Glowstone

จัดทำโดย

นายณัฐพล	บัวบาน	61010345
นายณัฐสิทธิ์	สารกาญจน์	61010362
นายทัณฑธร	สุขสังวาลย์	61010402
นายพศวีร์	และอรุณ	61010707
นายพิพิธพงศ์	จิตภักดิ์ไทย	61010750
นายภูมิไผท	จันทรศรีวงศ์	61010827

เสนอ

ดร.ปริญญา เอกปริญญา

รายงานนี้เป็นส่วนหนึ่งของรายวิชา

SOFTWARE ARCHITECTURE AND DESIGN

ภาคเรียนที่ 2 ปีการศึกษา 2564

Glowstone คืออะไร

Glowstone เป็น Lightweight Open-Source Minecraft Server ที่พัฒนาโดยใช้ภาษา Java ขึ้นมาใหม่ทั้งหมด โดยไม่นำเอาโค้ดภายใน Minecraft มาใช้เป็นรากฐาน (Decompile) ทำให้ไม่มี functionality ที่ไม่จำเป็นจากต้นฉบับ มีประสิทธิภาพที่สูงขึ้นได้ ง่ายต่อการปรับปรุงแก้ไข และยังคงความสามารถในการใช้งาน Plugin จาก Bukkit API, Spigot, Paper, และ fork ต่าง ๆ เหมาะสำหรับเซิร์ฟเวอร์ที่มีต้องการรองรับผู้เล่นจำนวนมาก แต่ไม่ต้องการการปรับแต่งเยอะ

ทั้งนี้ ข้อจำกัดปัจจุบันของ Glowstone คือมันยังไม่เสร็จทั้งหมด ทำให้ยังไม่สามารถรองรับ Plugin ได้ทั้งหมดจริง ๆ และอาจยังมี Bug เกิดขึ้น

สถาปัตยกรรมของ Glowstone

สถาปัตยกรรมของ Glowstone ประกอบด้วย

1. Client-Server Pattern

    ตัว Glowstone เป็นส่วนของ Server ในการนำ Software นี้ไปใช้ และมี Client ต่าง ๆ คือตัวเกม Minecraft client ต่าง ๆ ที่ผู้ใช้จะเข้ามาเล่น โดย Glowstone จะให้บริการต่าง ๆ ทั้งข้อมูลผู้เล่น ระบบ Chat การซิงค์ข้อมูลตำแหน่งและ Block ต่าง ๆ และอื่น ๆ

```
private void bind() {
    if (Networking.EPOLL_AVAILABLE) {
        ConsoleMessages.Info.NativeTransport.EPOLL.log();
    } else if (Networking.KQUEUE_AVAILABLE) {
        ConsoleMessages.Info.NativeTransport.KQUEUE.log();
    }

    CountDownLatch latch = new CountDownLatch(3);

    ProtocolProvider protocolProvider = new ProtocolProvider(config);
    networkServer = new GameServer( server: this, protocolProvider, latch);
    networkServer.bind(getBindAddress(Key.SERVER_PORT));

    if (config.getBoolean(Key.QUERY_ENABLED)) {
        queryServer = new QueryServer( server: this, protocolProvider, latch, config.getBoolean(Key.QUERY_PLUGINS))
        queryServer.bind(getBindAddress(Key.QUERY_PORT));
    } else {
        latch.countDown();
    }

    if (config.getBoolean(Key.RCON_ENABLED)) {
        rconServer = new RconServer( server: this, protocolProvider, latch, config.getString(Key.RCON_PASSWORD));
        rconServer.bind(getBindAddress(Key.RCON_PORT));
    } else {
        latch.countDown();
    }

    try {
        latch.await();
    } catch (InterruptedException e) {
        ConsoleMessages.Error.Rcon.BIND_INTERRUPTED.log(e);
        System.exit( status: 1);
    }
```

2. Microkernel Pattern

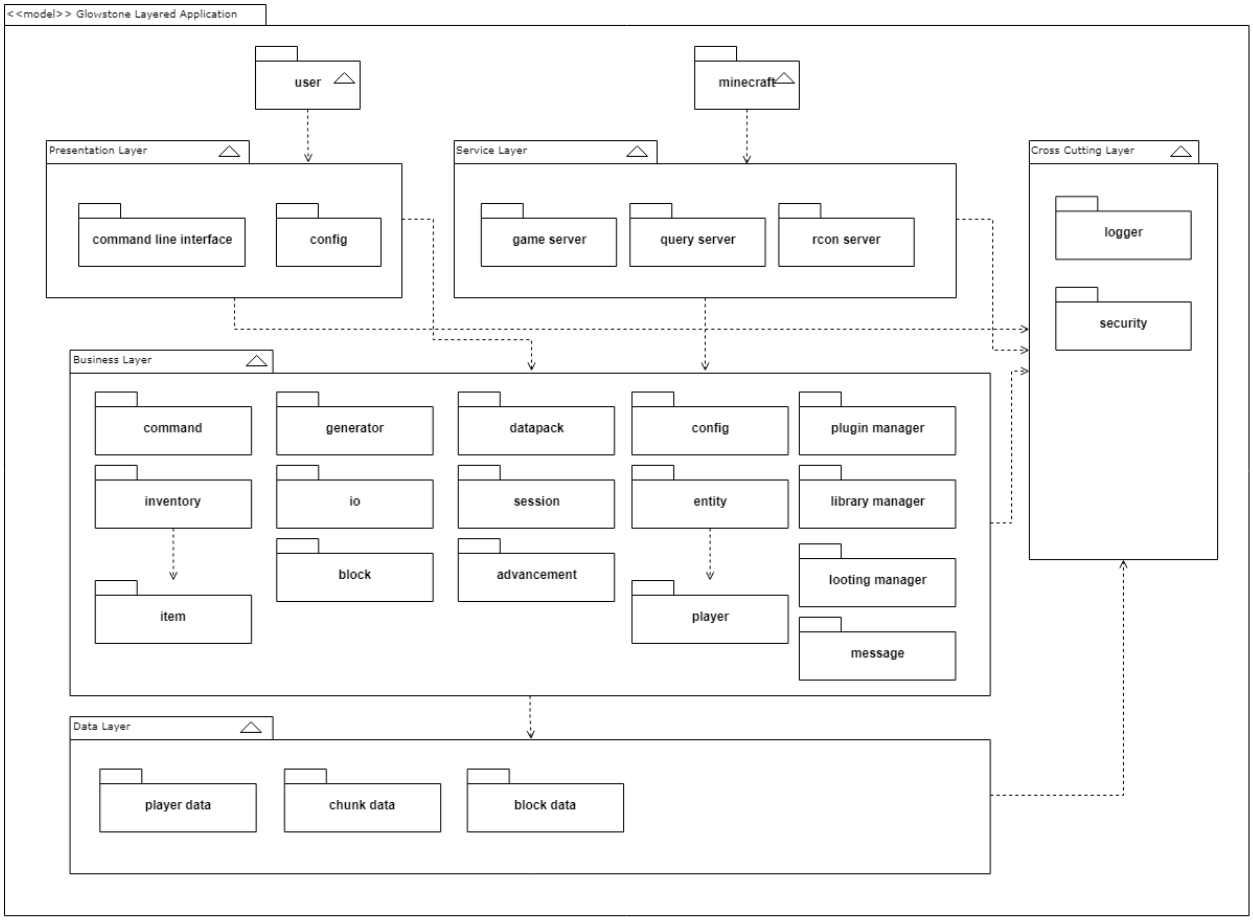
Glowstone ทำออกมาให้รองรับ Plugin และ Library จากแหล่งต่าง ๆ ทั้งจาก Minecraft Vanilla, Bukkit, Spigot และที่สร้างเพิ่มขึ้นมาเอง โดยมี LibraryManager สำหรับ register Library และ SimplePluginManager สำหรับ register Plugin จาก Third Party ต่าง ๆ

```
// Start loading plugins
String repository = config.getString(Key.LIBRARY_REPOSITORY_URL);
String libraryFolder = config.getString(Key.LIBRARIES_FOLDER);
Set<Library> libraries = aggregateLibraries(repository, libraryFolder);
new LibraryManager(repository, libraryFolder,
        config.getBoolean(Key.LIBRARY_CHECKSUM_VALIDATION),
        config.getInt(Key.LIBRARY_DOWNLOAD_ATTEMPTS), libraries).run();
loadPlugins();
enablePlugins(PluginLoadOrder.STARTUP);
```

```
File folder = new File(config.getString(Key.PLUGIN_FOLDER));
if (!folder.isDirectory() && !folder.mkdirs()) {
    ConsoleMessages.Error.Plugin.MKDIR.log(folder);
}

// detect plugin types
pluginTypeDetector = new GlowPluginTypeDetector(folder);
pluginTypeDetector.scan();

// scan plugins for @Field and @Box annotated fields
FieldSet annotatedFields = new FieldSet();
Boxes boxes = new Boxes();
LinkstoneRuntimeData.setFields(annotatedFields);
LinkstoneRuntimeData.setBoxes(boxes);
new LinkstonePluginScanner(annotatedFields, boxes)
        .scanPlugins(pluginTypeDetector.bukkitPlugins);

// clear plugins and prepare to load (Bukkit)
pluginManager.clearPlugins();
pluginManager.registerInterface(LinkstonePluginLoader.class);
Plugin[] plugins = pluginManager
        .loadPlugins(folder, pluginTypeDetector.bukkitPlugins);

// call onLoad methods
for (Plugin plugin : plugins) {
    try {
        plugin.onLoad();
    } catch (Exception ex) {
        ConsoleMessages.Error.Plugin.LOADING.log(
                ex, plugin.getDescription().getFullName()
        );
    }
}
```

3. Layered Pattern

UML Layered Architecture Package Diagram



จุดอ่อนของสถาปัตยกรรมของ Glowstone ประกอบด้วย

1. Single Point of Failure

   Glowstone server เก็บข้อมูลแบบ Stateful คือ data ทั้งหมดถูกเก็บรวมไว้ในที่เดียว ทั้งยังมี instance เดียว หาก Software หรือ Hardware มีปัญหาอาจทำให้บริการทั้งหมดหยุดไป หรือการตั้ง instance ชั่วคราว ทำได้ช้าเนื่องจากไม่มีข้อมูลสำรองกรณีเป็นปัญหาที่ Hardware

   วิธีแก้คือ เปลี่ยนวิธีการเก็บข้อมูลไปใช้ Database แทน และทำ Data Replication ไปที่เครื่องสำรองในเครือข่าย และสามารถเพิ่ม server instance และ proxy server สำหรับสลับ server ให้ client อัตโนมัติ ไว้เพื่อให้มี Availability และ Failure Transparency อยู่ตลอด

2. Not Enough Scalability

   ตัว Glowstone Server เป็น Stateful ไม่สามารถ scale ตัว server ได้กรณีมี inbound request จำนวนมาก

   วิธีแก้คือ เปลี่ยน Glowstone ให้เป็น Stateless โดยการเก็บข้อมูลใน Database แทน

Quality Attributes ของ Glowstone

Quality Attributes ของ Glowstone มีดังนี้

1. Configurability : Glowstone รองรับการตั้งค่าแบบด้วยไฟล์รูปแบบต่างๆเช่น JSON, XML หรือ Text
    - Tactics : Glowstone มีการ detect ไฟล์ตั้งค่าเพื่อนำมาตั้งค่าเซิร์ฟเวอร์โดยไม่ต้อง compile code ใหม่
    - อ้างอิง : [2] [3]

```
public ServerConfig(File directory, File configFile, Map<Key, Object> parameters) {
    checkNotNull(directory);
    checkNotNull(configFile);
    checkNotNull(parameters);

    this.directory = directory;
    this.configFile = configFile;
    this.parameters = parameters;

    config.options().indent(4).copyHeader(true).header(
        "glowstone.yml is the main configuration file for a Glowstone server\n"
            + "It contains everything from server.properties and bukkit.yml in a\n"
            + "normal CraftBukkit installation.\n\n"
            + "Configuration entries are documented on the wiki: "
            + "https://docs.glowstone.net/en/latest/Configuration_Guide/index.html\n"
            + "For help, join us on Discord: https://discord.gg/TFJqhsC");
}

////////////////////////////////////////////////////////////////////////////
// Modification

/**
 * Save the configuration back to file.
 */
public void save() {
    try {
        config.save(configFile);
    } catch (IOException e) {
        GlowServer.logger.log(Level.SEVERE, "Failed to write config: " + configFile, e);
    }
}
```

2. Compatibility

Glowstone รองรับการใช้งานร่วมกับ Bukkit, Spigot และ Paper plugins เพียงแค่นำไฟล์ .jar ไว้ใน plugins directory

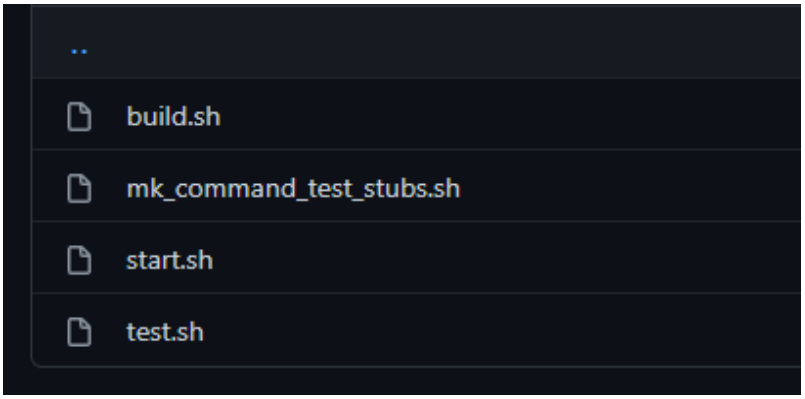
- Tactics : Glowstone มีการ detect ว่าไฟล์ .jar มาจาก plugin ตัวใด เพื่อนำไปใช้ในเกม Minecraft ร่วมกับตัว Glowstone server

```
File folder = new File(config.getString(Key.PLUGIN_FOLDER));
if (!folder.isDirectory() && !folder.mkdirs()) {
    ConsoleMessages.Error.Plugin.MKDIR.log(folder);
}

// detect plugin types
pluginTypeDetector = new GlowPluginTypeDetector(folder);
pluginTypeDetector.scan();

// scan plugins for @Field and @Box annotated fields
FieldSet annotatedFields = new FieldSet();
Boxes boxes = new Boxes();
LinkstoneRuntimeData.setFields(annotatedFields);
LinkstoneRuntimeData.setBoxes(boxes);
new LinkstonePluginScanner(annotatedFields, boxes)
        .scanPlugins(pluginTypeDetector.bukkitPlugins);
```

3. Usability

Glowstone มี CLI ในการใช้สั่งรัน server จากทาง Terminal รวมทั้งการกระทำอื่นๆ เช่น เตะคนออกจาก server

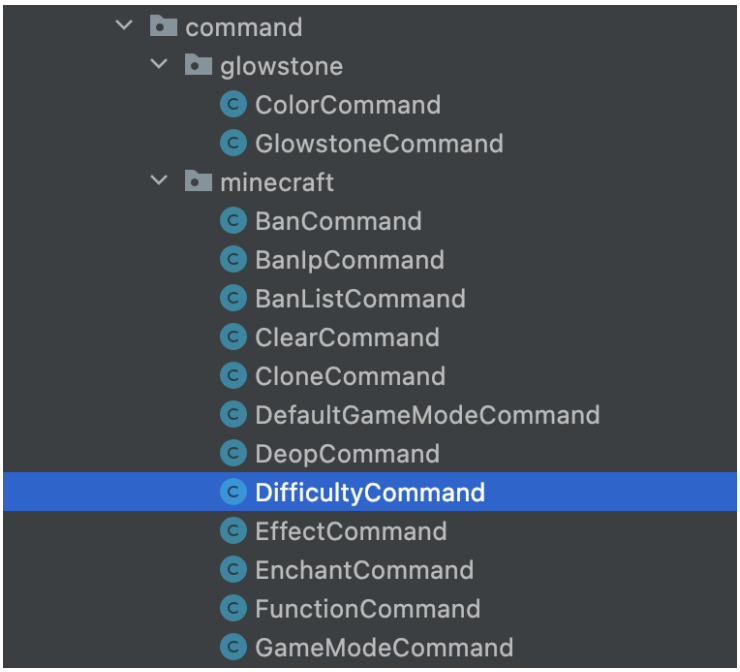
- Tactics : Glowstone มีการ print สถานะการทำงานบน Terminal มี script สำหรับทั้งการ build และ start server



4. Operability

ระบบสามารถสั่งงานและจัดการได้ง่าย

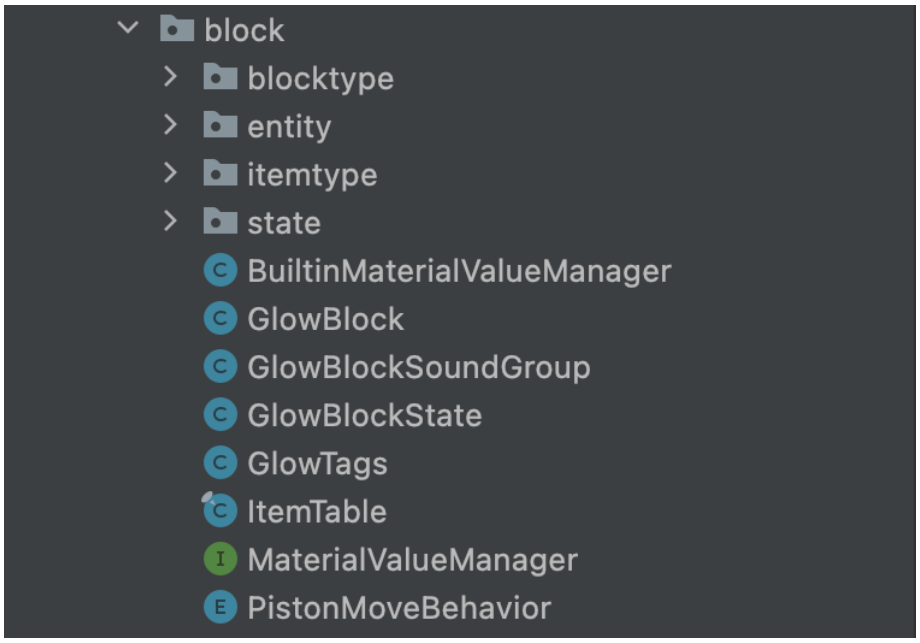
- Tactics : Glowstone มีการสร้างคำสั่งสำหรับให้ admin ควบคุมเซิร์ฟเวอร์สามารถใช้งานผ่านทาง command ภายในตัวเกม
- อ้างอิง : [4]



5. Modifiability

Glowstone มีการเขียน class ที่ยืดหยุ่นเหมาะแก้การที่นักพัฒนาสามารถเพิ่มเติมหรือแก้ไขได้ง่าย

- Tactics : Glowstone มีการเขียน abstract class ของแต่ละชนิดไว้ทำให้นักพัฒนาที่ต้องการเพิ่ม Block ใหม่โดยการ extends abstract class ไปพัฒนาต่อได้ง่าย



Design Pattern ของ Glowstone

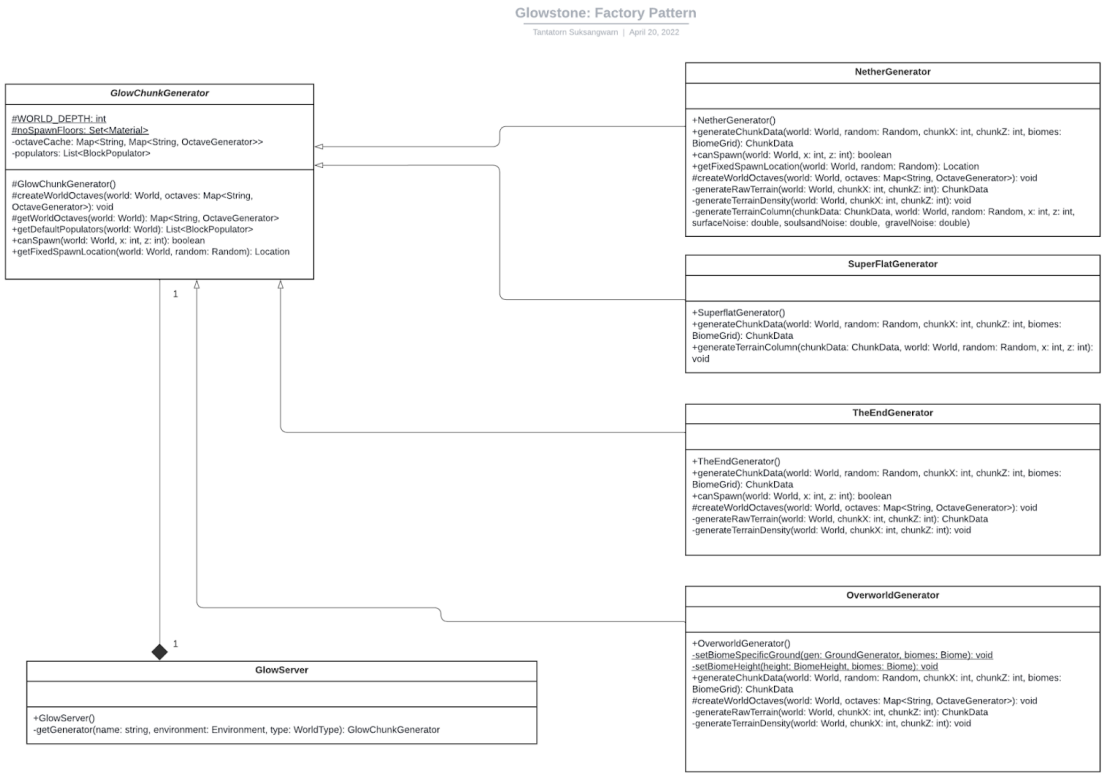
1. Factory Pattern

ฟังก์ชัน getGenerator ทำหน้าที่เป้น factory method สำหรับสร้าง ChunkGenerator ต่างๆ ชนิดกัน ตามชนิดของ world ที่เลือก เช่น Overworld, Nether, Super Flat

บรรทัดที่ 2422:2446 ของ class GlowServer มี method getGenerator ที่จะสร้าง object ตามค่าที่ส่งเข้าไปโดยแต่ละ object ที่สร้างจะเป็น ChunkGenerator เหมือนกัน

ไฟล์ที่เกี่ยวข้อง

- https://github.com/GlowstoneMC/Glowstone/blob/dev/src/main/java/net/glowstone/GlowServer.java



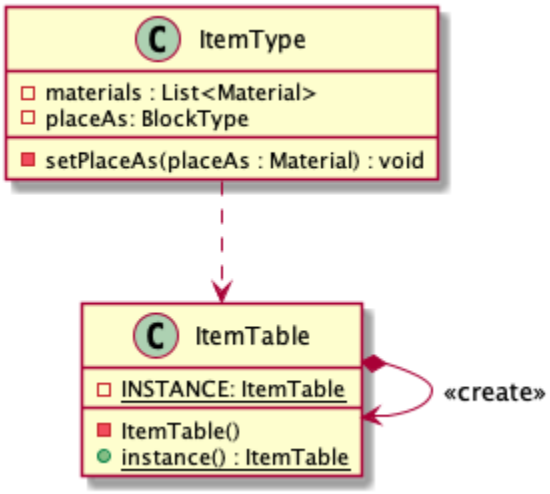
2. Singleton

เป็นการสร้าง object ที่มีได้เพียงตัวเดียว สังเกตจาก Constructor เป็น private ทำให้สร้าง object ที่ class อื่นไม่ได้การจะใช้ต้องเรียกผ่านฟังก์ชัน **instance()** เท่านั้น

บรรทัดที่ 176:178 ใน class ItemTable ที method instance เมื่อเรียกใช้งานจะได้ object ของ ItemTable ที่ทำการสร้างไว้ที่บรรทัดที่ 159

ไฟล์ที่เกี่ยวข้อง

- https://github.com/GlowstoneMC/Glowstone/blob/dev/src/main/java/net/glowstone/block/ItemTable.java




```
private static final ItemTable INSTANCE = new ItemTable();

static {
    INSTANCE.registerBuiltins();
}

private final EnumMap<Material, ItemType> materialToType = new EnumMap<>(Material.class);
private final Map<NamespacedKey, ItemType> extraTypes = new HashMap<>();
private int nextBlockId;
private int nextItemId;

////////////////////////////////////////////////////////////////////////////
// Data

private ItemTable() {
}

public static ItemTable instance() {
    return INSTANCE;
}
```

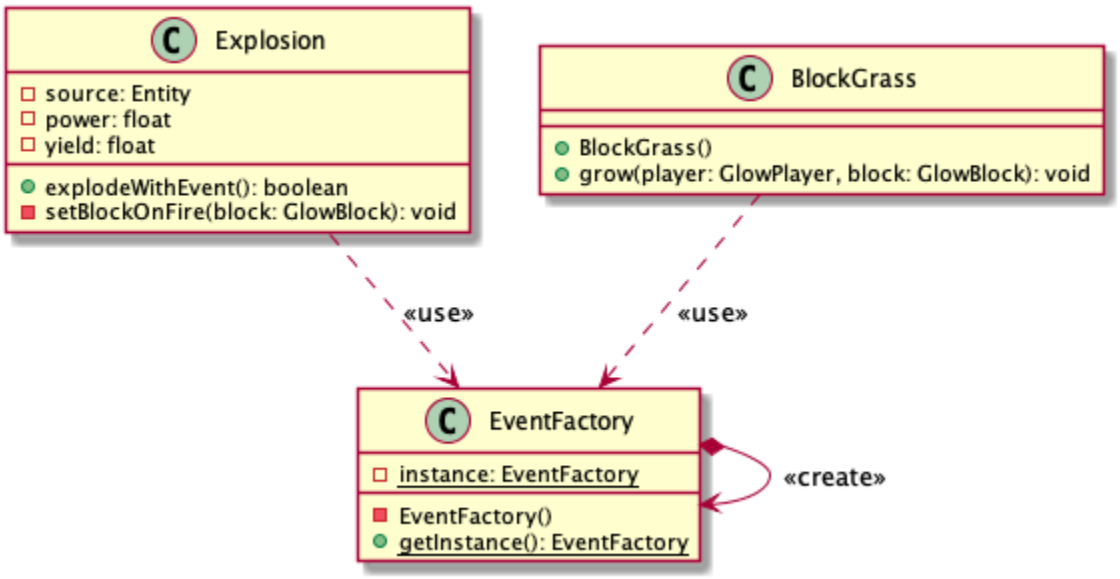
3. Singleton

เป็นการสร้าง object ที่มีได้เพียงตัวเดียว สังเกตจาก Constructor เป็น private ทำให้สร้าง object ที่ class อื่นไม่ได้การจะใช้ต้องเรียกผ่านฟังก์ชัน **getInstance()** เท่านั้น

Class EventFactory ใช้ Singleton Pattern เรียกใช้ object ผ่าน **EventFactory.getInstance()**

ไฟล์ที่เกี่ยวข้อง

- https://github.com/GlowstoneMC/Glowstone/blob/dev/src/main/java/net/glowstone/EventFactory.java



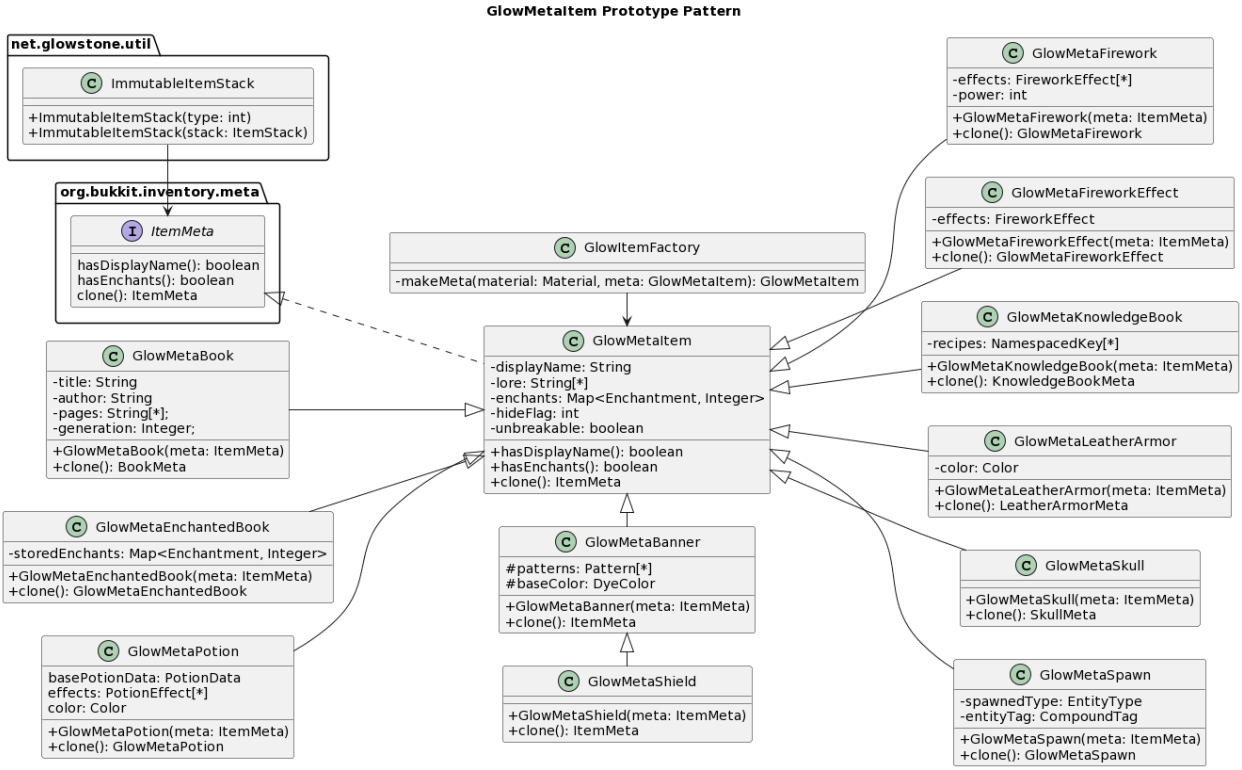
4. Prototype

ใช้สำหรับสร้าง object ซักอย่างซึ่งต้องการให้เหมือนกับ object ต้นแบบ โดยที่ไม่ต้องไปสนใจกับคลาสและตัวแปรของ object นั้น

Class GlowMetaItem ใช้ Prototype Pattern โดยพิจารณาจาก บรรทัดที่ 424 : 426 มี method clone() เพื่อคัดลอก object ออกมาเป็นอีก object หนึ่ง

ไฟล์ที่เกี่ยวข้อง

- https://github.com/GlowstoneMC/Glowstone/blob/2021.8.0/src/main/java/net/glowstone/inventory/GlowMetaItem.java



5. Observer Pattern

ใช้สำหรับส่งข้อมูลการเปลี่ยนแปลงที่กำหนดไปยัง object ต่างๆที่ทำการ subscribe อยู่ โดย ใน Glowstone มีการใช้งานในระบบ scoreboard เพื่อให้ผู้เล่นแต่ละคนสามารถ subscribe และทำการ broadcast ข้อมูลต่างๆที่เปลี่ยนแปลงไปยังผู้เล่นที่ subscribe ใน scoreboard นั่นอยู่

ใน class GlowScoreboard

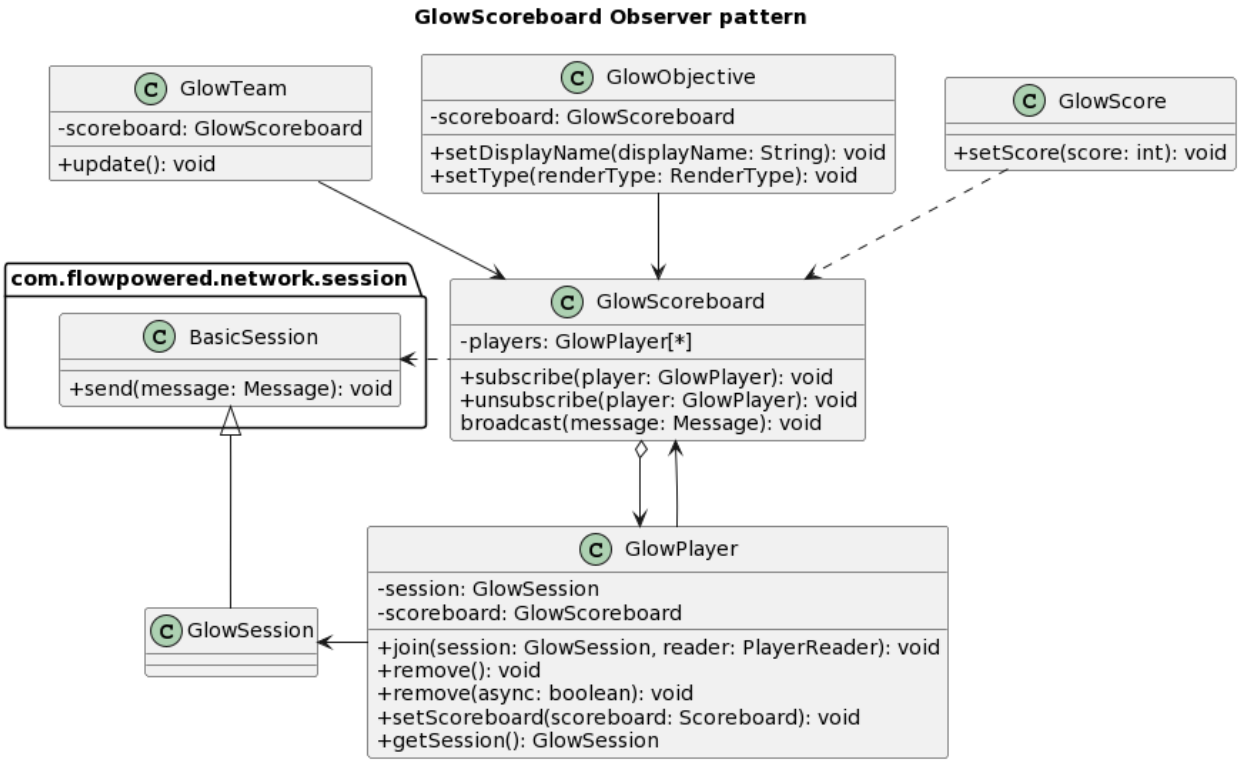
บรรทัดที่ 65:96 มีการสร้าง method subscribe เพื่อให้ player ที่ต้องการ subscribe การเปลี่ยนแปลงมาเรียกใช้งาน

บรรทัดที่ 102:120 มี method unsubscribe เพื่อยกเลิกการรับการเปลี่ยนแปลงที่จะเกิดในอนาคต

บรรทัดที่ 126:131 มี method broadcast เพื่อให้ Class ที่ต้องการเปลี่ยนแปลงข้อมูลส่งข้อมูลให้ player ที่ subscribe อยู่ เช่น การเปลี่ยนแปลงคะแนน การเปลี่ยนแปลงทีมที่ผู้เล่นแต่ละคนอยู่

ไฟล์ที่เกี่ยวข้อง

- https://github.com/GlowstoneMC/Glowstone/blob/2021.8.0/src/main/java/net/glowstone/scoreboard/GlowScoreboard.java



อ้างอิง

1. https://docs.glowstone.net/en/latest/index.html
2. https://apps.dtic.mil/sti/pdfs/ADA610822.pdf
3. https://users.csc.calpoly.edu/~djanzen/courses/common/qualityDefinitions.html
4. https://docs.glowstone.net/en/latest/Getting_Started/basic_admin.html