Persistent Aspect

```
public class Database {
	public void save(int result) {
		System.out.println("saving result: " + result + " to the database");
	}
}
```

With AspectJ

```
public aspect PersistentAspect {
	Database database = new Database();
	static final int limit = 10;
	pointcut methodExecution(): execution(* CalculatorInterface.*(..));
	after() returning(int result): methodExecution() {
		if(result > limit)
			database.save(result);
	}
}
```

With DynamicProxy

```
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
public class InvocationHandlerExample implements InvocationHandler {
    private final Object target;
    private final Database database = new Database();
    private static final int limit = 10;
    public InvocationHandlerExample(Object target) {
        this.target = target;
    }
    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        Object result = method.invoke(target, args);
        if (result instanceof Integer && (Integer) result > THRESHOLD) {
            database.save((Integer) result);
        }
        return result;
    }
}
import java.lang.reflect.Proxy;
public class PersistentAspect {
    public static void main(String[] args) {
        new PersistentAspect().go();
    }
    void go() {
        CalculatorInterface calculator = new Calculator();
        CalculatorInterface proxy = (CalculatorInterface) Proxy.newProxyInstance(
                CalculatorInterface.class.getClassLoader(),
                new Class[]{CalculatorInterface.class},
                new InvocationHandlerExample(calculator)
        );
        proxy.add(6, 7));  // This will be saved
        proxy.subtract(5, 2);  // This will not be saved
    }
}
```