Practical 04

1. Demonstrate how to use the “II” operator for combining the two pointcut expressions for Employee class.

Code:

Employee.java

```
package mypack;

public class Employee {
    private String name;
    private int id;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

EmployeeAspect.java

```
package mypack;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class EmployeeAspect {
    @Pointcut("execution(* mypack.Employee.get*(..))")
    public void getterMethods() {}

    @Pointcut("execution(* mypack.Employee.set*(..))")
    public void setterMethods() {}

    @Before("getterMethods() || setterMethods()")
    public void beforeGetterOrSetter() {
        System.out.println("Aspect triggered: Before getter or setter method execution.");
    }
}
```

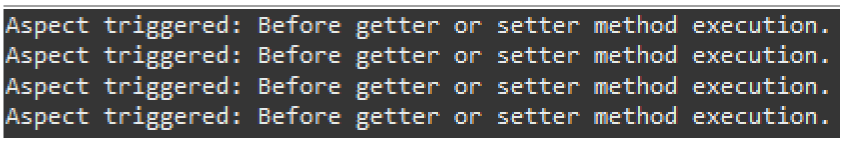
AOPMain.java

```
package mypack;

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

public class AOPMain {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        Employee employee = context.getBean(Employee.class);
        employee.setName("John Doe");
        employee.getName();
        employee.setId(1001);
        employee.getId();
    }
}
```

applicationContext.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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <aop:aspectj-autoproxy />
    <bean id="employee" class="mypack.Employee" />
    <bean id="employeeAspect" class="mypack.EmployeeAspect" />
</beans>
```

Output:



```
Aspect triggered: Before getter or setter method execution.
Aspect triggered: Before getter or setter method execution.
Aspect triggered: Before getter or setter method execution.
Aspect triggered: Before getter or setter method execution.
```

2. Create and apply a before advice for logging with the help of LoggerFactory class and advice should execute for the public method only.

Code
Employee.java

```
package q2;

public class Employee {
    private String name;
    private int id;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public void displayInfo() {
        System.out.println("Employee Info: Name = " + name + ", ID = " + id);
    }
}
```

EmployeeAspect,java

```
package q2;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@Aspect
public class EmployeeAspect {

    private static final Logger logger = LoggerFactory.getLogger(EmployeeAspect.class);

    @Before("execution(public * q2.Employee.*(..))")
    public void logBeforeMethodExecution() {
        logger.info("A public method in Employee is about to execute.");
    }
}
```

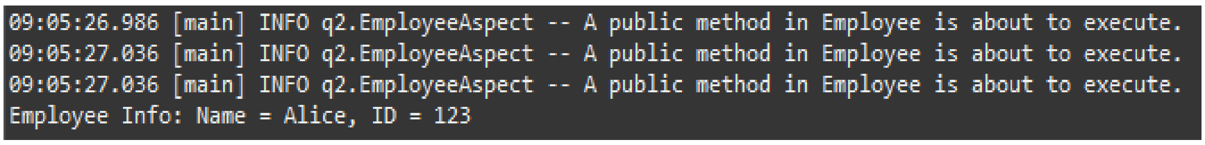
AOPMain.java

```
package q2;

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

public class AOPMain {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("question2.xml");

        Employee employee = context.getBean(Employee.class);
        employee.setName("Alice");
        employee.setId(123);
        employee.displayInfo();
    }
}
```

question2.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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <aop:aspectj-autoproxy />

    <bean id="employee" class="q2.Employee" />

    <bean id="employeeAspect" class="q2.EmployeeAspect" />
</beans>
```

Output:



```
09:05:26.986 [main] INFO q2.EmployeeAspect -- A public method in Employee is about to execute.
09:05:27.036 [main] INFO q2.EmployeeAspect -- A public method in Employee is about to execute.
09:05:27.036 [main] INFO q2.EmployeeAspect -- A public method in Employee is about to execute.
Employee Info: Name = Alice, ID = 123
```

3. Write a program to demonstrate Spring AOP – pointcuts for Image class.

Code:

Image.java

```
package q3;

public class Image {
    public void loadImage() {
        System.out.println("Image is being loaded.");
    }
    public void displayImage() {
        System.out.println("Image is being displayed.");
    }
    public void deleteImage() {
        System.out.println("Image is being deleted.");
    }
}
```

ImageAspect.java

```
package q3;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class ImageAspect {

    @Pointcut("execution(* q3.Image.load*(..))")
    public void loadMethods() {}

    @Pointcut("execution(* q3.Image.display*(..))")
    public void displayMethods() {}

    @Pointcut("execution(* q3.Image.delete*(..))")
    public void deleteMethods() {}

    @Before("loadMethods()")
    public void beforeLoad() {
        System.out.println("Before loading the image.");
    }

    @Before("displayMethods()")
    public void beforeDisplay() {
        System.out.println("Before displaying the image.");
    }

    @Before("deleteMethods()")
    public void beforeDelete() {
        System.out.println("Before deleting the image.");
    }
}
```

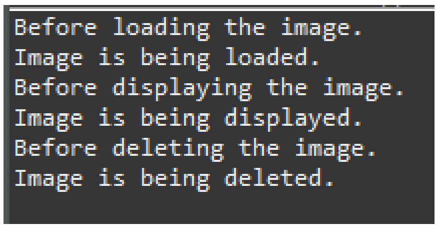
AOPMain.java

```
package q3;

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

public class AOPMain {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("questin3.xml");

        Image image = context.getBean(Image.class);

        image.loadImage();
        image.displayImage();
        image.deleteImage();
    }
}
```

questin3.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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <aop:aspectj-autoproxy />

    <bean id="image" class="q3.Image" />

    <bean id="imageAspect" class="q3.ImageAspect" />
</beans>
```

Output:



```
Before loading the image.
Image is being loaded.
Before displaying the image.
Image is being displayed.
Before deleting the image.
Image is being deleted.
```

4. Demonstrate how to use the “II” operator for combining the two pointcut expressions for Order class.

Code:

Order.java

```
package q4;

public class Order {
    public void placeOrder() {
        System.out.println("Order has been placed.");
    }
    public void cancelOrder() {
        System.out.println("Order has been canceled.");
    }
    public void trackOrder() {
        System.out.println("Tracking order.");
    }
}
```

OrderAspect.java

```
package q4;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class OrderAspect {
    @Pointcut("execution(* q4.Order.place*(..))")
    public void placeMethods() {}
    @Pointcut("execution(* q4.Order.cancel*(..))")
    public void cancelMethods() {}

    @Before("placeMethods() || cancelMethods()")
    public void logBeforePlaceOrCancel() {
        System.out.println("Logging: Order is being placed or canceled.");
    }
}
```

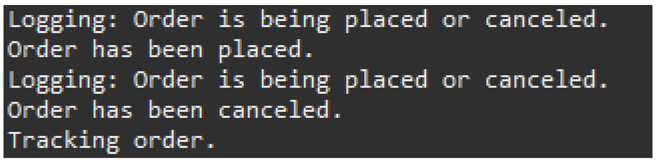
AOPMain.java

```
package q4;

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

public class AOPMain {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("question4.xml");

        Order order = context.getBean(Order.class);
        order.placeOrder();
        order.cancelOrder();
        order.trackOrder();
    }
}
```

Question4.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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">
    <aop:aspectj-autoproxy />

    <bean id="order" class="q4.Order" />
    <bean id="orderAspect" class="q4.OrderAspect" />
</beans>
```

Output:



```
Logging: Order is being placed or canceled.
Order has been placed.
Logging: Order is being placed or canceled.
Order has been canceled.
Tracking order.
```

5. Write a program to demonstrate Spring AOP – after advice for Order class.

Code:

Order.java

```
package q5;

public class Order {
    public void placeOrder() {
        System.out.println("Order has been placed.");
    }

    public void cancelOrder() {
        System.out.println("Order has been canceled.");
    }

    public void trackOrder() {
        System.out.println("Order is being tracked.");
    }
}
```

OrderAspect.java

```
package q5;

import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
@Aspect
public class OrderAspect {
    @After("execution(* q5.Order.*(..))")
    public void logAfterMethodExecution() {
        System.out.println("Logging: A method in Order class has been executed.");
    }
}
```

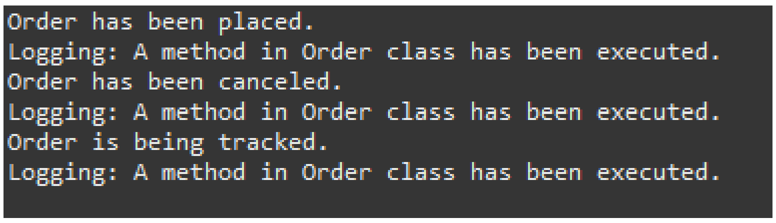
AOPMain.java

```
package q5;

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

public class AOPMain {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("question5.xml");

        Order order = context.getBean(Order.class);

        order.placeOrder();
        order.cancelOrder();
        order.trackOrder();
    }
}
```

question5.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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">
    <aop:aspectj-autoproxy />
    <bean id="order" class="q5.Order" />
    <bean id="orderAspect" class="q5.OrderAspect" />
</beans>
```

Output:



```
Order has been placed.
Logging: A method in Order class has been executed.
Order has been canceled.
Logging: A method in Order class has been executed.
Order is being tracked.
Logging: A method in Order class has been executed.
```