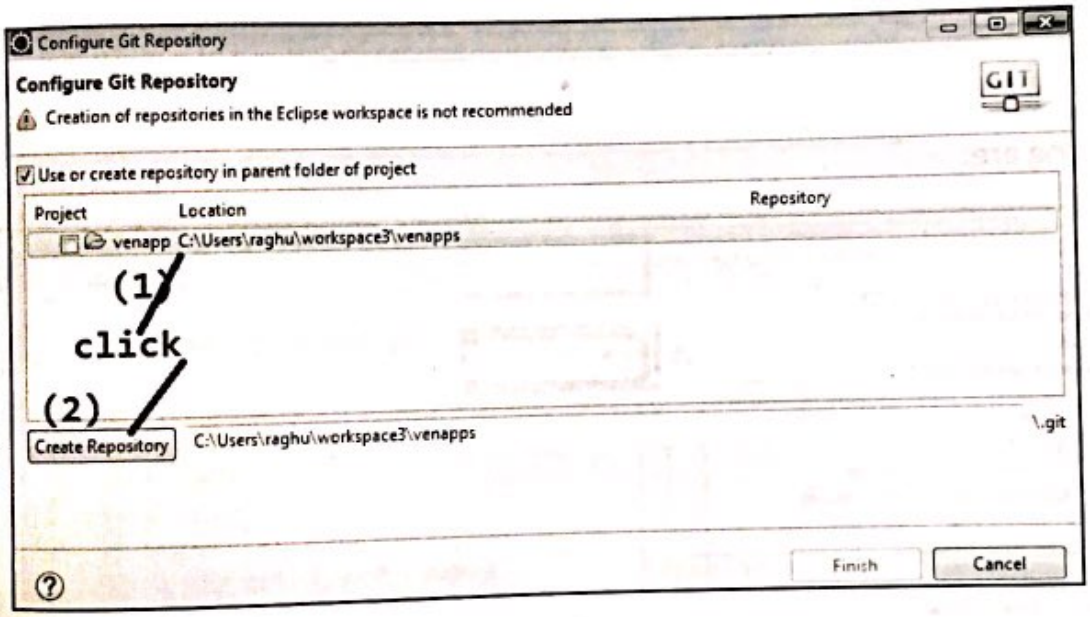
Click on check box:



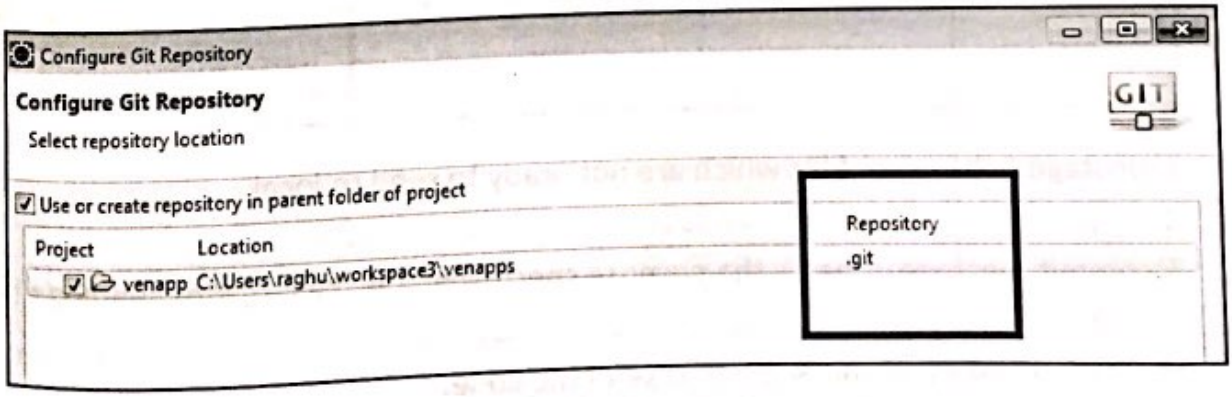
looks as below:

Scanned with CamScanner





then observe repository name:



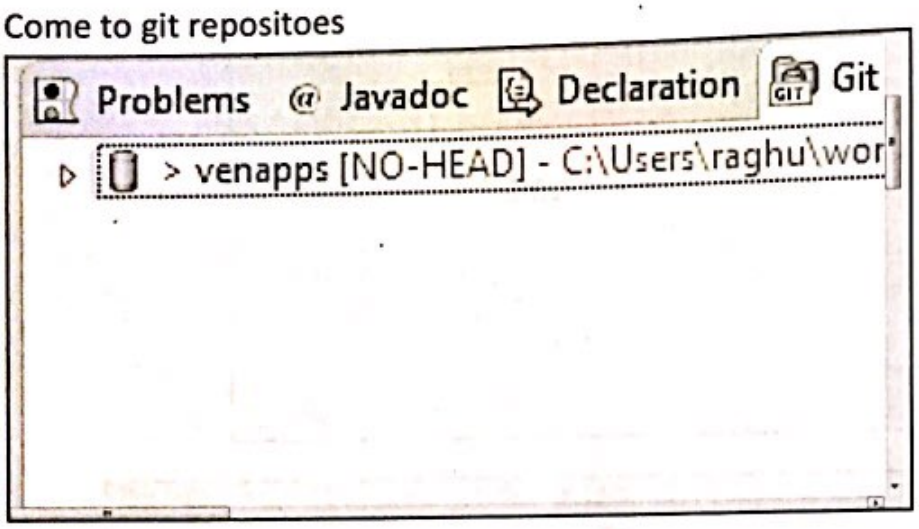
click on finish.

Scanned with CamScanner

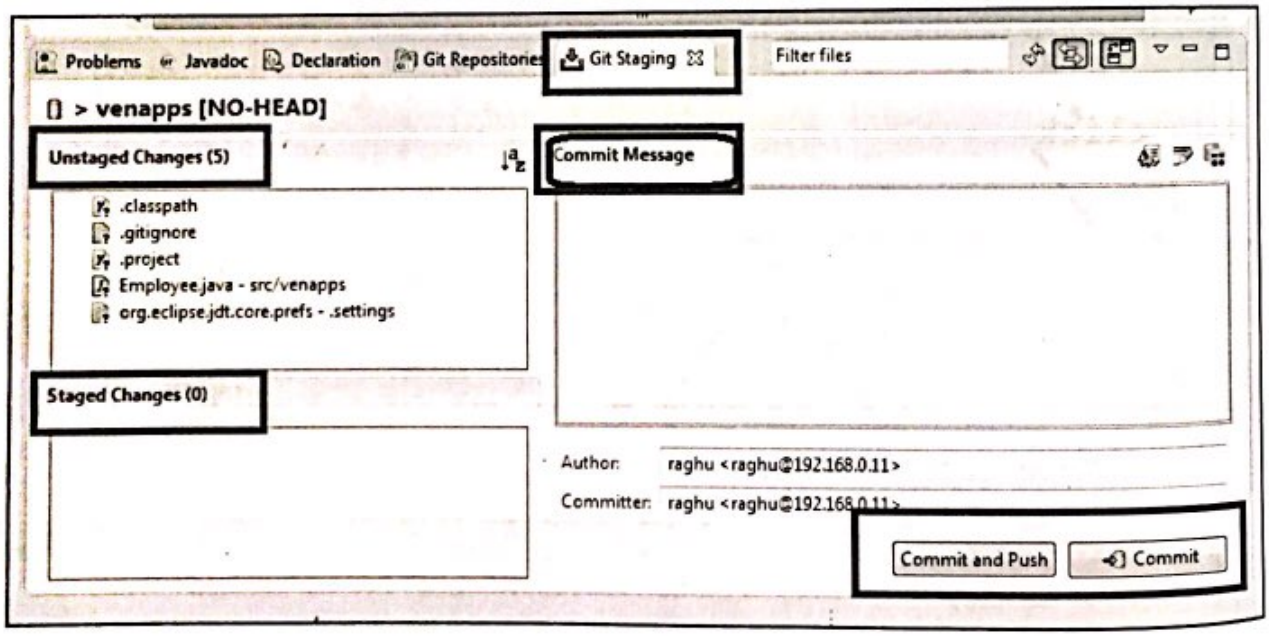
By this we created local and remote repositories.

Now link local and remote, using firsh commit and push.

Come to git repositoes



In staging are:



1)Unstaged changes : files which are not ready to send to local

2)staged chages: file which needs to move to local/remote

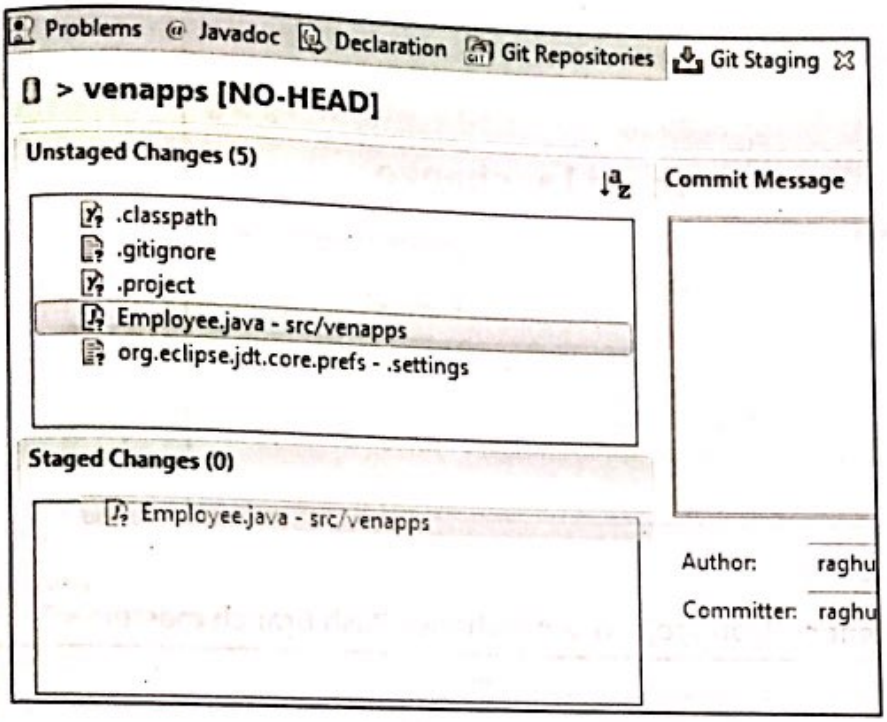
3)commit message: what is the purpose specify here as message for sending of this code.

4) click commit/ commit & push to send the code.

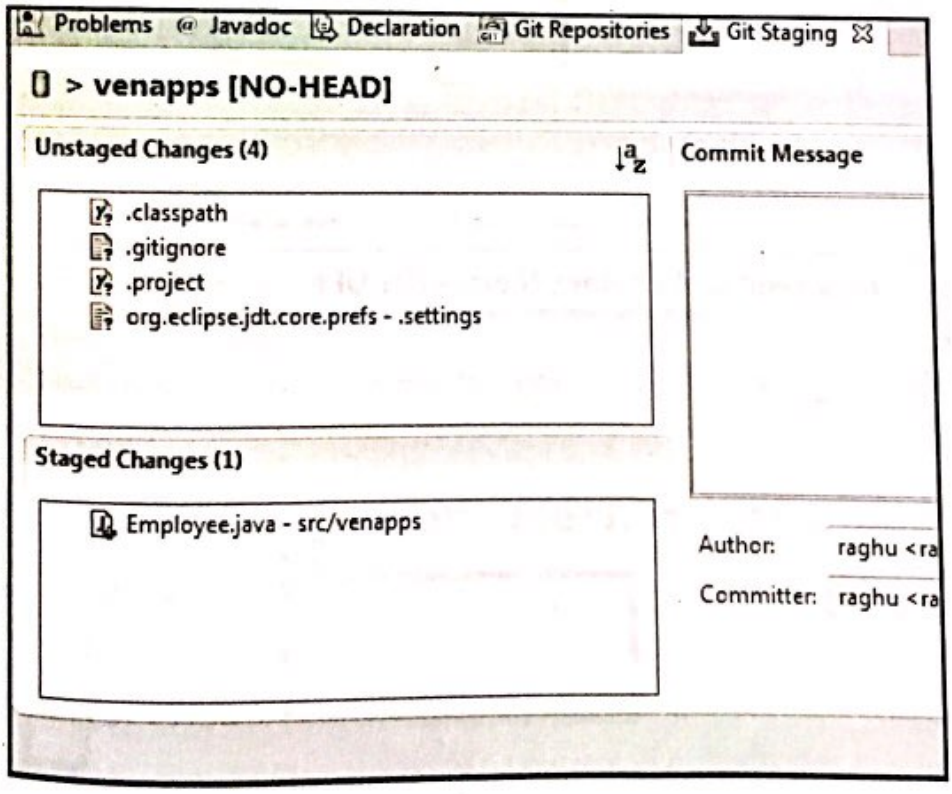
*)Click on a file in unstage drag and drop into stage.

Click and Drag:

Scanned with CamScanner



After drop

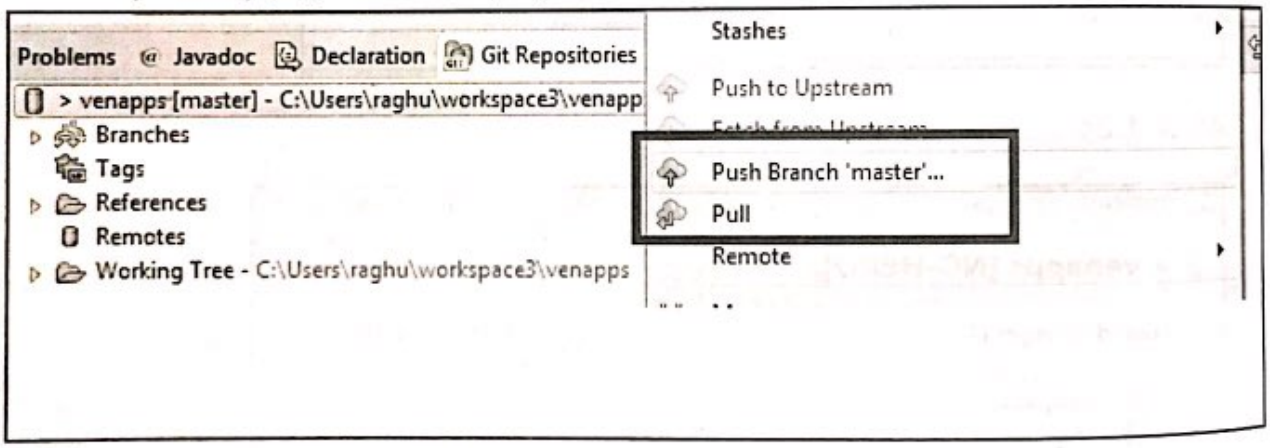


Enter commit message

Scanned with CamScanner

And click on commit.

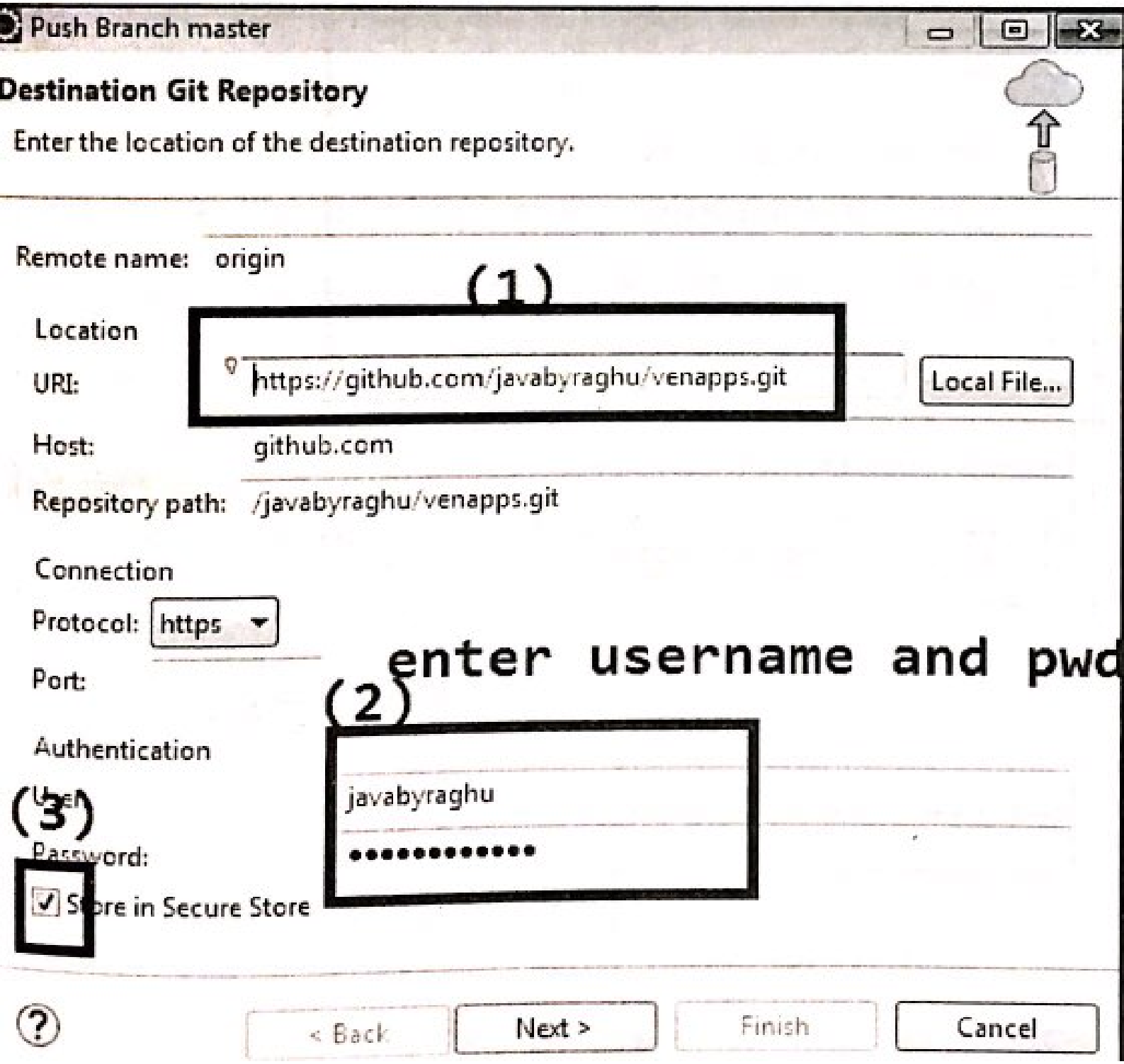
In GitRepository , right click on projectName>choose Push Branch master



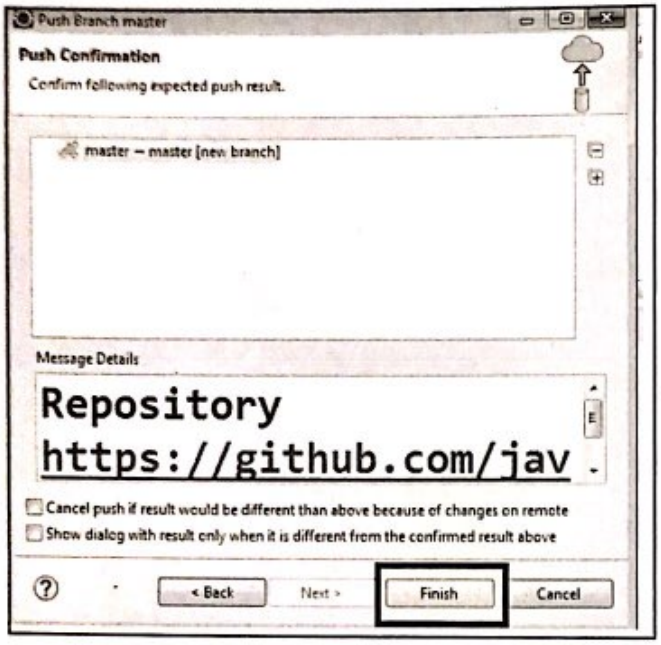
After Push option one screen looks below, there enter URI (Repository link)

Ex: https://github.com/javabyraghu/venapps.git

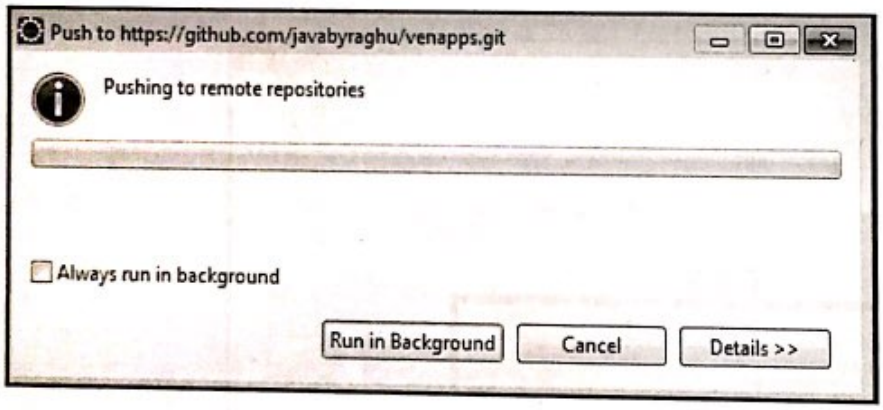
Scanned with CamScanner



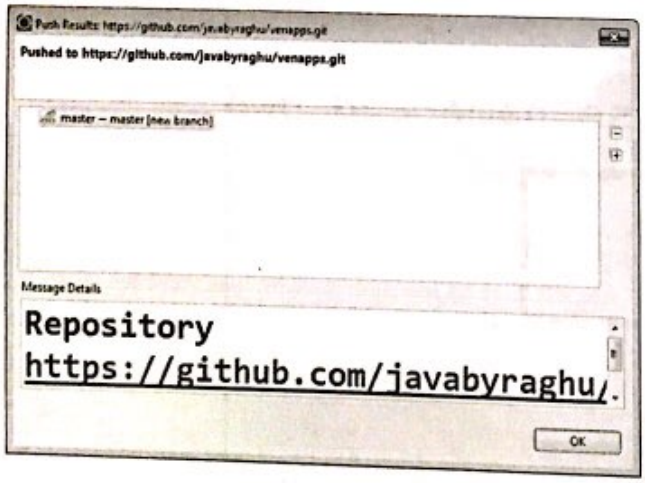
Scanned with CamScanner



Shows as below:

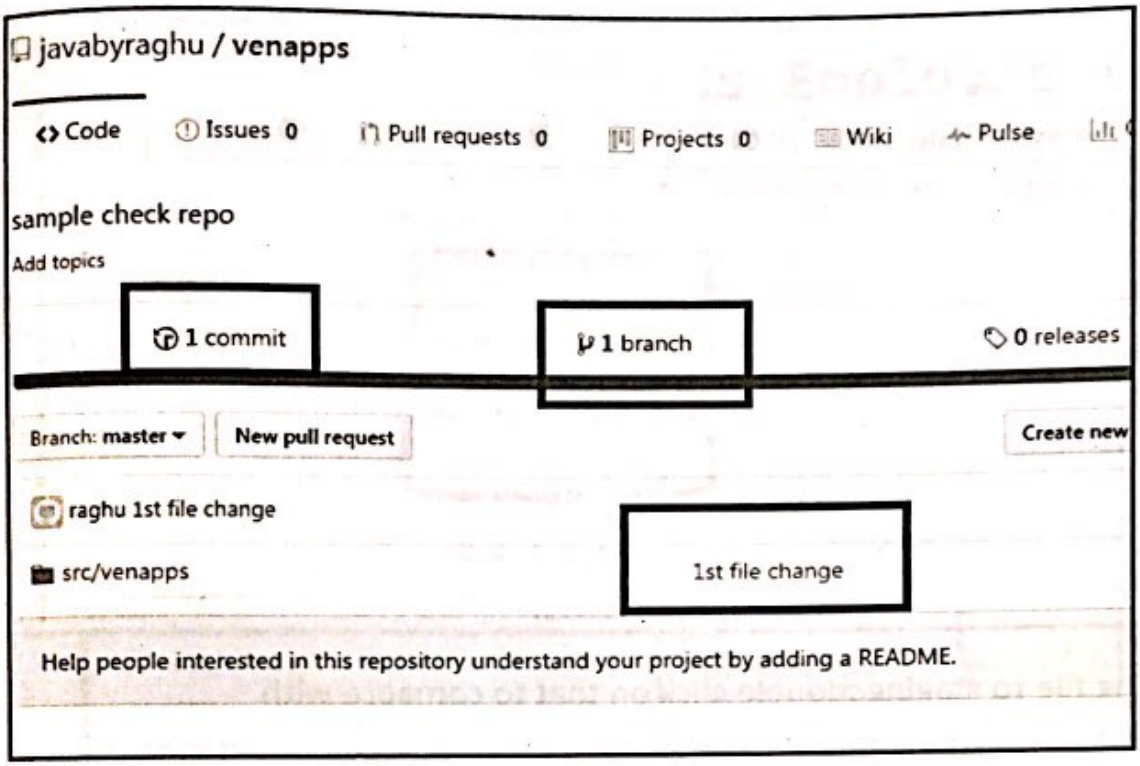


On success:

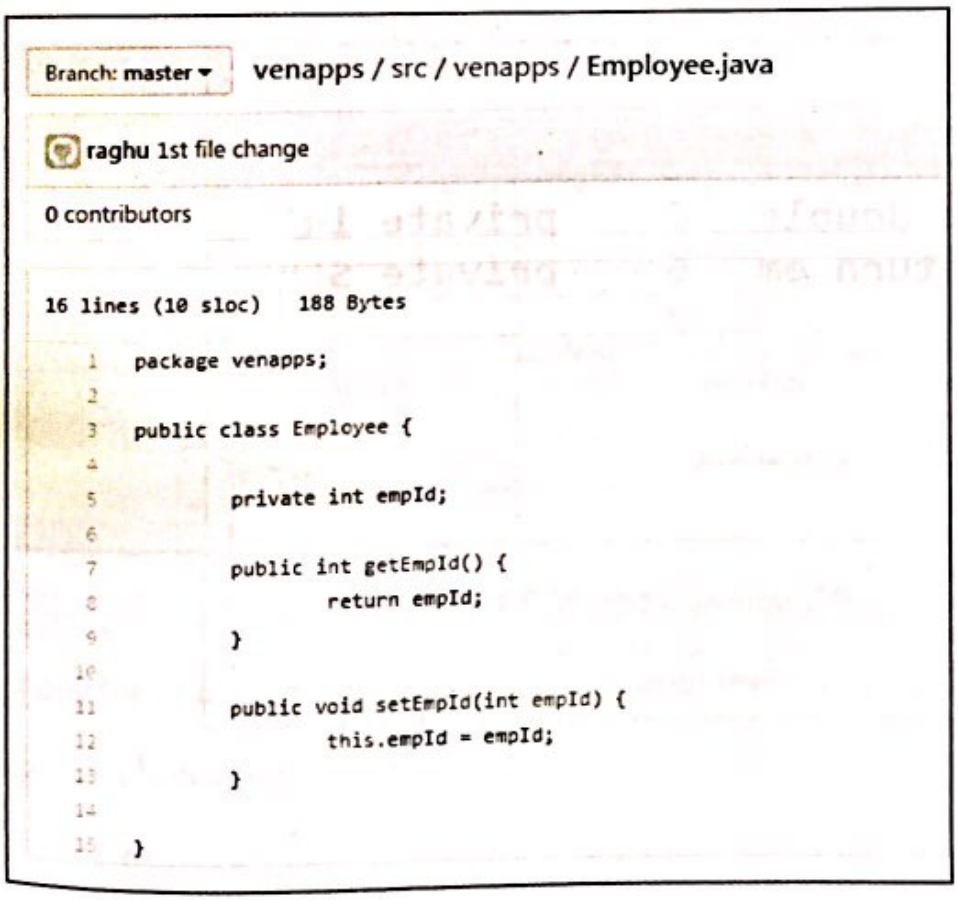


Scanned with CamScanner

Come back to browser and refresh:

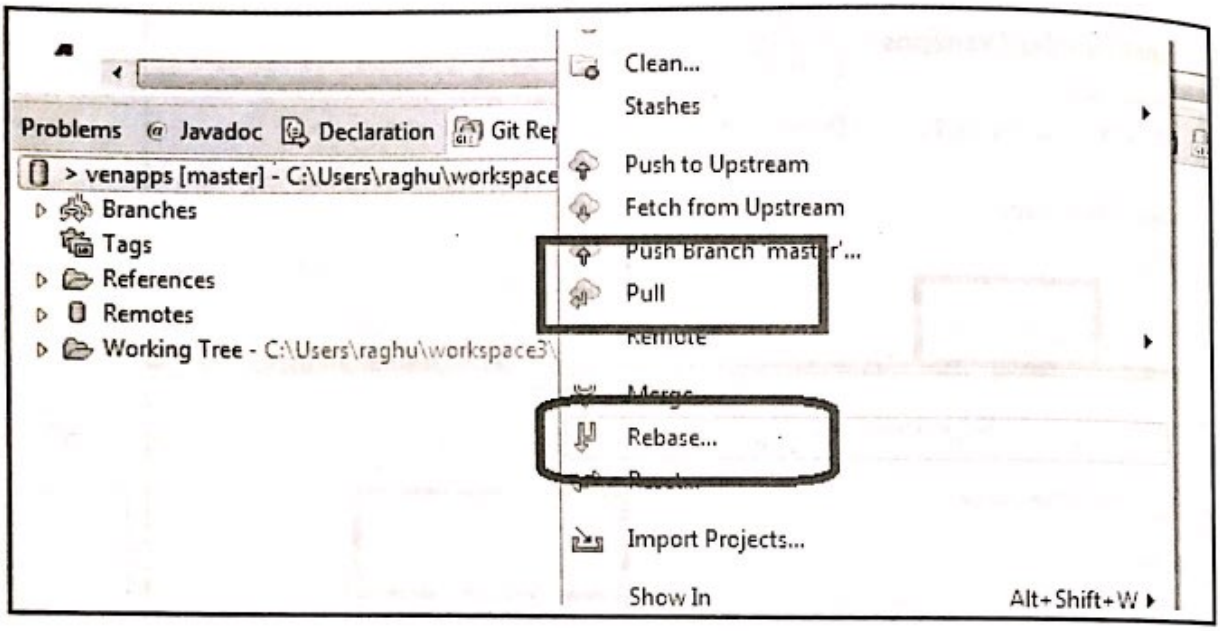


Click on src/venapp>Employee.java

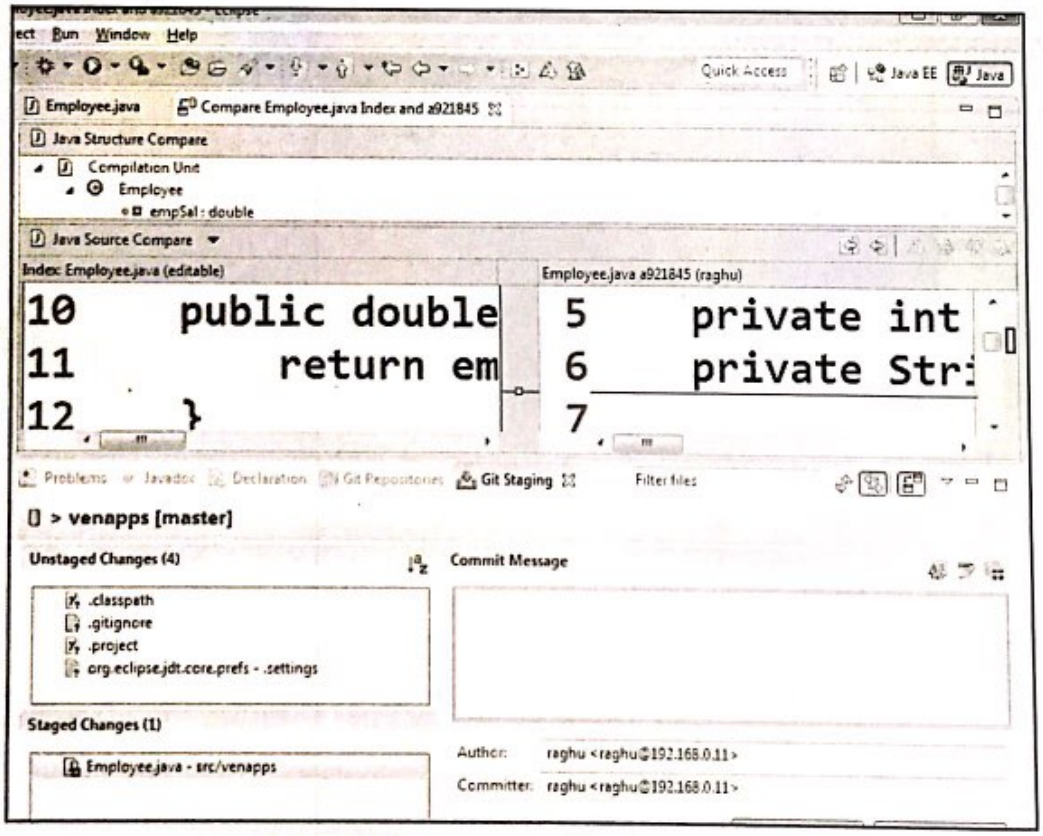


Scanned with CamScanner

Right click on project: pull and then rebase for other update:



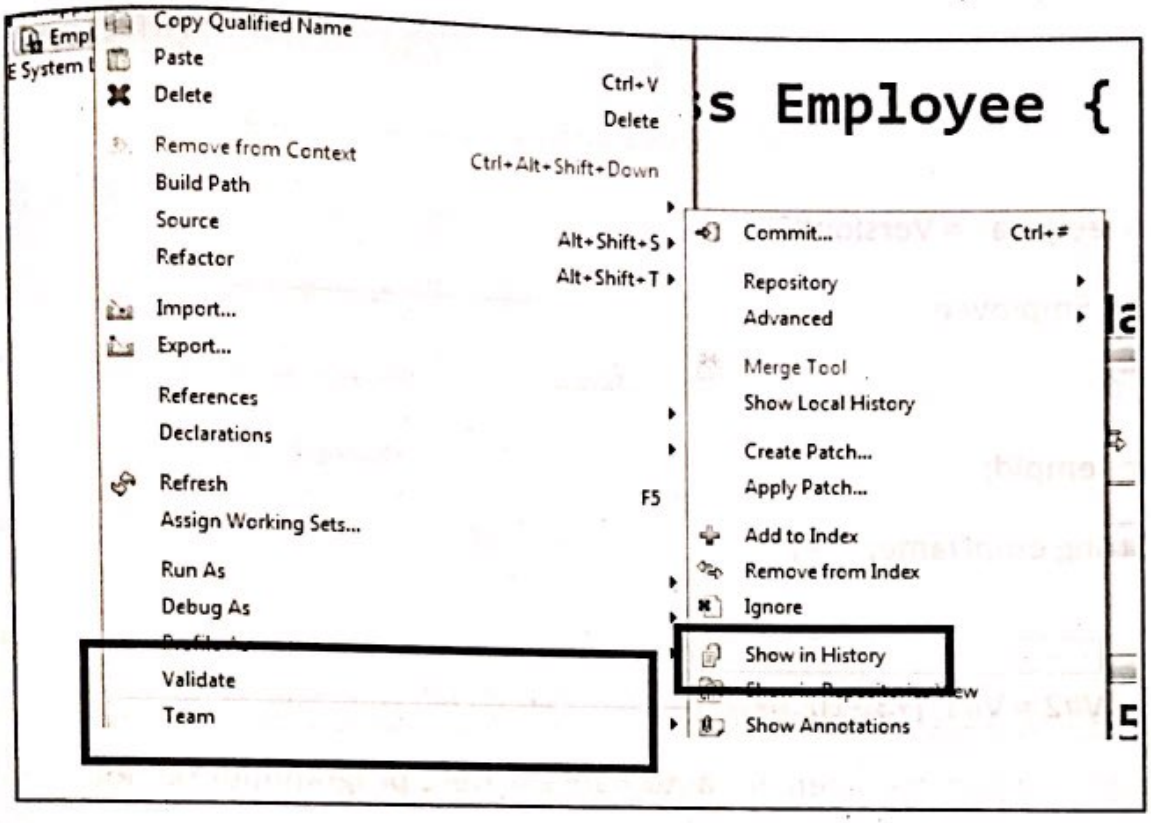
After Ading file to staging ,double click on that to comapre with Old one (previous one)looks as :



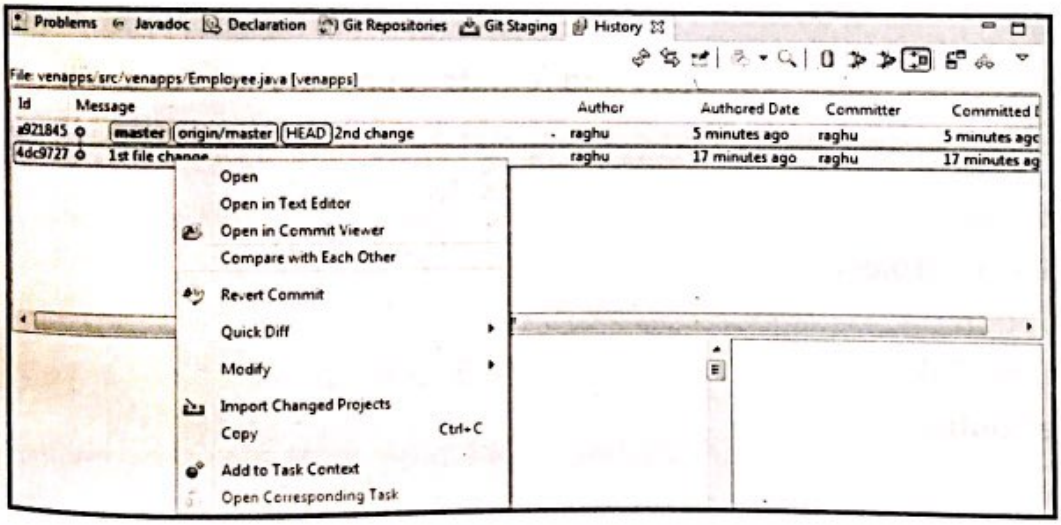
To see history:

Scanned with CamScanner

Right click on File (ex: Employee.java)> Team>Showin History



View history:



** GIT Merge Conflicts **

If any file code is modified(added/removed) then file version will be changed.

ex: Employee.java = Version#1

1. class Employee
2. {

Scanned with CamScanner

3.

4. }

Employee.java = Version#2

1. class Employee
2. {
3.+ int empId;
4.+ String empName; - }
5.+ }

V#2 = V#1 [+3,-1].

Git manages version management automatically Here programmer do not need to remember any version number.

** On git pull & rebase, local repository and remote repository versions will become equal.

1. Repositories
2. right click on project
3. "pull" option
4. again right click
5. "rebase" option.

Merge conflict example:

Step#1: Dev#1 has done git pull & rebase

Step#2: Dev#2 has done git pull & rebase

Step#3: Dev#1 modified Employee.java

Scanned with CamScanner

(that is changed to Version#2) and

did add/commit/push.

Now git has Employee.java Version#2

Step#4: Dev#2 Still working with old code

of Employee.java (Version#1).

File modified and did add/commit/

push, then git shows

Git - Version Conflict: Dev#2

Employee.java(version#1) not

matched with Git Repo Employee.java

(Version#2). Please update before

commit/push.

To Resolve merge conflict steps are.

- project -> right click -> reset
- project -> right click -> pull & rebase
- project -> right click -> merge

Now code will be updated to V#2,

write your code, then >>> add/commit/push.

face book group Id: **https://www.facebook.com/groups/thejavatemple/**

email :

javabyraghu@gmail.com

Scanned with CamScanner

6. ECLIPSE SHORTCUTS

1. Manage Files and Projects	
Ctrl+N	Create new project using the Wizard
Ctrl+Alt+N	Create new project, file, class, etc.
Alt+F, then .	Open project, file, etc.
Ctrl+Shift+R	Open Resource (file, folder or project)
Alt+Enter	Show and access file properties
Ctrl+S	Save current file
Ctrl+Shift+S	Save all files
Ctrl+W	Close current file
Ctrl+Shift+W	Close all files
F5	Refresh content of selected element with local file system

2. Editor Window	
Focus/ cursor must be in Editor Window for these to work.	
F12	Jump to Editor Window
Ctrl+Page Down/Ctrl+Page Up	Switch to next editor / switch to previous editor
Ctrl+M	Maximize or un-maximize current Editor Window (also works for other Windows)
Ctrl+E	Show list of open Editors. Use arrow keys and enter to switch
Ctrl+F6/Ctrl+Shift+F6	Show list of open Editors. Similar to ctrl+e but switches immediately upon release of ctrl
Alt+Arrow Left/Alt+Arrow Right	Go to previous / go to next Editor Window
Alt+-	Open Editor Window Option menu
Ctrl+F10	Show view menu (features available on left

Scanned with CamScanner

	vertical bar: breakpoints, bookmarks, line numbers, ...)
Ctrl+F10, then n	Show or hide line numbers
Ctrl+Shift+Q	Show or hide the diff column on the left (indicates changes since last save)

3. Navigate in Editor	
Home/End	Jump to beginning / jump to end of indention. Press home twice to jump to beginning of line
Ctrl+Home/End	Jump to beginning / jump to end of source
Ctrl+Arrow Right/Arrow Left	Jump one word to the left / one word to the right
Ctrl+Shift+Arrow Down/Arrow Up	Jump to previous / jump to next method
Ctrl+L	Jump to Line Number. To hide/show line numbers, press ctrl+F10 and select 'Show Line Numbers'
Ctrl+Q	Jump to last location edited
Ctrl+./Ctrl+,	Jump to next / jump to previous compiler syntax warning or error
Ctrl+Shift+P	With a bracket selected: jump to the matching closing or opening bracket
Ctrl+[+]/Ctrl+- on numeric keyboard	Collapse / Expand current method or class
Ctrl+[/]/Ctrl+* on numeric keyboard	Collapse / Expand all methods or classes
Ctrl+Arrow Down/Ctrl+Arrow Up	Scroll Editor without changing cursor position
Alt+Page Up/Alt+Page Down	Next Sub-Tab / Previous Sub-Tab

4. Select Text	
Shift+Arrow Right/Arrow Left	Expand selection by one character to the left / to the right
Ctrl+Shift+Arrow Right/Arrow Left	Expand selection to next / previous word

Scanned with CamScanner

Shift+Arrow Down/Arrow Up	Expand selection by one line down / one line up
Shift+End/Home	Expand selection to end / to beginning of line
Ctrl+A	Select all
Alt+Shift+Arrow Up	Expand selection to current element (e.g. current one-line expression or content within brackets)
Alt+Shift+Arrow Left/Arrow Right	Expand selection to next / previous element
Alt+Shift+Arrow Down	Reduce previously expanded selection by one step

5. Edit Text	
Ctrl+C/Ctrl+X/Ctrl+V	Cut, copy and paste
Ctrl+Z	Undo last action
Ctrl+Y	Redo last (undone) action
Ctrl+D	Delete Line
Alt+Arrow Up/Arrow Down	Move current line or selection up or down
Ctrl+Alt+Arrow Up/Ctrl+Alt+Arrow Down/	Duplicate current line or selection up or down
Ctrl+Delete	Delete next word
Ctrl+Backspace	Delete previous word
Shift+Enter	Enter line below current line
Shift+Ctrl+Enter	Enter line above current line
Insert	Switch between insert and overwrite mode
Shift+Ctrl+Y	Change selection to all lower case
Shift+Ctrl+X	Change selection to all upper case

6. Search and Replace	
Ctrl+F	Open find and replace dialog
Ctrl+K/Ctrl+Shift+K	Find previous / find next occurrence of search

Scanned with CamScanner

	term (close find window first)
Ctrl+H	Search Workspace (Java Search, Task Search, and File Search)
Ctrl+J/Ctrl+Shift+J	Incremental search forward / backwards. Type search term after pressing ctrl+j, there is now search window
Ctrl+Shift+O	Open a resource search dialog to find any class

7. Indentions and Comments	
Tab/Shift+Tab	Increase / decrease indent of selected text
Ctrl+I	Correct indention of selected text or of current line
Ctrl+Shift+F	Autoformat all code in Editor using code formatter
Ctrl+/	Comment / uncomment line or selection (adds '//')
Ctrl+Shift+/	Add Block Comment around selection (adds '/... */')
Ctrl+Shift+	Remove Block Comment
Alt+Shift+J	Add Element Comment (adds '/** ... */')

8. Editing Source Code	
Ctrl+Space	Opens Content Assist (e.g. show available methods or field names)
Ctrl+1	Open Quick Fix and Quick Assist
Alt+/	Propose word completion (after typing at least one letter). Repeatedly press alt+/ until reaching correct name
Ctrl+Shift+Insert	Deactivate or activate Smart Insert Mode (automatic indention, automatic brackets, etc.)

Scanned with CamScanner

Ctrl+O	Show code outline / structure
F2	Open class, method, or variable information (tooltip text)
F3	Open Declaration: Jump to Declaration of selected class, method, or parameter
F4	Open Type Hierarchy window for selected item
Ctrl+T	Show / open Quick Type Hierarchy for selected item
Ctrl+Shift+T	Open Type in Hierarchy
Ctrl+Alt+H	Open Call Hierarchy
Ctrl+Shift+U	Find occurrences of expression in current file
Ctrl+move over method	Open Declaration or Implementation

10. Refactoring	
Alt+Shift+R	Rename selected element and all references
Alt+Shift+V	Move selected element to other class or file (With complete method or class selected)
Alt+Shift+C	Change method signature (with method name selected)
Alt+Shift+M	Extract selection to method
Alt+Shift+L	Extract local variable: Create and assigns a variable from a selected expression
Alt+Shift+I	Inline selected local variables, methods, or constants if possible (replaces variable with its declarations/ assignment and puts it directly into the statements)

11. Run and Debug	
Ctrl+F11	Save and launch application (run)
F11	Debug
F5	Step Into function

Scanned with CamScanner

F6	Next step (line by line)
F7	Step out
F8	Skip to next Breakpoint

12. The Rest	
Ctrl+F7/Ctrl+Shift+F7	Switch forward / backward between views (panels). Useful for switching back and forth between Package Explorer and Editor.
Ctrl+F8/Ctrl+Shift+F8	Switch forward / backward between perspectives
Ctrl+P	Print
F1	Open Eclipse Help
Shift+F10	Show Context Menu right click with mouse

Scanned with CamScanner