

Design Patterns and Principles

Mandatory Exercises

Singleton Pattern

- Visit my GitHub source code here: [GitHub](#)

Code:

```
package design.patterns.creational.singleton;

import java.util.Date;

public class Logger {
    private static Logger logger;
    private Date loginDate;

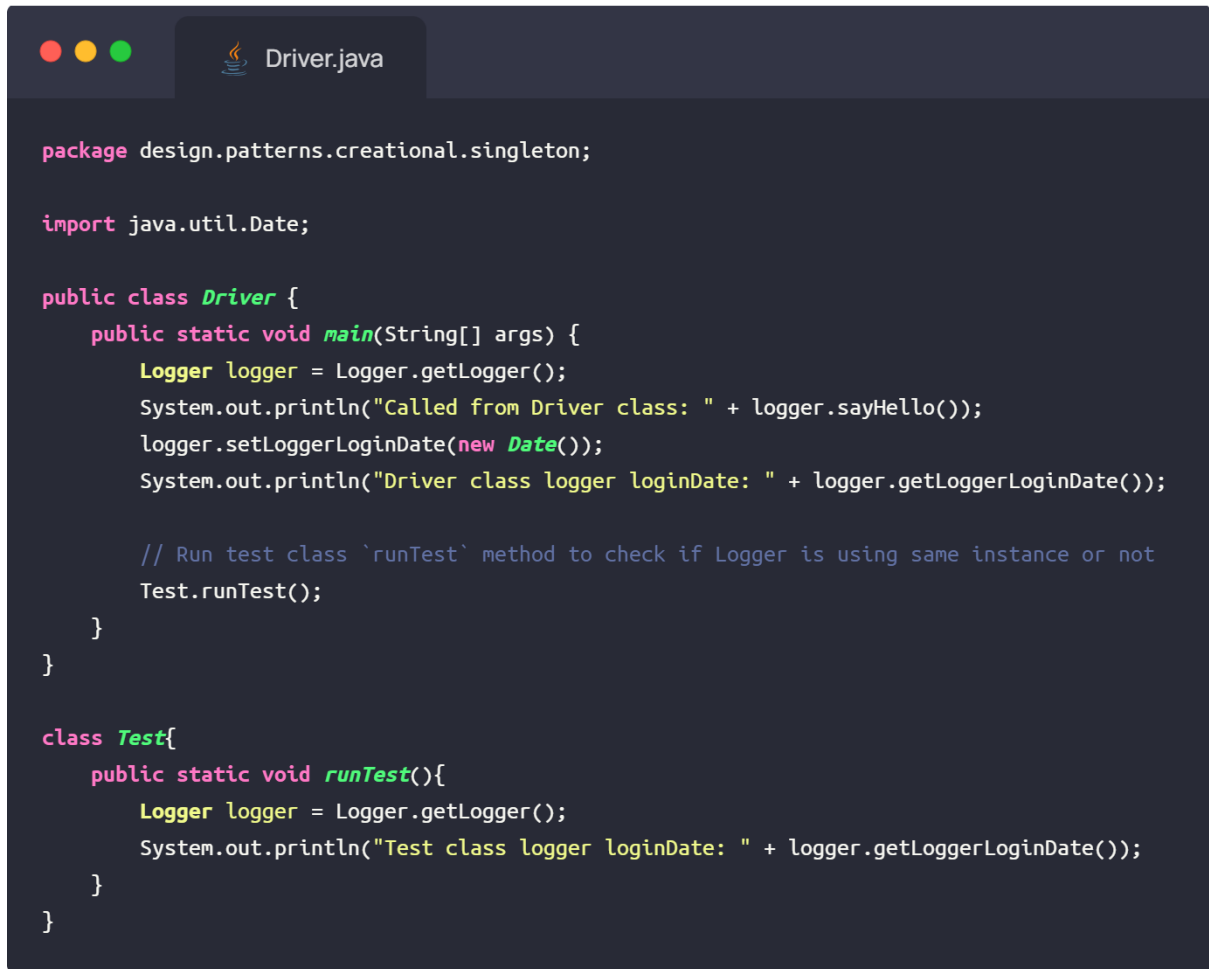
    private Logger(){}

    public static Logger getLogger(){
        if(logger == null)
            logger = new Logger();
        return logger;
    }

    public String sayHello(){
        return "Hello, from Logger skeleton class";
    }

    public void setLoggerLoginDate(Date date){
        loginDate = date;
    }

    public Date getLoggerLoginDate(){
        return loginDate;
    }
}
```



```
package design.patterns.creational.singleton;

import java.util.Date;

public class Driver {
    public static void main(String[] args) {
        Logger logger = Logger.getLogger();
        System.out.println("Called from Driver class: " + logger.sayHello());
        logger.setLoggerLoginDate(new Date());
        System.out.println("Driver class logger loginDate: " + logger.getLoggerLoginDate());

        // Run test class `runTest` method to check if Logger is using same instance or not
        Test.runTest();
    }
}

class Test{
    public static void runTest(){
        Logger logger = Logger.getLogger();
        System.out.println("Test class logger loginDate: " + logger.getLoggerLoginDate());
    }
}
```

Results:

```
Called from Driver class: Hello, from Logger skeleton class
Driver class logger loginDate: Sun Jun 22 01:12:50 IST 2025
Test class logger loginDate: Sun Jun 22 01:12:50 IST 2025
```

Factory Method Pattern

- Visit my GitHub source code here: [GitHub](#)
- Classes and interfaces created are
 1. Document.java (Interface)
 2. DocumentFactory.java (Abstract class)
 3. ExcelDocument.java (class implements Document interface)
 4. ExcelDocumentFactory.java (class extends DocumentFactory abstract class)
 5. PdfDocument.java (class implements Document interface)
 6. PdfDocumentFactory.java (class extends DocumentFactory abstract class)
 7. WordDocument.java (class implements Document interface)
 8. WordDocumentFactory.java (class extends DocumentFactory abstract class)
 9. DocumentFactoryTest.java (Client or Driver class to test)

Result:

```
Enter document type either pdf, word or excel: word
Save Word document...
Open Word document...
```

Other exercises

Creational Design Patterns

1. Builder

- Visit my GitHub source code here: [GitHub](#)
- **Results:**

```
Basic Computer: Computer CPU=Intel i5, RAM=8GB, Storage=null, GraphicsCard=null  
Gaming Computer: Computer CPU=AMD Ryzen 9, RAM=32GB, Storage=1TB SSD, GraphicsCard=NVIDIA RTX 4080
```

Structural Design Patterns

1. Adapter

- Visit my GitHub source code here: [GitHub](#)
- **Results:**

```
Payment of Rs.100.0 processed using Stripe.  
Payment of Rs.150.0 sent using PayPal.
```

2. Decorator

- Visit my GitHub source code here: [GitHub](#)
- **Results:**

```
Sending Email notification: Hello! How have you been?  
Sending SMS notification: Hello! How have you been?  
Sending SLACK notification: Hello! How have you been?
```

3. Proxy

- Visit my GitHub source code here: [GitHub](#)

- **Results:**

```
Image not cached. Loading...
Loading image from remote server: img1.jpg
Displaying image: img1.jpg

Using cached image: img1.jpg
Displaying image: img1.jpg
```

Behavioral Design Patterns

1. Strategy

- Visit my GitHub source code here: [GitHub](#)
- **Results:**

```
Paid $250.0 using Credit Card (123-456)
Paid $150.0 using PayPal (harsha@gmail.com)
```

2. Observer

- Visit my GitHub source code here: [GitHub](#)
- **Results:**

```
MobileApp: User1 registered.
WebApp: User2 registered.

Stock price updated to: Rs.100.25
User1 (Mobile App) received stock update: Rs.100.25
User2 (Web App) received stock update: Rs.100.25

Stock price updated to: Rs.102.75
User1 (Mobile App) received stock update: Rs.102.75
User2 (Web App) received stock update: Rs.102.75
WebApp: User2 removed.

Stock price updated to: Rs.98.5
User1 (Mobile App) received stock update: Rs.98.5
```

3. Command

- Visit my GitHub source code here: [GitHub](#)
- **Results:**

```
The light is ON.  
The light is OFF.
```

4. MVC

- Visit my GitHub source code here: [GitHub](#)
- **Results:**

```
Customer ID    : 001  
Customer Name  : Harsha  
Customer Email: harsha@gmail.com  
  
Customer not found with ID: C003
```

References:

1. <https://refactoring.guru/design-patterns>
2. <https://www.geeksforgeeks.org/system-design/java-design-patterns/>