JAVA DESIGN PATTERNS ASSIGNMENT SOLUTIONS

Assignment 1 - Singleton

```
public class LoggerSingleton implements Serializable

{

    public static volatile  LoggerSingleton instance;

   private LoggerSingleton()

{

}

   public static LoggerSingleton getInstance()

{

   if(instance == null)

{

   synchronized (LoggerSingleton.class)
{
   if (instance == null)
{
    instance = new LoggerSingleton();
}
}
}
       return instance;
}
       public void log(String message)
{
    System.out.println(message);
}
       public Object readResolve()
```

```
{
       return instance;
}
}

public class LoggerSingleton implements Serializable
{
    public static volatile  LoggerSingleton instance;
       private LoggerSingleton()
{
}
    public static LoggerSingleton getInstance()
{
    if(instance == null)
```

```
{
    synchronized (LoggerSingleton.class)
{
    if (instance == null)
{
    instance = new LoggerSingleton();
}
}
}
    return instance;
}
    public void log(String message)
{
    System.out.println(message);

}

    public Object readResolve()

{

    return instance;

}

}
```

Assignment 2 - Factory

```
public interface Person

{

        void wish(String message);

}

public class Male implements Person {
        @Override

        public void wish(String message) {

        System.out.println("Male class"+" "+message);

}

}

public class Female implements Person {

        @Override

        public void wish(String message) {

        System.out.println("Female class"+" "+message);

}

}

public class PersonFactory {

        public static Person create(String message)
```

```
{

        Person p=null;

        if(message=="Male")

        p=new Male();

        else

        p=new Female();

        return p;

}

}

public class Test {

        public static void main(String[] args) {

        Person p=PersonFactory.create("Male");

        p.wish("Hello!");

}
```

```
}
```

Assignment 3 - Template Method

```
public abstract class ComputerManufacturer {

    public void buildComputer()

{

    addHardDisk();

    addRam();

    addKeyboard();

}

    abstract void addHardDisk();

    abstract void addRam();

    abstract void addKeyboard();
```

```
}

public class DesktopManufacturer extends
ComputerManufacturer {

        @Override

        void addHardDisk() {

        System.out.println("Desktop - HardDisk added!");

}

        @Override

        void addRam() {

        System.out.println("Desktop - RAM added!");

}

        @Override

        void addKeyboard() {

        System.out.println("Desktop - Keyboard added!");
}

}


public class LaptopManufacturer extends
ComputerManufacturer {

    @Override

    void addHardDisk() {

    System.out.println("Laptop - HardDisk added!");

}

    @Override

    void addRam() {

    System.out.println("Laptop - RAM added!");

}

    @Override
```

```
        void addKeyboard() {

        System.out.println("Laptop - Keyboard added!");

}

}

public class Test {

        public static void main(String[] args) {

        ComputerManufacturer cm=new DesktopManufacturer();

        cm.buildComputer();

}

}
```

Assignment 4 - Adapter

```
        public class PaymentApp

{
        public int pay(int rupees)

{

        PaymentAdapter adapter = new PaymentAdapter();

        return adapter.pay(rupees);

}

        public static void main(String[] args)

{

    PaymentApp app = new PaymentApp();

    System.out.println( app.pay(1000)+"$");

}

}


public class PaymentAdapter

{
        public int pay(int rupees)

{

        return rupees/74;

}

}

public interface PaymentProcessor

{

        int pay(int dollars);

}
```

Assignment 5 - MVC

```
        <html>

        <body>

        <h3>Enter two number:</h3>
```

```
    <form  action="greatestController" method="post">

  Number 1 : <input name="number1"/><br/>

  Number 2: <input name="number2"/><br/>

  <input type="submit"/>

    </form>

    </body>

    </html>

    public class GreaterNumberModel

{

    public int findGreater(int a, int b)

{

    if (a > b)

{

    return a;
}
    else
{
    return b;
}
}
}
    import java.io.IOException;
    import javax.servlet.RequestDispatcher;
    import javax.servlet.ServletException;
    import javax.servlet.annotation.WebServlet;
    import javax.servlet.http.HttpServlet;
    import javax.servlet.http.HttpServletRequest;
```

```
    import javax.servlet.http.HttpServletResponse;

    /**

    * Servlet implementation class GreatestController

    */

    @WebServlet("/greatestController")

    public class GreatestController extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {

    int number1 =
Integer.parseInt(request.getParameter("number1"));

    int number2 =
Integer.parseInt(request.getParameter("number2"));

    GreaterNumberModel greaterNumberModel = new
GreaterNumberModel();
```

```
        int findGreater =
greaterNumberModel.findGreater(number1, number2);

        request.setAttribute("greaterNumber", findGreater);

        RequestDispatcher requestDispatcher =
request.getRequestDispatcher("greaterresult.jsp");

        requestDispatcher.forward(request, response);

}

}

        <%@ page language="java" contentType="text/html;
charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

        <!DOCTYPE html>

        <html>

        <head>

        <meta charset="ISO-8859-1">
        <title>Greater Number</title>

        </head>

        <body>

        <%

    int greater = (Integer)
request.getAttribute("greaterNumber");

    out.println("Greater of two numbers is :" + greater);

        %>

        </body>

        </html>
```

Assignment No 6 - DAO

```
        import com.mydao.model.student;

        public interface StudentDAO
```

```
{

        void save(Student student);

        void update(Student student);

        void delete(Student student);

}


        import java.util.List;

        import org.hibernate.SessionFactory;

        import org.springframework.stereotype.Repository;

        import
org.springframework.beans.factory.annotation.Autowired;

        import com.mydao.dao.StudentDao;

        import com.mydao.model.student;

        @Repository("studentDAO")

        public class studentDAOImpl extends
```

```
HibernateDaoSupport implements studentDAO

{

        @Autowired

    public void anyMethodName(SessionFactory
sessionFactory)

{

    setSessionFactory(sessionFactory);

}

        public void save(Student student)

{

        getHibernateTemplate().save(student);

}

        public void update(Student student)

{
        getHibernateTemplate().update(student);

    }

        public void delete(Student student){

        getHibernateTemplate().delete(student);

    }

    }
```