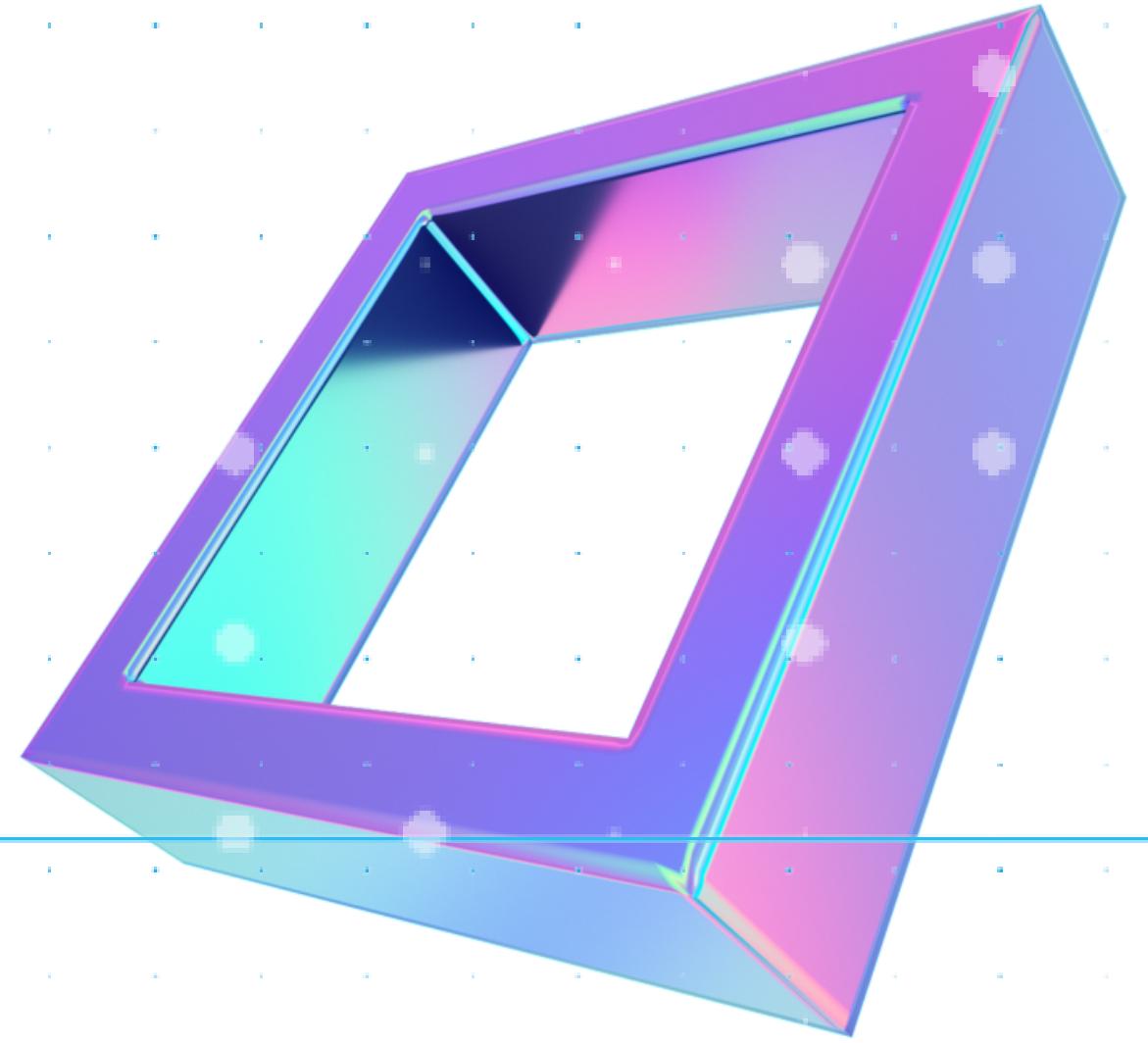


TSUNAMI SECURITY SCANNER





topics

What will we present today.

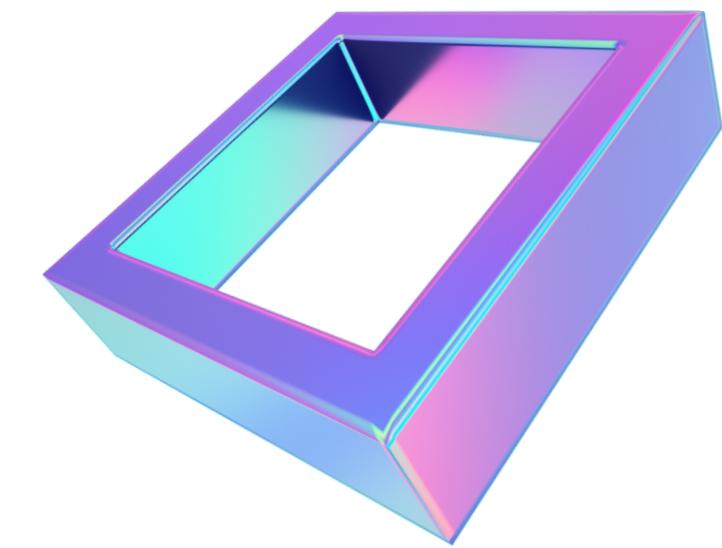
01 : What is Tsunami Security Scanner ?

02 : Tsunami's Architecture

03 : Quality Artributes

04 : Tsunami's Design pattern

05 : Member



The background features abstract, semi-transparent 3D geometric shapes in shades of purple, blue, and pink, including a large rectangular prism on the left and smaller spheres and cylinders on the right.

What is Tsunami Security Scanner ?

THIS??!!

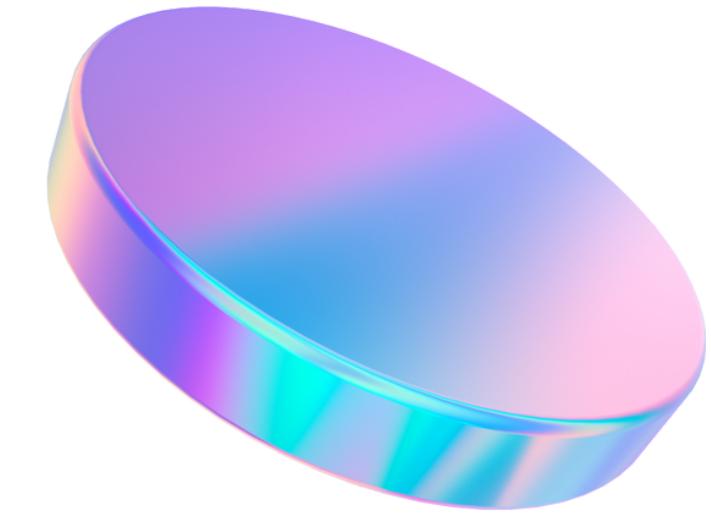




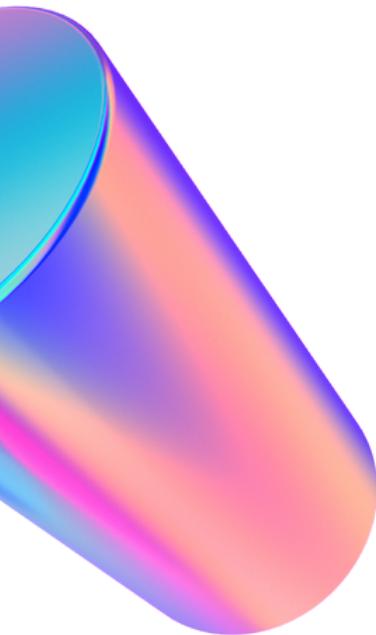


Tsunami Security Scanner!!!

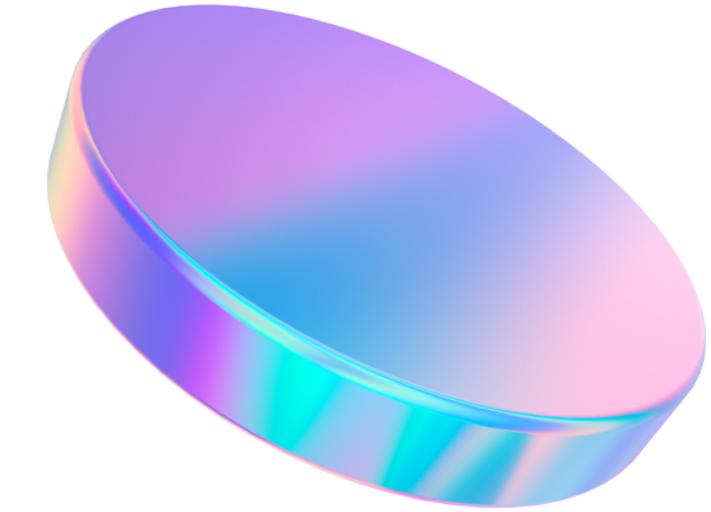
tsunami security scanner ?



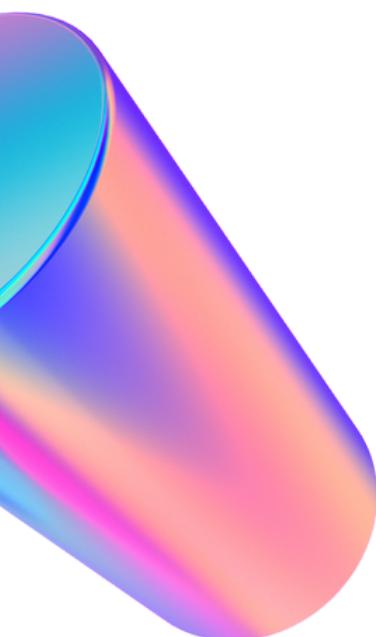
"สึนามิ" ดังกล่าว เป็นแพลตฟอร์มทั่วไปและเป็นสากล ซึ่งมี
การกำหนดฟังก์ชันการทำงานผ่านปลั๊กอิน กล่าวคือมี
ปลั๊กอินสำหรับการสแกนพอร์ตตาม nmap และปลั๊กอิน
สำหรับตรวจสอบพารามิเตอร์การพิสูจน์ตัวตนที่ไม่น่าเชื่อถือ
ตาม Ncrack รวมถึงปลั๊กอินที่มีตัวตรวจสอบช่องโหว่ใน
Hadoop Yarn, Jenkins, Jupyter และ WordPress



tsunami security scanner ?



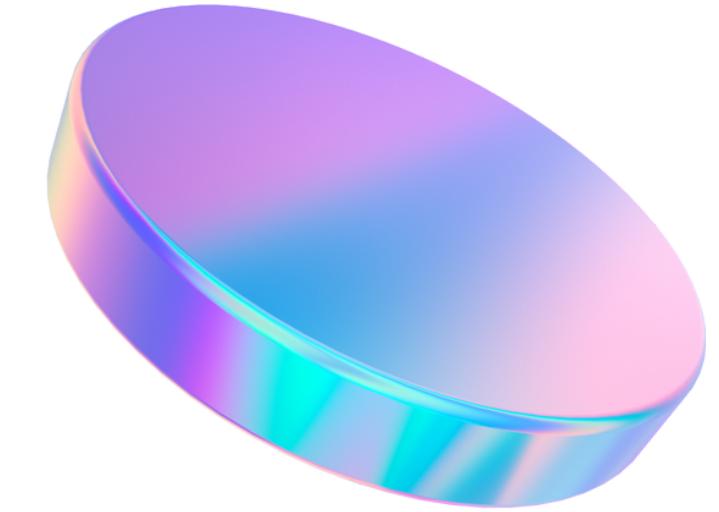
วัตถุประสงค์ ของโครงการคือ เป็นเครื่องมือสำหรับการ
ตรวจหาช่องโหว่อย่างรวดเร็ว ใน บริษัท ขนาดใหญ่ที่มี
โครงสร้างพื้นฐานเครือข่ายที่กว้างขวาง การเปิดเผยข้อมูล
เกี่ยวกับปัญหาใหม่ที่สำคัญทำให้เกิดการแย่งบันกับผู้โจนติที่
ต้องการโจนติโครงสร้างพื้นฐานของธุรกิจก่อนที่ปัญหาจะได้
รับการแก้ไข



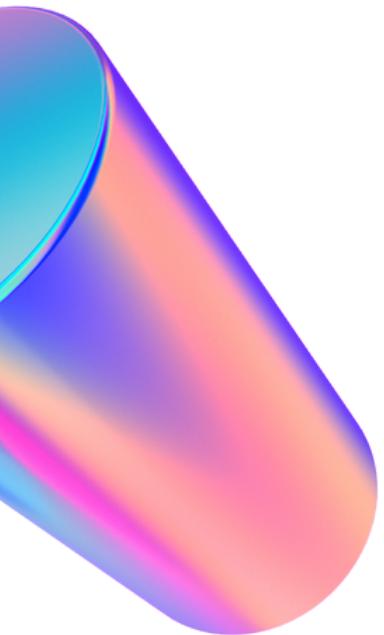
The background features several abstract, translucent 3D geometric shapes in shades of purple, blue, and pink, floating and intersecting in the upper corners.

Tsunami Architecture

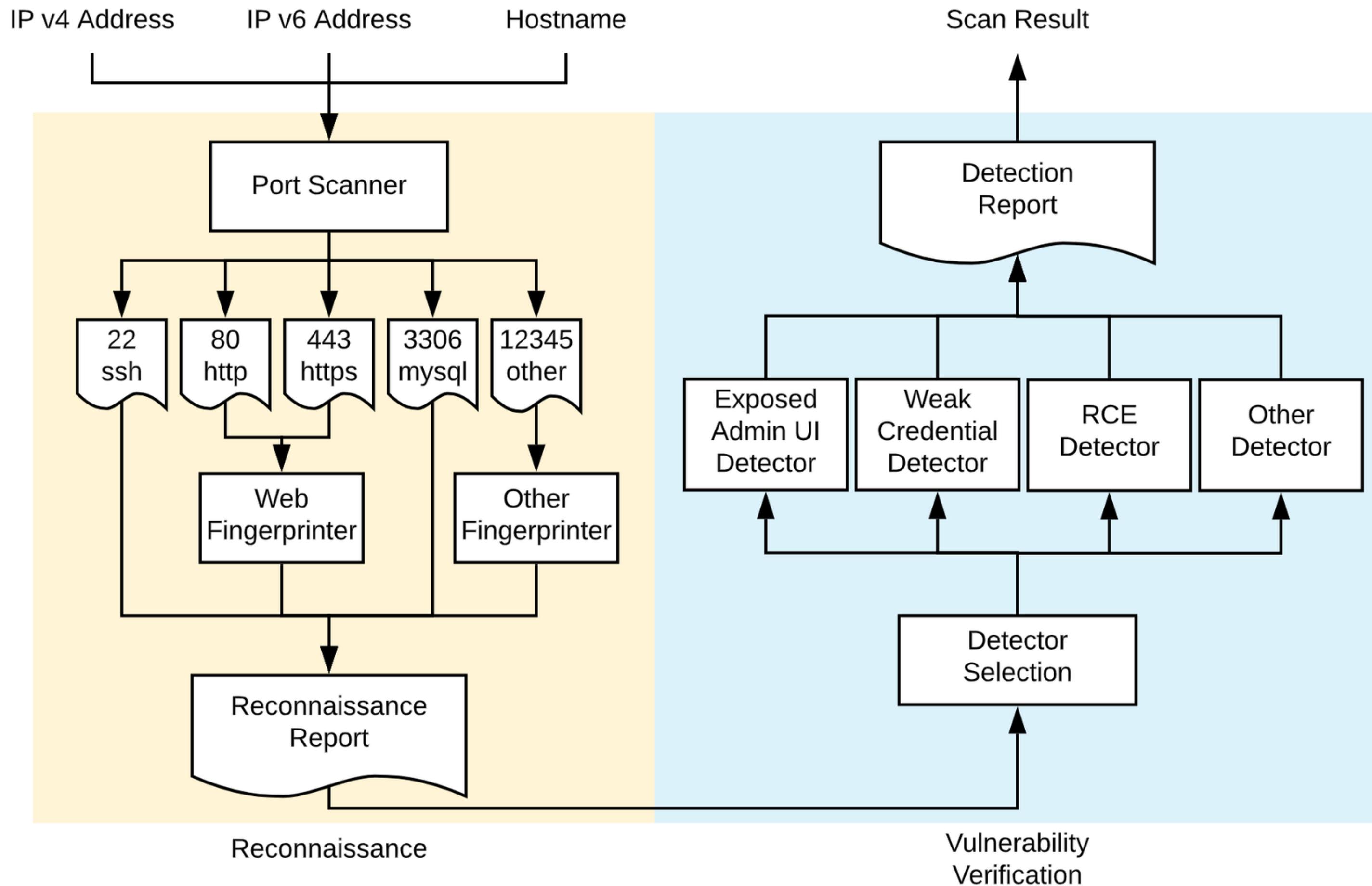
Pipes and Filters's Architecture



Tsunami security scanner มีลักษณะการทำงานแบบ Pipes and filters's Architecture เพราะมีการทำงานของแต่ละลำดับขึ้น โดยมี input และ output และแต่ละชั้นจะมีกระบวนการทำงานของตัวเองแตกต่างกัน และบางชั้นจะมีการทำงานแบบขนานกัน และการที่ขึ้นตอนนึง มีการ fork แยกออกเป็นหลายๆ task ที่สามารถทำงานพร้อมๆ กันได้



Pipes and Filters's Architecture

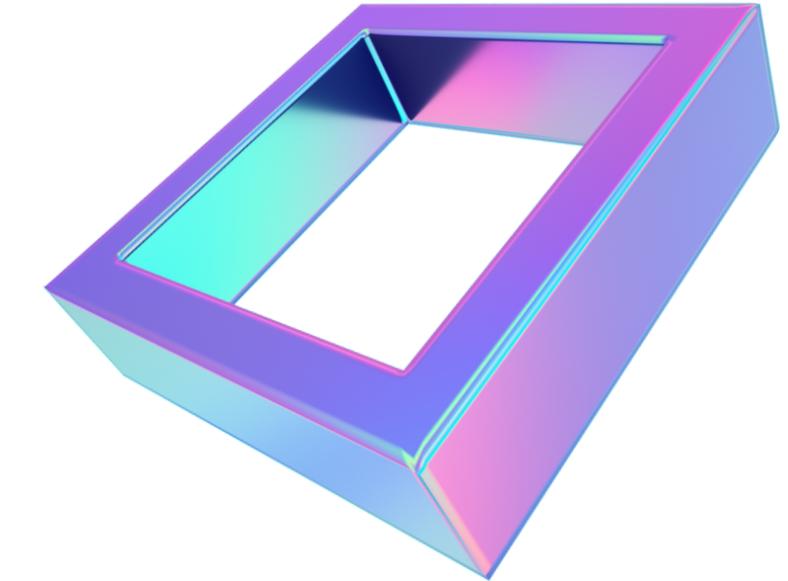


The background features three abstract 3D geometric shapes: a rectangular prism on the left with faces in shades of purple, pink, and blue; a sphere in the top right with a gradient from purple to blue; and a cylinder in the bottom right with a gradient from blue to orange.

Quality Artributes



SECURITY



01

สืบมายเป็นเครื่องสแกนช่องโหว่ด้านความปลอดภัยที่ออกแบบมาสำหรับเครือข่ายขนาดใหญ่มาก ซึ่งเดินสร้างขึ้นโดย Google เพื่อสแกนเครือข่ายขนาดใหญ่ของตนเอง

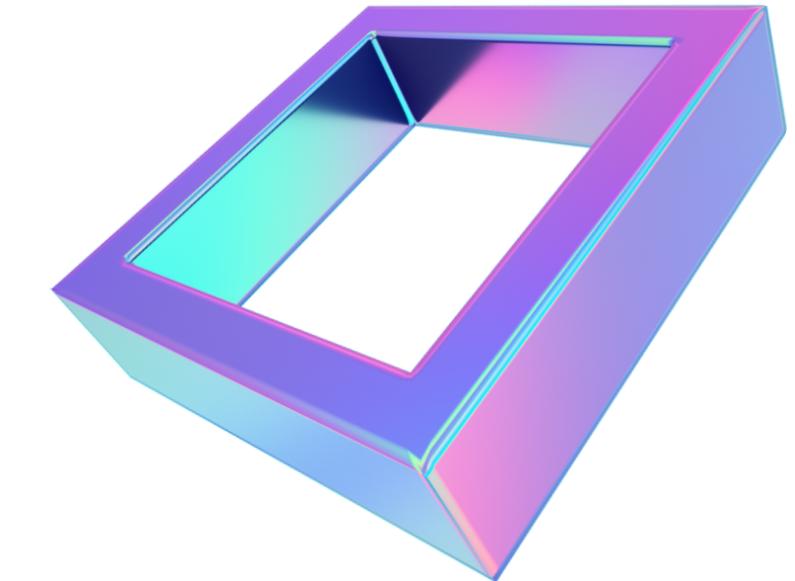
02

ไม่ถูกการตรวจสอบช่องโหว่ยังเป็นวิธีที่สามารถขยายสืบมายผ่านปลั๊กอินซึ่งเป็นวิธีการที่ก่อรักษาความปลอดภัยสามารถเพิ่มเวกเตอร์การโจมตีและช่องโหว่ใหม่ ๆ เพื่อตรวจสอบภัยในเครือข่าย





MODIFIABILITY

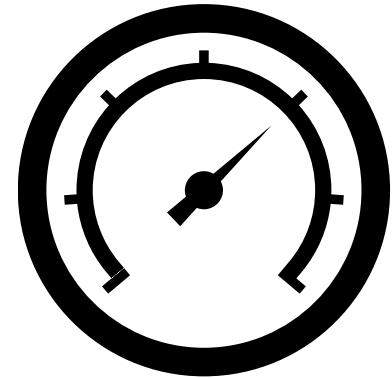


01

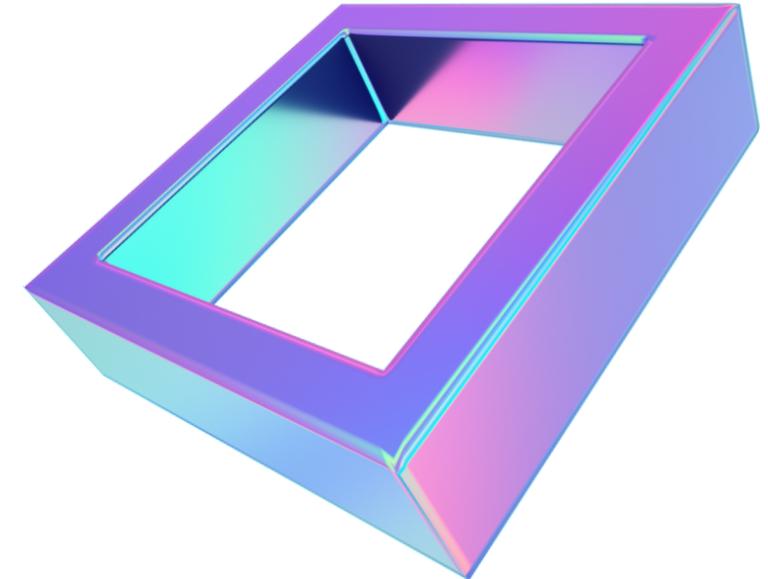
เนื่องจากระบบ Object Port
ขยายชีวิต และแยกออกจากกัน
ดังนั้นหากมีเทคโนโลยีใหม่ๆ ที่
เพิ่ม Port ชนิดใหม่เข้ามาตัว
ระบบสามารถเพิ่ม Object
Port ใหม่ได้ง่าย

02

สามารถแก้ไขแต่ละโมดูลได้ง่ายและ
ไม่กระทบต่อโมดูลอื่นภายในระบบ
โดยยังคงความสามารถสแกน
ระบบได้



PERFORMANCE EFFICIENCY



01

Tsunami อนุญาตให้ผู้ใช้สามารถกำหนดเวลาปริมาณงานหรือดำเนินการคำสั่งของระบบ

02

Tsunami สามารถทำงานเบื้องหลังด้วยตัวเองได้แบบอัตโนมัติ และสามารถสแกนพอร์ตเน็ตเวิร์ก ที่เปิดอยู่ได้อัตโนมัติ

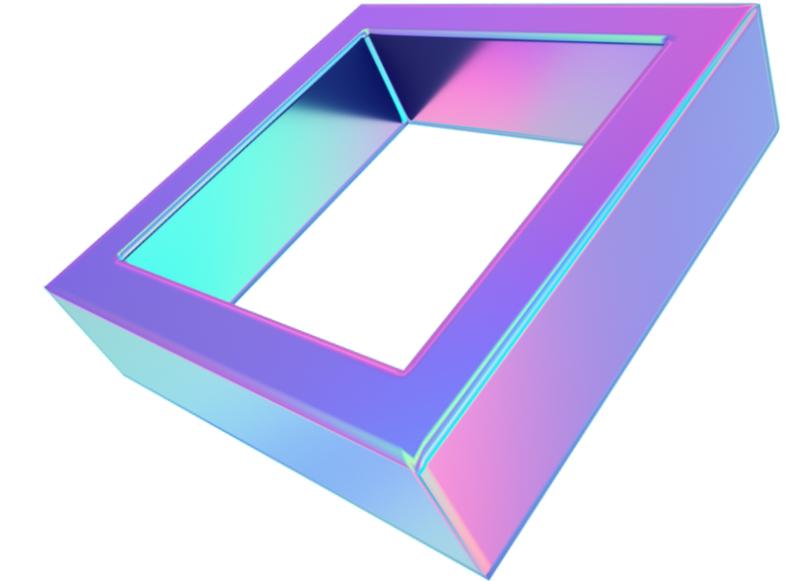
ข้อด้วย และ วิธีแก้ไข

ข้อด้วย

Tsunami ยังทำงานด้วย command line โดยรับ Input และแสดง Output เก่านั้นทำให้ยังมีข้อเสียเรื่อง Usability ที่ผู้ใช้งานยังสามารถเข้าถึงการใช้งานได้ยากอยู่

วิธีแก้ไข

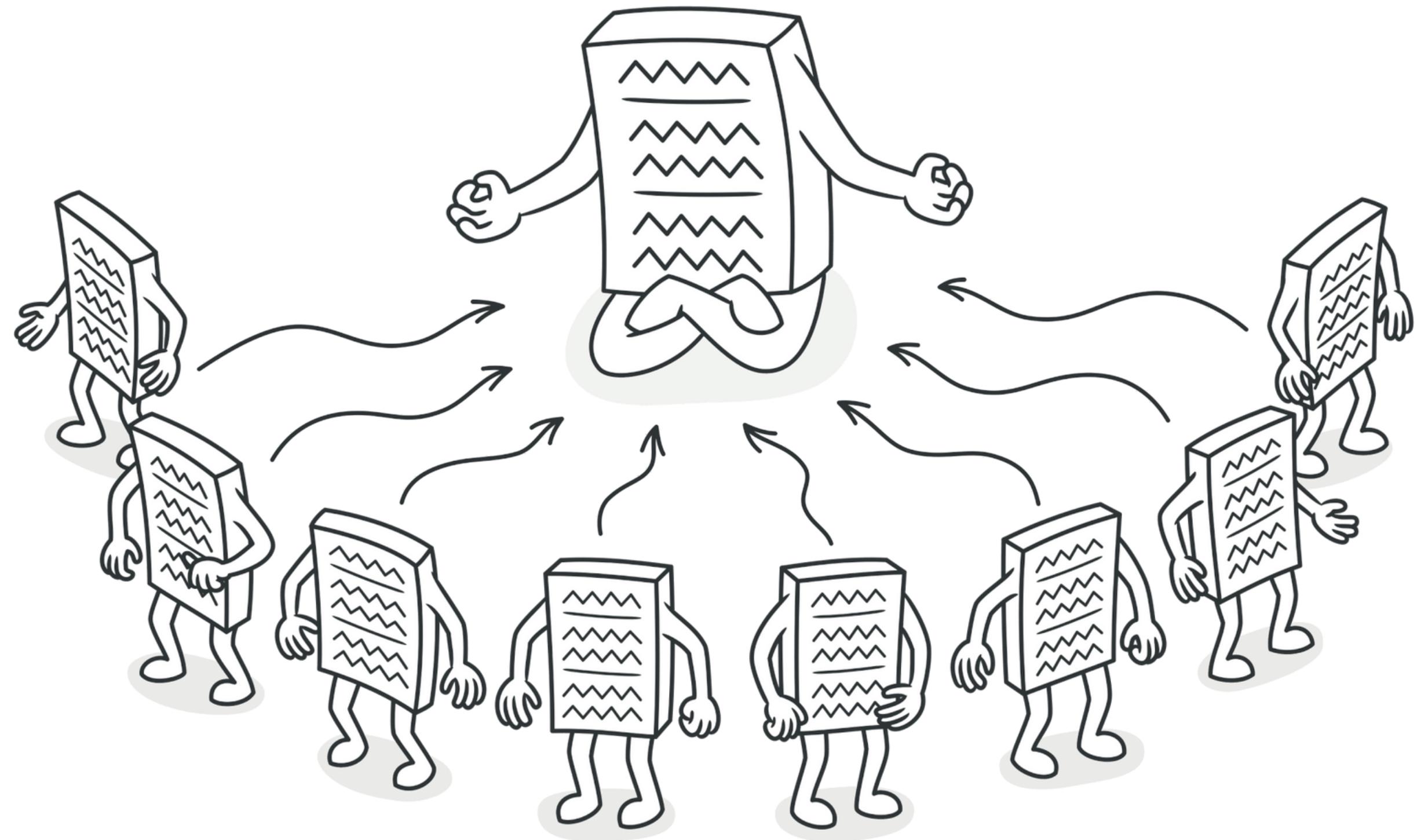
จะต้องออกแบบและสร้างตัวหน้าแอปพลิเคชันเพื่อให้ผู้ใช้งานสะดวกต่อการใช้งานมากขึ้น



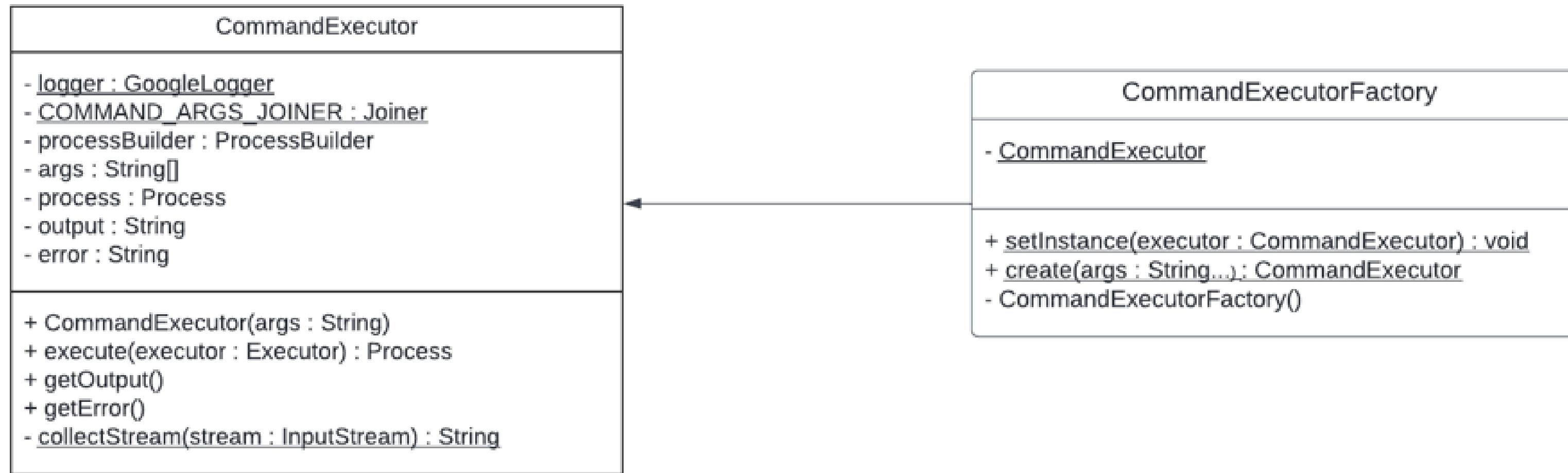
The background features three abstract 3D geometric shapes: a rectangular prism on the left with faces in purple, pink, blue, and white; a sphere in the top right with a gradient from purple to blue; and a cylinder in the bottom right with a gradient from blue to orange.

Tsunami Design Pattern

Singleton



Singleton



Singleton

```
/** Utility class to simplify the creation and testing of {@link CommandExecutor} instances. */
public class CommandExecutorFactory {

    private static CommandExecutor instance;

    /**
     * Sets an executor instance that will be returned by all future calls to {@link
     * CommandExecutorFactory#create(String...)}
     *
     * @param executor The {@link CommandExecutor} returned by this factory.
     */
    public static void setInstance(CommandExecutor executor) {
        instance = executor;
    }

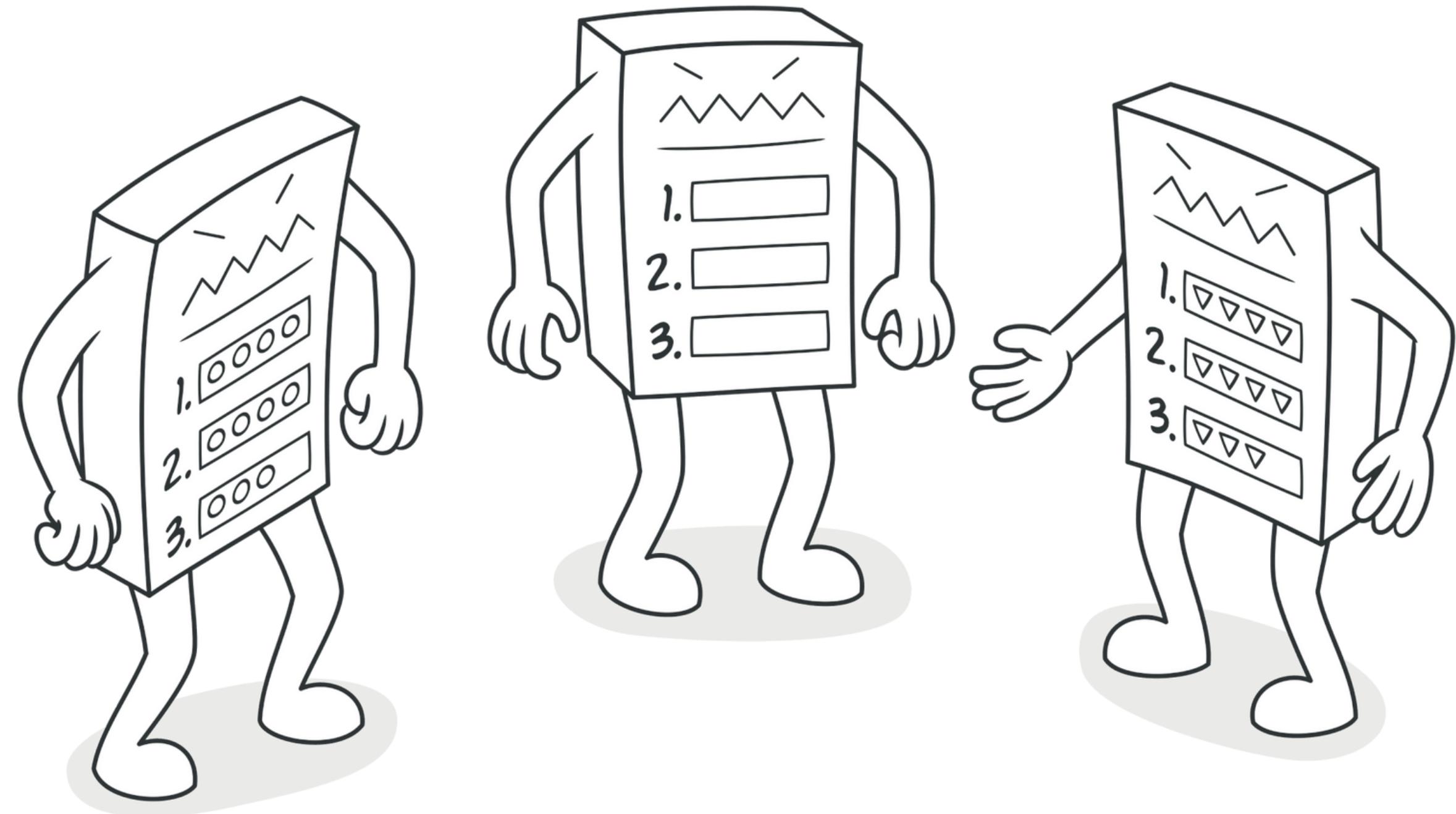
    /**
     * Creates a new {@link CommandExecutor} if none is set.
     *
     * @param args List of arguments to pass to the newly created {@link CommandExecutor}.
     * @return the {@link CommandExecutor} instance created by this factory.
     */
    public static CommandExecutor create(String... args) {
        if (instance == null) {
            return new CommandExecutor(args);
        }
        return instance;
    }

    private CommandExecutorFactory() {}
}
```

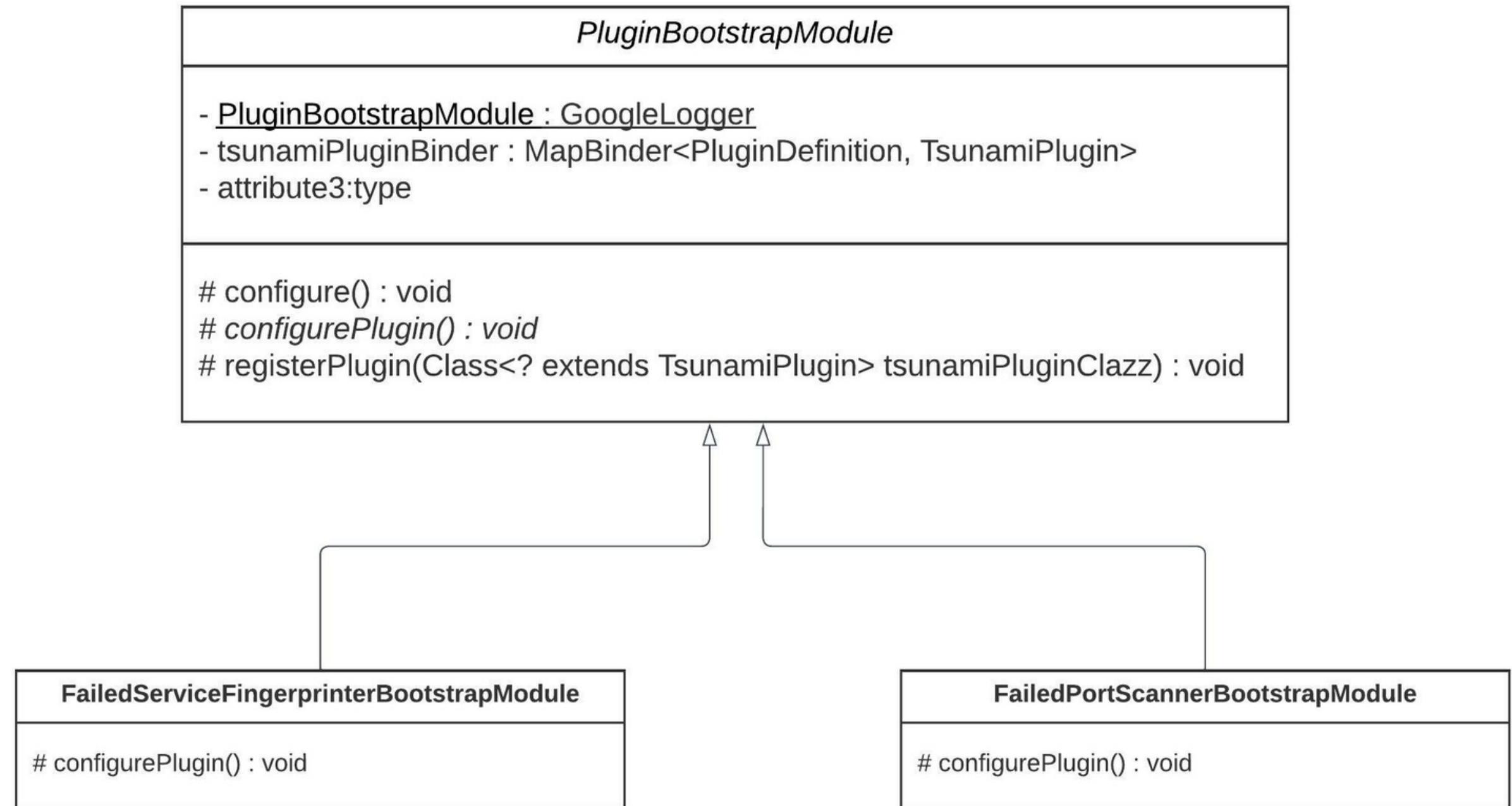
Singleton

จาก Source Code มีการใช้งาน Singleton ที่ไฟล์ CommandExecutorFactory.java ที่มีการกำหนดค่า instance เป็นตัวแปรเดียวที่ใช้สำหรับทุกๆ Object และการประกาศให้ Constructor เป็น private

Template Method



Template Method



Template Method

```
public abstract class PluginBootstrapModule extends AbstractModule {
    private static final GoogleLogger logger = GoogleLogger.forEnclosingClass();

    private MapBinder<PluginDefinition, TsunamiPlugin> tsunamiPluginBinder;

    @Override
    protected final void configure() {
        tsunamiPluginBinder =
            MapBinder.newMapBinder(binder(), PluginDefinition.class, TsunamiPlugin.class);
        configurePlugin();
    }

    /**
     * All bootstrapping logic for a {@link TsunamiPlugin} should be implemented in this method.
     * {@link PluginBootstrapModule#registerPlugin(Class)} must also be called in order to register
     * the plugin to Tsunami.
     */
    protected abstract void configurePlugin();

    /**
     * Register a {@link TsunamiPlugin} to Tsunami's plugin module using Guice's multibinding feature.
     *
     * @param tsunamiPluginClazz the {@link Class} for the {@link TsunamiPlugin} to be registered.
     */
    protected final void registerPlugin(Class<? extends TsunamiPlugin> tsunamiPluginClazz) {
       .checkNotNull(tsunamiPluginClazz);

        tsunamiPluginBinder
            .addBinding(PluginDefinition.forPlugin(tsunamiPluginClazz))
            .to(tsunamiPluginClazz);
        logger.atInfo().log("Plugin %s is registered.", tsunamiPluginClazz);
    }
}
```

```
/** Bootstrapping module for {@link FailedPortScanner}. */
public final class FailedPortScannerBootstrapModule extends PluginBootstrapModule {

    @Override
    protected void configurePlugin() {
        registerPlugin(FailedPortScanner.class);
    }
}
```

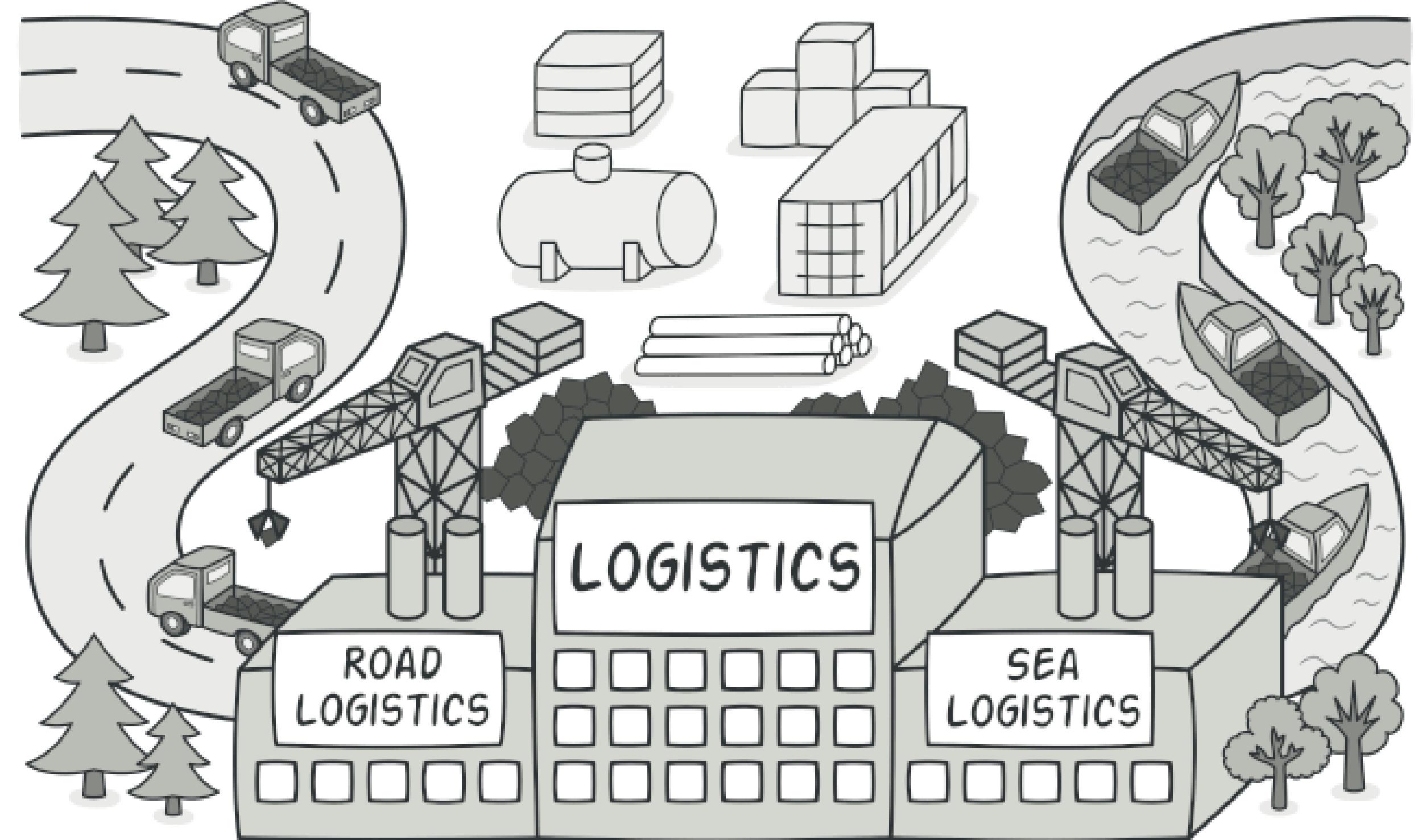
```
public final class FailedServiceFingerprinterBootstrapModule extends PluginBootstrapModule {

    @Override
    protected void configurePlugin() {
        registerPlugin(FailedServiceFingerprinter.class);
    }
}
```

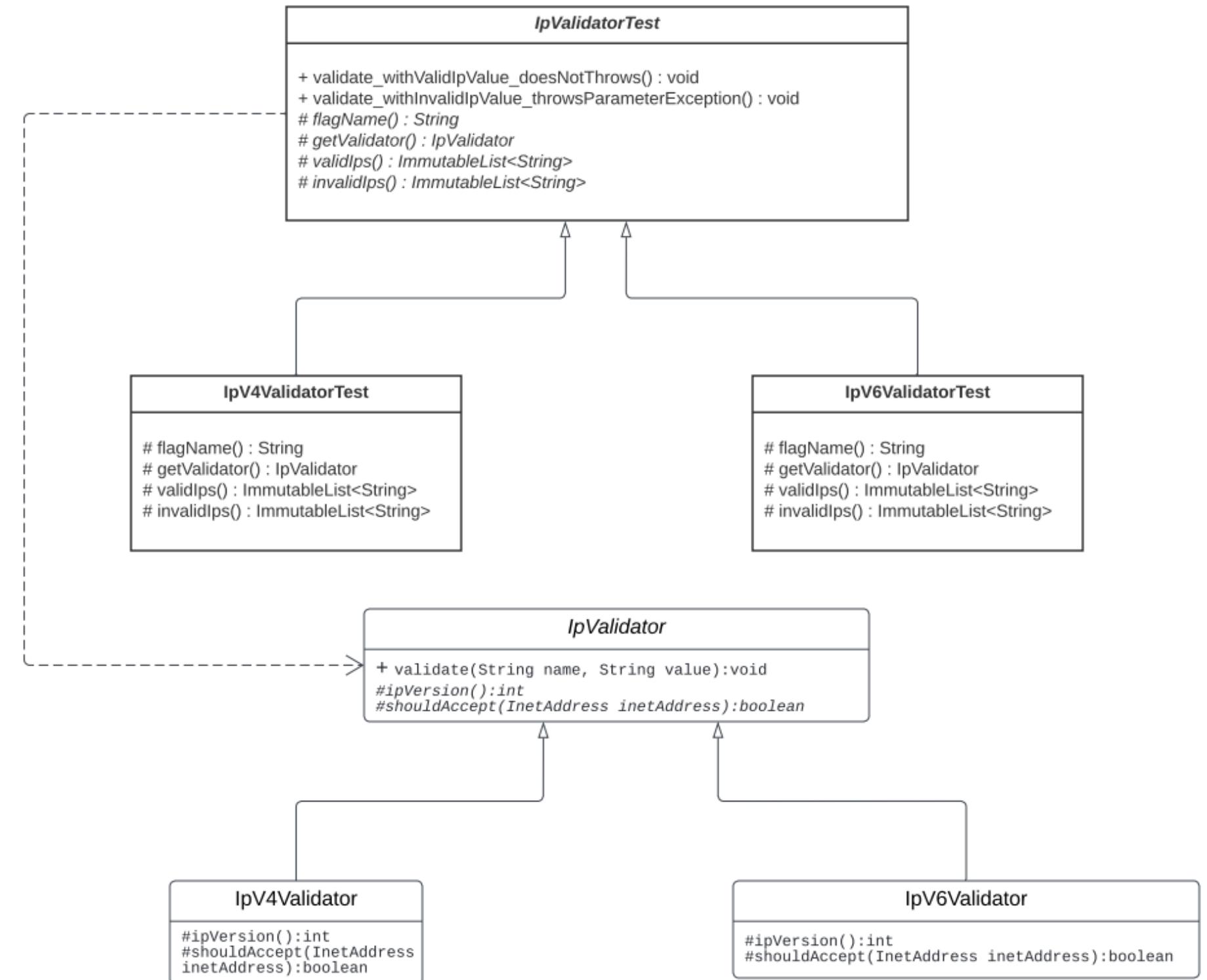
Template Method

จาก Source Code มีการใช้งาน Template method โดยจะเห็น Subclass FailedServiceFingerprinterBootstrapModule และ FailedPortScannerBootstrapModule ที่มีการเปลี่ยนแปลงการทำงาน ใน method configurePlugin ถึงแม้ว่า method ที่ชื่อ configurePlugin ที่ทำหน้าที่เป็น Template Method อาจจะมี code ลำดับการทำงานไม่เยอะ แต่ก็นับว่าเป็น Template Method

Factory Method



Factory Method



Factory Method

```
package com.google.tsunami.main.cli.option.validator;

import com.google.common.collect.ImmutableList;
import org.junit.runner.RunWith;
import org.junit.runners.JUnit4;

/** Tests for {@link IPv6Validator}. */
@RunWith(JUnit4.class)
public class IPv6ValidatorTest extends IpValidatorTest {

    @Override
    protected String flagName() {
        return "ip-v6-target";
    }

    @Override
    protected Ipvalidator getvalidator() {
        return new IPv6Validator();
    }

    @Override
    protected ImmutableList<String> validIps() {
        return ImmutableList.of(
            "0:e:e:0:0:0:1",
            "fe80::a",
            "fe80::1",
            "fe80::2",
            "fe80::42",
            "fe80:3dd0:7f8e:57b7:34d5",
            "fe80:3dd0:7f8e:57b7:0:0:0",
            "::4:0:0:ffff",
            "0:0:3::ffff",
            "7::0.128.0.127");
    }

    @Override
    protected ImmutableList<String> invalidIps() {
        return ImmutableList.of(
            "", "bogus_string", "1234", "127.0.0.1", "www.google.com", "[1:2e]", "[fe80:a]");
    }
}
```

```
package com.google.tsunami.main.cli.option.validator;

import static com.google.common.truth.Truth.assertThat;
import static org.junit.Assert.assertThrows;

import com.beust.jcommander.ParameterException;
import com.google.common.collect.ImmutableList;
import org.junit.Test;

/** Base class for IP validators. */
public abstract class IpValidatorTest {

    @Test
    public void validate_withValidIpValue_doesNotThrows() {
        for (String validIp : validIps()) {
            try {
                getValidator().validate(flagName(), validIp);
            } catch (ParameterException e) {
                throw new AssertionError("Unexpected ParameterException for IP: " + validIp, e);
            }
        }
    }

    @Test
    public void validate_withInvalidIpValue_throwsParameterException() {
        for (String invalidIp : invalidIps()) {
            ParameterException exception =
                assertThrows(
                    ParameterException.class, () -> getValidator().validate(flagName(), invalidIp));
            assertThat(exception)
                .hasMessageThat()
                .isEqualTo(
                    String.format(
                        "Parameter %s should point to a valid IP v%d address, got '%s'",
                        flagName(), getValidator().ipVersion(), invalidIp));
        }
    }

    protected abstract String flagName();

    protected abstract IpValidator getValidator();

    protected abstract ImmutableList<String> validIps();

    protected abstract ImmutableList<String> invalidIps();
}
```

```
package com.google.tsunami.main.cli.option.validator;

import com.google.common.collect.ImmutableList;
import org.junit.runner.RunWith;
import org.junit.runners.JUnit4;

/** Tests for {@link IPv4Validator}. */
@RunWith(JUnit4.class)
public class IPv4ValidatorTest extends IpValidatorTest {

    @Override
    protected String flagName() {
        return "ip-v4-target";
    }

    @Override
    protected IpValidator getValidator() {
        return new IPv4Validator();
    }

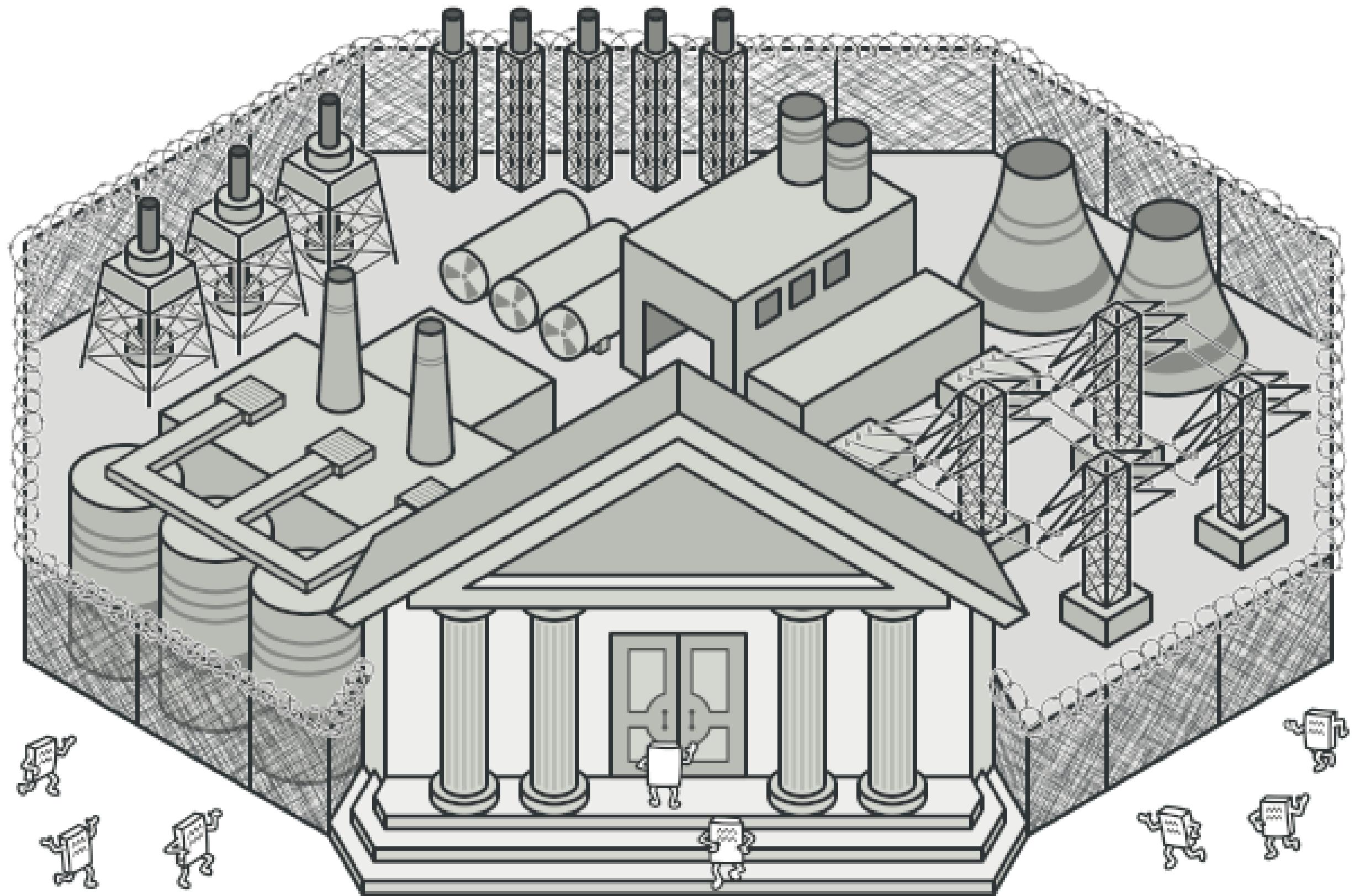
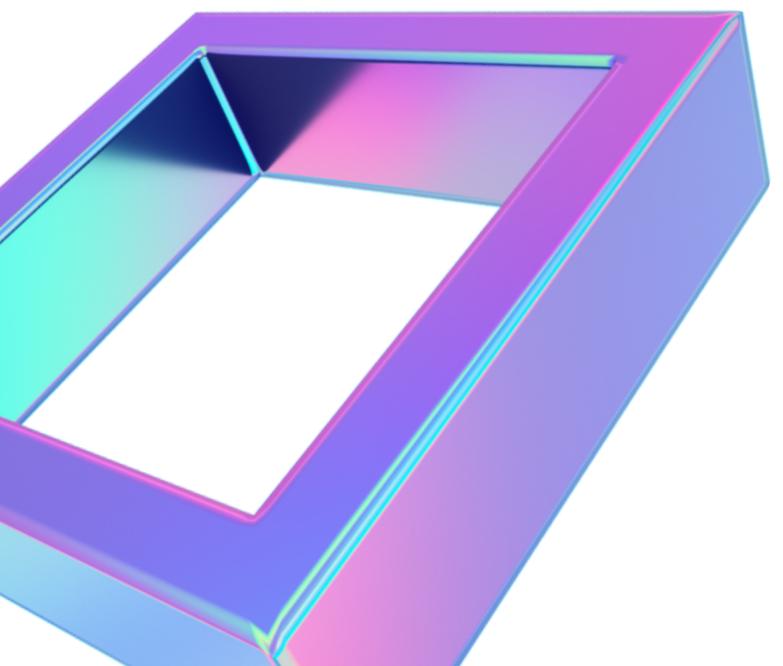
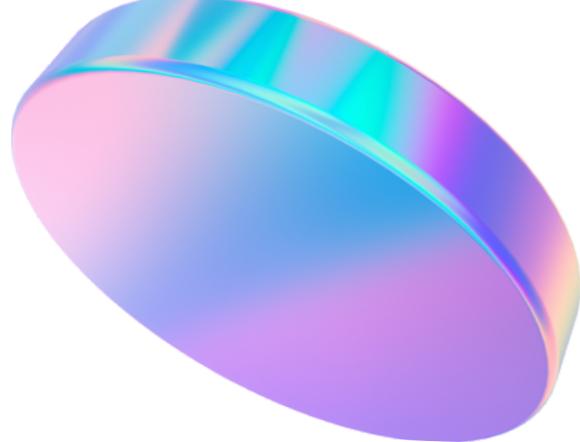
    @Override
    protected ImmutableList<String> validIps() {
        return ImmutableList.of("127.0.0.1", "8.8.8.8");
    }

    @Override
    protected ImmutableList<String> invalidIps() {
        return ImmutableList.of("", "bogus_string", "1234", "2002:af4:9b91::", "www.google.com");
    }
}
```

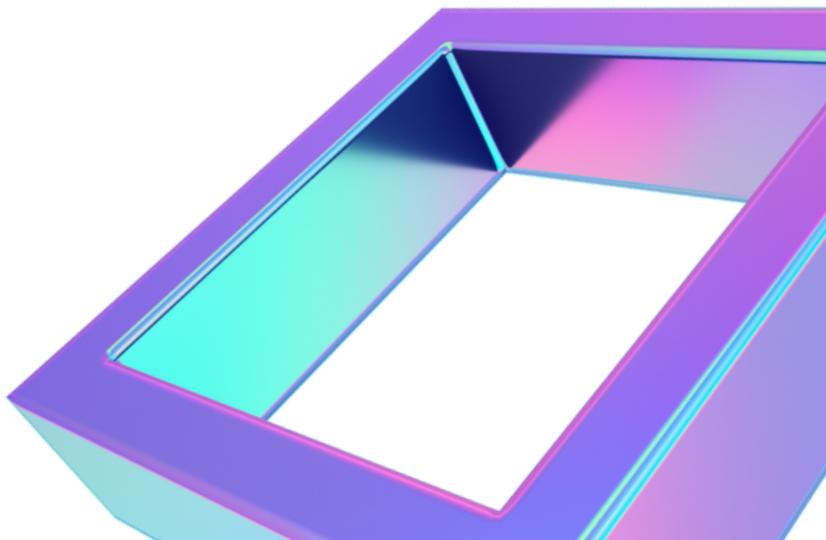
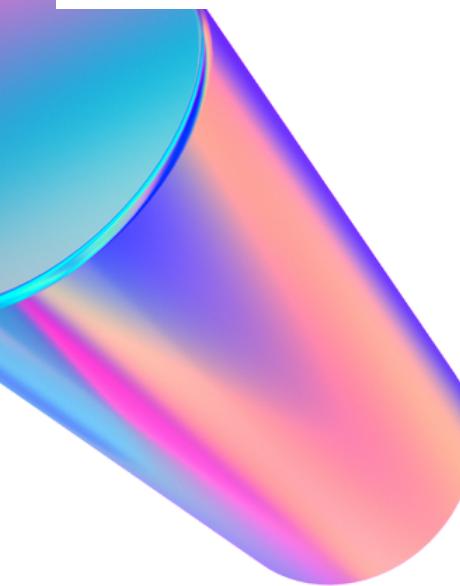
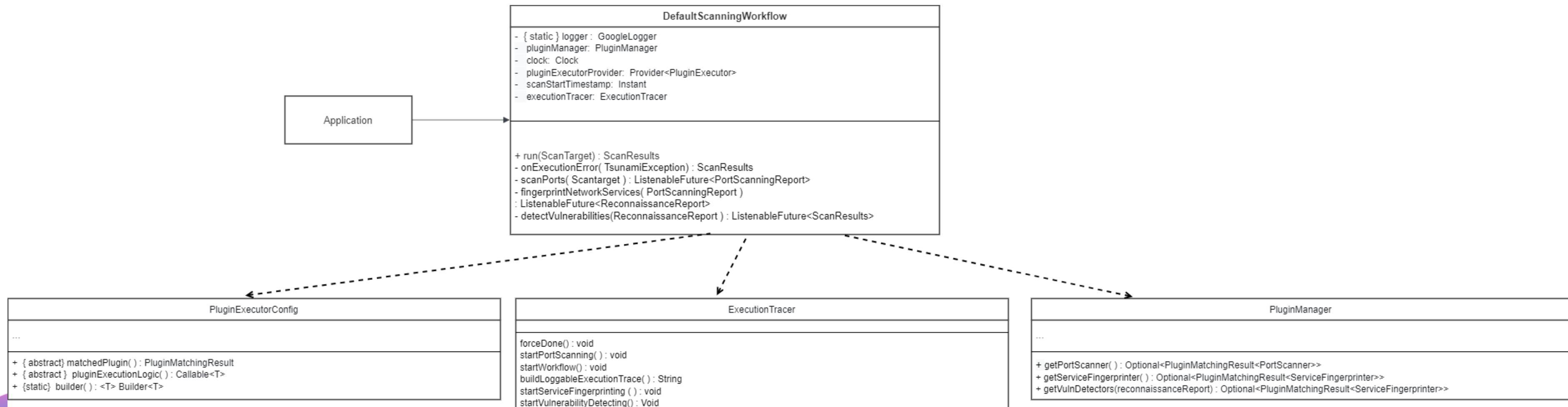
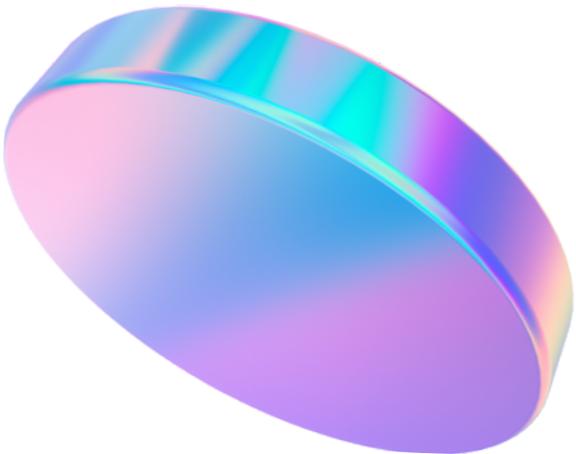
Factory Method

จาก Source Code มีการใช้งาน Factory method
IpValidatorTest ทำหน้าที่เป็น abstract class ให้กับ subclass
IpV4validatorTest กับ IpV6validatorTest แต่ข้างใน
IpV4validatorTest มีการสร้าง object เพิ่มมาซึ่งก็คือ
Ipv4Validator โดย Ipv4Validator เป็น subclass ของ
Validator ที่อยู่อีก folder หนึ่ง

Facade



Facade



Facade

```
public ScanResults run(ScanTarget scanTarget) throws ExecutionException, InterruptedException {
    return runAsync(scanTarget).get();
}

/**
 * Performs the scanning workflow asynchronously.
 *
 * @param scanTarget the IP or hostname target to be scanned
 * @return A {@link ListenableFuture} over the result of the scanning workflow.
 */
public ListenableFuture<ScanResults> runAsync(ScanTarget scanTarget) {
    checkNotNull(scanTarget);
    scanStartTimestamp = Instant.now(clock);
    executionTracer = ExecutionTracer.startWorkflow();
    logger.atInfo().log("Starting Tsunami scanning workflow.");
    return FluentFuture.from(scanPorts(scanTarget))
        .transformAsync(this::fingerprintNetworkServices, directExecutor())
        .transformAsync(this::detectVulnerabilities, directExecutor())
        // Unfortunately FluentFuture doesn't support future peeking.
        .transform(
            scanResults -> {
                logger.atInfo().log("%s", executionTracer.buildLoggableExecutionTrace(scanResults));
                return scanResults;
            },
            directExecutor()
        )
        // Execution errors are handled and reported back in the ScanResults.
        .catching(PluginExecutionException.class, this::onExecutionError, directExecutor())
        .catching(ScanningWorkflowException.class, this::onExecutionError, directExecutor());
}
```

Reference : <https://github.com/google/tsunami-security-scanner/blob/master/workflow/src/main/java/com/google/tsunami/workflow/DefaultScanningWorkflow.java>

Reference : <https://github.com/google/tsunami-security-scanner/blob/545fe11d13bdcba623c129982c1ef2818ffe6199/workflow/src/main/java/com/google/tsunami/workflow/ExecutionTracer.java#L64>

Facade

จะเห็นได้ว่า DefaultScanningWorkflow.java เป็นผู้บังคับหลักสำหรับการทำงานของ Scanning Port ต่างๆ ตั้งนี้จึงมีการใช้งานหลาย Subsystem หลายตัวเพื่อให้สามารถทำงานได้ตาม workflow ดังนี้จึงเหมาะสมกับ คิดของ Facade ที่ช่วยให้สามารถทำงานกับหลายๆ Complex Subsystem

นางสาวนิษกานต์

นายนริชญ์

นายณัฐพล

นายชาติกุล

นายธนรัช

นายณัฐภัทร

THANK
YOU

รายชื่อสมาชิก

62010193 นายชาติถุล รัตนฤทธิ์กุล

62010241 นายณัฐพงษ์ นะสาโร

62010282 นายณัฐภัทร ธรรมกิจเจริญ

62010299 นางสาวณิชกานต์ สุขุมวัฒโนยกัย

62010465 นายนรัชญ์ ออยบัว

62010346 นายธนาธช จุว่อน