Dynamic Proxy

```
public interface DinamicProxyInterface {
	void sayHello();
}

public class DinamicProxyConcreteClass implements DinamicProxyInterface {
	@Override
	public void sayHello() {
		System.out.println("Hello, World");
	}
}
```

Proxy

```
import java.lang.reflect.Proxy;
public class DynamicProxyExample {

	public static void main(String[] args) {
		new DynamicProxyExample ().go();
	}

	void go() {
		// creare l'oggetto target
		DinamicProxyInterface hello = new DinamicProxyConcreteClass();

		// creare l'handler di invocazione
		InvocationHandlerExample handler = new InvocationHandlerExample(hello);

		// creare il proxy
		DinamicProxyInterface proxy = (DinamicProxyInterface) Proxy.newProxyInstance(
				DinamicProxyInterface.class.getClassLoader(),
				new Class[] {DinamicProxyInterface.class},
				handler);

		// usare il proxy
		proxy.sayHello();
	}
}
```

InvocationHandler

```
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
public class InvocationHandlerExample implements InvocationHandler {
	private final Object target;
	public InvocationHandlerExample (Object target) {
		this.target = target;
	}
	@Override
	public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
		System.out.println("u were invoking the method: " + method.getName());
		Object result = method.invoke(target, args);
		System.out.println("success, method (" + method.getName() + ") get invoked");
		return result;
	}
}
```