Exercise 1. Assume that there exists a Line class within package FigureElement which takes two FigureElement.Points as arguments. Write an aspect called CheckDegenerateLines that throws an exception DegenerateLineException when the constructor of FigureElement.Line is called with identical arguments.

Exercise 2. Consider the following class:

```java
public class TestWithin {
    static class Outer {
        public void one() {}

        public void two() {
            one();
        }

        public void three() {
            one();
            two();
        }

        public static void main(String[] args) {
            new Outer().three();
            new Outer().one();
        }
    }

    Write the output of the program for the aspect:

    before(): call(* one()) && withincode(public void three()) {
        System.out.println("Pointcut matched.");
    }

    Explain where and why the pointcuts are matched.
```
Exercise 3. Write an aspect that picks out any call to the add* methods of Java’s List and counts the total number of calls made.

Exercise 4. Suppose that there exists a package called mypkg. Write an aspect that picks out any call to methods in classes of mypkg or in any of its sub-packages, and throws an exception in the following cases:

- Any argument which is an object reference is null;
- Any integer argument is negative.

Exercise 5. Consider the following class:

class FigureElement {
    private int x;
    private double y;

    public FigureElement(int x, double y) {
        this.x = x;
        this.y = y;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(double y) {
        this.y = y;
    }
}

1. Write an aspect that logs whenever any private field within FigureElement is set outside the constructor.

2. Modify the aspect to contain information about the class and field name, the new value, and the location from which it was changed.

Exercise 6. Consider the following class:

public class ECS {
    // assume marks between 0% and 100%
    public void transferMark(String id, double mark) {
        System.out.println("Recorded mark " + mark +
                         "% for student " + id);
    }
}

Write an aspect that intercepts all calls made to ECS.transferMark and increases all marks by 5 percent.
Exercise 7. Describe in words the joinpoints matched by the following pointcut designators:

1. call(public * java.lang.*ss.is*())
2. call(* String.*(..)) && args(*, int)
3. call(java.io..new(..))

Exercise 8. Consider the following class:

```java
public class A {
    private int k = 1;

    public A() { }

    public void doOne() {
        if (k-- != 0)
            doOne();
        else
            doTwo();
    }

    public void doTwo() { return; }

    public static void main(String[] args) {
        A someA = new A();
        someA.doOne();
    }
}
```

Explain what happens and write the output of the program when each of the following advices is applied.

Advice 1:
```java
before(): call(* A.*(..)) && cflow(call(* doOne())) {
    System.out.println(thisJoinPoint.getSignature());
}
```

Advice 2:
```java
before(): call(* A.*(..)) && cflow(call(* doTwo())) {
    System.out.println(thisJoinPoint.getSignature());
}
```

Advice 3:
```java
before(): call(* A.*(..)) && cflowbelow(call(* doTwo())) {
    System.out.println(thisJoinPoint.getSignature());
}
```